**User Manual**

**The DTU-DADS-ASF (Version 0.15.1): Technical Description of Framework, Structure and Functions for Modelers**

**By:**

**Tariq Halasa, MSc PhD**

**User guide manual version 10.2015**

**Introduction:**

The DTU-DADS-ASF (Version 0.15.1) is an adjusted version of the model from the DTU-DADS (Version 0.10) (Boklund et al., 2013; Halasa et al., 2014a), which is an upgraded version of the original DADS (Davis Animal Disease Spread) model (Bates et al., 2003). The ASF model of DTU-DADS (DTU-DADS-ASF) is simulating the spread of African swine fever between swine herds through animal movements, abattoir, low risk, and medium risk contacts, and through local spread. The model simulates a set of control strategies including the construction of protection and surveillance zones, forward and backward tracing of direct and indirect movements and contacts, national standstill and surveillance of herds in the zones and from tracing. Furthermore, the model avails an extra function that sets full control on diagnosed herds. This is a useful strategy in case of lack of resource for culling of detected herds. As an alternative control strategy, the model simulates depopulation of susceptible herds within a specific distance from detected herds. A set of extra processes are simulated in the model such as detection and are explained in details in the next sections.

The purpose of this document is to describe the framework, structure and functions of the DTU-DADS-ASF (V. 0.15.1) model. The document will be technically directed that will facilitate the process for other modelers to work with the model. First the framework and structure of the model will be described, and then the different functions will be described in details in a way that will allow modelers to understand the code when they read the description of the code for the specific function. It is important to mention that this document will not focus on input parameters as they are described in another document.

**Model Framework and Structure:**

The model runs in the basic package of the freeware R and hence it does not require downloading extra packages. It consists of 15 code files that include the different functions of the model. The files are:

1. SourceASF.R: This file includes the sourceASF function which sources all files and reads them into R, that R will recognize the functions of the model.
2. ASF.R: This file includes the ASF function.
3. ASFEngine.R: This file includes the ASFEngine function.
4. ASFoptions.R: This file includes the ASFoptions function.
5. initialize.R: This file includes the following functions:
   1. selectIndexHerd.
   2. createASFvars.
   3. initializeASFvars.
6. process\_herds.R: This file includes the following functions:
   1. updateHerds.
   2. DIRinf3.
   3. INDflex.
   4. LASinf.
   5. constructAInfHerd. This function includes a list of functions as described below.
7. controlMeasures.R: This file include the following functions:
   1. SurvZone.
   2. ProtZone.
   3. SurvZonesHerds
   4. controlAll
   5. traceDC
   6. traceIDC
   7. controlDiag
8. opt\_intervention.R: This file includes the following function:
   1. DummyInter.
   2. CullRing.
   3. SurvDead
9. limitMovements.R: This file includes the limitMovements function.
10. Dist.R: This file includes the constructDist function.
11. ConstpMat.R: This file includes the ConstpMat function.
12. chronicle.R: This file includes the constructChronicle function.
13. summaries.R: This file includes the sumTH function.
14. Misc.R: This file includes several ad hoc functions, of which the rpert and RandContacts functions are being actively used in the current version of the model.
15. Graphics.R: This file includes several graphical functions. These functions are not activated in the current version of the model, and hence they are not further documented here.

The model is structured in one global environment that runs within a function called ASF. Within this function different functions are re-scoped to run in the global environment of ASF that they can be called and run from and within other functions. Thus in order to run the model (in the current version), it is needed to source the files by running the sourceASF function, and then index herds are read in R, thereafter the ASF function is run with the pre-defined options, as described below in the ASFoptions function.

The model includes as well several external datasets that are imported into it and are necessary to model running. These datasets are described in Appendix I of this document. Furthermore, the model produces output files. These are explained in Appendix II.

**Description of Model Functions:**

The model consists of several functions as mentioned above. The functions, the way they work and their tasks are:

1. **ASF function**: This function holds the global environment of the model. It is a function of an option called “args”, which is the ASFoptions function. The function starts by importing the args option splitting the arguments in ASFoptions to argument names and values. Thereafter the model will parse textual parameters and functions in the ASFoptions function that R can read them as code, and then all function that needs access from internal variables and functions of the model are re-scoped within the global environment. Graphics and user supplied functions such as the sumTH function are also re-scoped in the global environment. Afterward the infection, basic and optional control functions that are provided in the ASFoptions file are split into argument name and argument, and re-scoped in the global environment to be further processed later in the function. Then the chronicle is constructed by calling the constructChronicle function, the aInfHerd matrix (matrix for within herd disease spread) is constructed by calling the constructAInfHerd, the distance object (Dist) (matrix of distance between infected and susceptible herds) is constructed by calling the constructDist function. The summayFunction (sumTH) and the createASFvars functions are run. Thereafter infection, basic and optional control strategies are initialized and executed. The ASFEngine is then called, which is the function that actually run the different functions overtime. Finally the wrapUp function of the chronicle is called to wrap up the run and the “final” function of the summaryFunction is called to provide the requested outputs of the run.
2. **ASFoptions**: This function includes a list of input variables. These variables can be fixed values, vectors, distributions, matrixes (as lock up tables) and/or functions as textual inputs. A description of the input variable is provided beside each variable inside the ASFoptions file. At the end of the function, it checks for each model run whether the default values were changed when the ASF function was run or not, and if a variable is changed, then it over-writes the default value by the new added value. In this way the user can change the default values as he/she wishes. This is useful for sensitivity analysis.
3. **ASFEngine**: This function is actually the running engine of the model. It runs the model from 1 to n number of iterations. The function starts by initializing the variables that should be used over the different iterations by calling the initializeASFvars function. Thereafter the model runs over days from gTime=1 to the maximum number of days to run as defined by the user (maxTime). Then it determines the vector of infectious herds (infHerdNums). Thereafter the model calls the updateHerds and limitMovements functions. Thereafter the model determines the relative contact of the herd (relDC). This is a vector to determine whether the herds have restrictions on movements and contacts and by how much. Then it starts by initializing the control strategies, if the first outbreak is detected. Then the model will initialize the optional intervention InterMethods if the disease is detected and the optional intervention is activated. Thereafter it updates the vector of infectious herds (infHerdNums) and runs the infection functions (newInfMethods). Then it runs the summary function to make the daily requested and then the per-iteration requested summaries. Finally the model clears the distance (Dist), aInfHerds matrixes and the new infection and optional intervention methods.
4. **createASFvars**: This function creates the necessary data that the model needs to run over the whole running time and iterations. This data will not be re-imported over the different iterations. The model starts by loading the farm file data (aHerd) and loads the ad hoc .csv file herdtypes that includes different inputs categorized per herd type. And then it modifies the herdtypes file in case tornedo plots are used to allow their use and then it parses the input variables of the herdtypes file that R can understand them as code. Then it creates different necessary variables and matrixes and it normalizes the distances probability functions from the different relevant ad hoc .csv files after loading the files. Then it reads the rest of the ad hoc .csv files and globalizes them. Thereafter it transforms the latent and subclinical frequencies from the herdtypes file for the within herd spread of the disease. Thereafter the model creates extra necessary variables and globalizes them and then it makes sure that herds have the correct herd category that matches with the categories in the herdtypes file (just to make sure that categorization is done correctly), and then it determines which herds are eligible for vaccination and which for culling. And then it initializes the Reed frost functions to be used for between herd spread of the disease. And then the model creates the initTimeInfected and initstatus variables that are needed by other functions, and globalizes them. Thereafter the model initiates distributions of number of animals moved per batch and distribute them over the different herd-size categories. This is needed for simulating disease spread through animal movements. Finally the model initializes the matrix (OverLapMat) that includes surveyed herds, if detailed data is called.
5. **initializeASFvars**: This function is called at the beginning of each simulated epidemic (each iteration). This function starts by setting the seed to be equivalent to iteration number if the start seed is < 0 otherwise it will be further processed later. Then the function creates variables that should be re-initialized per iteration and globalize them. Thereafter it distributes the inputs from herdtypes file into all herds in the aHerd dataset based on the herd type categories defined in the herdtypes file and the aHerd dataset. And then it generates herd specific variables to calculate the probability of disease spread within a herd, given one infected animal (Reed-Frost) based on the contacts (K) and the number of susceptible animals within the herd. Thereafter it prepares the infection and control function for initialization, by calling the initialize part of those functions. Thereafter it resets values for different variables, and then it initializes values for the index herd, either directly or using a step in file. The step in file is not being used actively in the current version. Thereafter it records the information of the index herd into the chronicle.
6. **updateHerds**: This function works on the depopulating of diagnosed herds or herds from ring-depopulation that have been set in queue for depopulation in the depopQueue matrix. It starts by removing duplicate ID in the matrix and then it looks up other herds of the same farm to be depopulated. These herds will be processed as diagnosed herds. It creates a matrix (cullMat) which includes information about the herds to be depopulated and their size and category for culling. Thereafter it starts culling herds based on the available resources and the number of animals within the herds. Thereafter the model updates variables about the depopulated herds and simulates one day of intra-herd dynamics.
7. **DIRinf3**: This function simulates the spread infection through direct animal movement between herds. It includes a list of functions that are init, day and cleaniteration. Init is called in the initialization process, but it is here empty and does nothing. Day is the process that occurs daily, while cleaniteration function will clear the pMat to prepare for a new iteration. The function starts by initializing the probability distance matrix (pMat). Then in the day part, first, it determines the number of outgoing contacts per infected herd based on the herd-specific lambda in a Poisson distribution, the restriction probability and the infectiousness of the infectious herd. Thereafter it determines the number of moved animals per batch based on random distributions that are dependent on the herd size. And then it determines the probability that at least one moved animal is infected. Thereafter it determines the number of infectious contacts based on a random process, and then it determines the number of infected animals in each infectious batches and makes sure that there is at least one infected animal in each infectious batch. Thereafter it imports the probability distance information and multiplies it by the risk of disease transmission per herd. And then, per infected herd, it determines the contact herds. The contact herds must have a herd size that is more than the number of animals within the batch, in order to prevent inflation of herd sizes. Thereafter it updates the direct contact trace matrix, which includes the infectious and contact herds and the time they were contacted. And then it determines whether the contact herds will be infected, based on whether the herd is in a restricted zone and the status of the herd. Thereafter newly infected herds are determined and information about them is updated in the intra-herd matrix, the aHerd matrix and the chronicle. The newly infected herds will be set as subclinically infected and information will be updated in the aHerd list regarding for instance time and mode of infection. Thereafter the chronical will save information about the herd.
8. **Indflex:** This function simulates the spread of infection though indirect contacts. It is flexible with different options allowing simulating spread through different kind of contacts. As in DIRinf, the function contains a list of functions that are init, day and cleaniteration. Init and cleaniteration are similar to those in DIRinf. Indflex starts by building the relevant pMat, and then the ”day” part will determine the number of contacts based on the lambda of that contact type and the relative contact of the affected herd in a Poisson distribution. If it is a contact through abattoir truck, then the contacts is re-calculated based on another lambda (user defined) representing the number of herds that will be contacted if a contact would happen. If the disease has been detected, the lambda for those herds will be reduced to the user defined value. Thereafter the number of contacts is determined per infectious herd based on the infectiousness of the herd and the number of contacts that was obtained from the lambda in a Poisson distribution in a random way. Thereafter it imports the probability distance information and multiplies it by the risk of disease transmission per herd. At this stage it is possible to reduce the risk of infection depending on whether this is desired or not and whether disease has been detected or not. The reduction level can be defined by the user. Thereafter, for each infectious herd with a contact, the probabilities of contacting other herds is determined for each herd in the country based on a contact probability matrix between the different herd types (in case it is an abattoir contact, otherwise this factor is dropped), the distance probabilities and the risk of disease transmission. Based on the outcome of these probabilities contact herds are selected, and then the process continues as explained in the DIRinf function above.
9. **LASinf**: This function represents the spread of infection through local area spread. The function is flexible with user defined options regarding the size of the radius of the circle to be covered by this function. The function includes a list of functions that are init, day and cleaniteration. Init and cleariteration are empty. Day function starts by initializing a matrix between infectious herds and the other herds in the country. Then the distances are categorized according to the predefined categories in the ASFoptions and the categories are then replaced by probabilities based on a user defined look up table that represent the risk of infection given the distance from the infectious herd. Thereafter contact herds will be selected based on the probability matrix, infectiousness of the infectious herd, and susceptibility of the contact herds. Thereafter the process will continue as in the previous infection functions.
10. **constructAInfHerd**: This function includes 3 matrixes (called herds, DeadMat and DeadMatSur) that keep track of information about the infected herds. It includes as well a list of functions that updates information of the infected herds and provides information to other functions about the infected herds that can be used in the infection and control processes. The “herds” matrix includes 19 columns and their names and position in the matrix is described in a vector called herdsCol. The DeadMat includes information about the number of dead animals within the infected herds at each day during the past user defined number of days. The number of columns is the number of days and is defined by the “DaysDead” variable, which has an input value in the ASFoptions file. The DeadMatSur is built in the same way as DeadMat, but the number of columns can be different depending on a user defined number in the ASFoptions file called “DaysSurDead”. The function starts by initiating the matrix “herds” and the column names, and then it initiates the latent, subclinical and clinical matrixes. These 3 matrixes include the number of days the individual animals of the infected herds would be latent, subclinical or clinical, respectively. These matrixes have a pre-specified number of columns, in which each column represents number of days. For instance, if the subclinical matrix would have 10 columns then it means, the maximum number of days an animal can be subclinical is 10 days. The 3 matrixes must have the same dimension, but not necessarily the same distribution of the latent, subclinical or clinical periods (the distributions are specified in the typesfile). Thereafter, the function creates a random function (rpoly2) that samples animals from the latent, subclinical and clinical distributions and tabulates them in the latent, subclinical and clinical matrixes, respectively. Thereafter a list of 20 functions is initiated. These functions are 1) getIDs, which provides the ids of the infected herds; 2) getstatus, which provides the status of the herds; 3) getTDiag, which provides the time the herds were diagnosed; 4) getDelHerds, which provides the herds that were removed from the matrix; 5) getHerds, which provides the “herds” matrix; 6) setHerds, which includes herds to the “herds” matrix; 7) getDiagnosed, which provides information about whether herds were diagnosed or not (0 or 1 values); 8) getDiagnosedIDs, which provides the ids of the diagnosed herds; 9) getTClic, which provides the time the infected herds showed clinical signs; 10) setDiagnosed, which sets the infected herds to be diagnosed, once they were diagnosed; 11) getDelIDs, which provides the ids of the removed herds of the matrix; 12) getInfnessDC, which provides the direct contact infectiousness values of the infectious herds (infectiousness of the herds from moving animals). This is dependent on the proportion of the infected animals within the herd. 13) getInfnessIDC, which provides the indirect contact infectiousness values of the infectious herds (infectiousness of the herds from indirect contact to the infectious herds). This is dependent on the number of the subclinically infected animals multiplied by the infectiousness value of subclinical animals, the number of clinically infected animals and the infection pressure from leftovers of the dead animals during the past user defined number of days, all divided the number of live animals in the herd. 14) anyInf, which provides information about the number of herds that are still infected; 15) addInf, which adds newly infected herds into the matrix with default information in order to fill the columns of the “herds” matrix in relation to these newly infected herds. The function starts by checking whether these herds already existing in the “herds” matrix, and includes only those that are not existing, and then it sets default information for these herds in the “herds” matrix. The default information are corresponding to the 19 columns of the “herds” matrix and are temporarily created in “tmp” matrix, that is re-created every time the function is called. Then it distributes the animals on number of days being latent, subclinical or clinical in the respective matrixes. 16) delInf, which removes herds from the “herds” matrix, and from the latent, subclinical and clinical matrixes. This is carried out once the herds are diagnosed and culled. 17) simDay, which simulates the infection process within a herd on daily bases. It starts by initiating a binomial process of animals to be infected based on the number of infectious animals within the herd and the leftovers of dead animals within the past user defined number of days, and then it reduces the number of susceptible animals once some animals became latent following infection. Then it distributes the newly latent animals on the number of days they would be latent, and then it distributes the newly subclinical animals on the number of days they would be subclinical, and then it distributes the newly clinical animals on the number of days they would be clinical. Thereafter, it updates the latent, subclinical and clinical waiting states of the infected animals that are in the latent and subclinical matrixes. Then it updates the number of dead animals within the DeadMat and the DeadMatSur. Thereafter it checks for herds that generally changed the whole status, from susceptible to latent, latent to subclinical or subclinical to clinical. Thereafter, it updates clinical herds to become recovered if all animals in the herd have been infected and non is infectious and the herd has’t been detected or in case the disease faded out in the herd before all animals are infected and after the infectious period of the leftovers of dead animals have elapsed without the herd is being diagnosed. Thereafter, it updates this information in the chronical and the aHerd list. 18) getDead, which provides the number of dead animals during the past user defined number of days “DaysSurDead” for a user defined herd IDs. 19) getInfected, which provides the number of subclinical and clinical animals within the herd. 20) wipe is the last function in the list that empty all matrixes used in the constructAInfHerd function, preparing it for a new iteration**.**
11. **SurvZone:** This function simulates the surveillance zones. It is flexible with several user defined options. The function starts by initializing expressions and parsing input distributions. It consists of a list of functions that are init, day, getIn and getLabel. Init will initialize variables that are used to impose movement and contact restrictions. getIn will provide the herds in the zones, and getLabel, will provide the label of the zone. The day function is the one that runs daily and used to build the zones and assign herds to surveillance visits. It starts by checking whether any zones should be left and does so for those that have been imposed for the defined surveillance zone duration. It then checks whether new detected herds should be included as zones centers and includes them. It then for each center defines the herds that should only be in surveillance zones (herds that are also in protection zones are excluded). Thereafter, it updates the variables that represent the restrictions on animal movements and contacts for these herds. Thereafter it sets herd for surveillance visits. By default, all herds in the surveillance zones will get a second visit (40 days after the initiation of the zone), while it is optional to give a first visit (after 2 days from zone initiation). Herds in overlapping zones will be visited every a user defined number of days, if they are still in a zone. It will then record the herds that are today in surveillance zones. This info will be used the day after for assigning new visits. Thereafter, it will insure that herds that had a scheduled visit because they were in overlapping SZ will not get the visit unless they are still in the SZ. In case detailed data is needed, all herds in the surveillance zones today will be printed out. Finally the function will create a counter for the number of days the herd was generally in a zone and specifically in surveillance zone and when it was first sat in a surveillance zone.
12. **ProtZone:** This function simulates the protection zones. It is flexible with several user defined options. The function starts by initializing expressions and parsing input distributions. It consists of a list of functions that are init, day, getIn and getLabel. Init will initialize variables that are used to impose movement and contact restrictions. getIn will provide the herds in the zones, and getLabel, will provide the label of the zone. The day function is the one that runs daily and used to build the zones and assign herds to surveillance visits. It starts by checking whether any zones should be left and does so for those that have been imposed for the defined surveillance zone duration. It then checks whether new detected herds should be included as zones centers and includes them. It then for each center defines the herds that should only be in protection zones. Thereafter, it updates the variables that represent the restrictions on animal movements and contacts for these herds. Thereafter it sets herd for surveillance visits. By default, all herds in the surveillance zones will get a first visit (after 2 days from zone initiation) and a second visit (45 days after the initiation of the zone). Herds in overlapping zones will be visited every a user defined number of days, if they are still in a zone after receiving the second visit and a user defined number of days. It will then record the herds that are today in protection zones. This info will be used the day after for assigning new visits. Thereafter, it will insure that herds that had a scheduled visit because they were in overlapping PZ will not get the visit unless they are still in the PZ. In case detailed data is needed, all herds in the protection zones today will be printed out. Finally the function will create a counter for the number of days the herd was generally in a zone and specifically in protection zone and when it was first sat in a protection zone.
13. **controlAll**: This function is used to impose 100% restriction on animal movements from all herds (national standstill) for a user defined period of time. It sets the relative direct contact probability of the herds to 0 for the defined period of time. This means no herd would in or out move animals.
14. **traceDC**: This function creates tracing of direct contacts. The function starts by parsing optional inputs and creating expressions to be used as variables. It consists of a list of functions that are init, day and getLabel. Init will initialize variables that are the time needed to trace a contact and the delay time to set the herd in the queue for a visit. The direct tracing consists of backward and forward tracing. In both tracing ways, the function starts by determining the herds that should be traced today. Then it determines the herds that will be traced based on a probability of not forgetting such contact. In both cases of forward and backward tracing, the herds will be set for surveillance visit based on the delay time. This is carried out in the day function of the list, while getLabel provides a label code that informs the type of traced contact.
15. **traceIDC**: This function creates tracing of indirect contacts. The function starts by parsing optional inputs and creating expressions to be used as variables. It consists of a list of functions that are init, day and getLabel. Init will initialize variables that are related to the time needed to trace a contact, the probability of not forgetting the contact and to trace it, per contact type, and the delay time to set the herd in the queue for a visit. The indirect tracing consists of backward and forward tracing. In both tracing ways, the function starts by determining the herds that should be traced today. Then it determines the herds that will be traced based on a probability of not forgetting such contact, per contact type. Thereafter the herds will be set for surveillance visit based on the delay time. This is carried out in the day function of the list, while getLabel provides a label code that informs the type of traced indirect contact.
16. **controlDiag**: This function sets a full control over diagnosed herds. This is important incase resources are not sufficient to cull diagnosed herds within the same day. The function starts by parsing and initializing expressions to be used later as variables. It consists of a list of functions that are init, day, getIn and getLabel. Init will initialize variables that are used to impose movement and contact restrictions. In the day function it imposes the restriction on direct and indirect movements on diagnosed herds. getIn will provide the diagnosed herds, and getLabel, will provide the label of the control measure.
17. **SurvZonesHerds**: This function processes the herds that have been set for surveillance or tracing visits. It consists of a list of functions that are init, day and getLabel. Init and getLabel are empty functions. In the day function, it starts by selecting the herds that should get the second protection visit (PV2) today. It then selects the herds that should get the first protection visit (PV1) today. Thereafter it selects the herds that should get the first surveillance visit (SV1) today, in case first surveillance visit is applied. And then it selects the herds that should get the second surveillance visit (SV2) today. Thereafter it selects the herds that should get the traced indirect contact visit (TIDCV) and the direct traced direct contact visit (TDCV). Thereafter the function randomly selects a proportion of the herds for PV1 and SV1 to be tested and then it arranges the selected herds for visiting and those that have to be tested for testing. Thereafter it sets the selected herds in the queue for visits, removes duplicated herds and removes herds that are set for depopulation, in case ring-depopulation is applied. Thereafter the herds are set to be visited depending on the available resources. Herds that will be clinically surveyed will be recorded and maybe selected to be tested in case of suspicion. Suspicion occurs if the herd has a higher mortality than expected by a user defined number, and at least a user defined number of animals have died. If this is the case, then the herd will be sampled and a user defined number of animals will be select for the sample. The probability of disease detection is then dependent the number of selected animals, the number of subclinically and clinically infected animals, and the total number of animals. The selected herds will then be subjected for a binomial process for detection based on their probability of detection. Detected herds will then be set for depopulation and information about them will be updated in the aHerd list and herds matrix using the aInfHerd list of functions. Herds that must be tested will be tested using PCR and/or serology as defined and the probability of detection will then be calculated for each tested herd based on the visit purpose and the test used. For testing with PCR alone, the detection is dependent on the number of actively infected animals. For detection with serology alone, it is dependent on the number of survived infected animals. For detection using both, then it will be dependent on the number of active infections and the survived infected animals. The number of sampled animals and the total number of animals within the herd are also used to calculate the probability of detection. For herds where the sample size is equal to the herd size and there is at least one infected animal, the probability of detection is set to 100%. Thereafter selected herds will be subjected to a binomial process for detection and the detected ones will be set for depopulation. Information about these herds will then be updated in the aHerd list and herds matrix. The function will then print out all surveyed herds incase detailed data is needed. Thereafter it updated information about the surveyed herds and extends the duration of the protection and surveillance zones in case herds are queuing for longer than the duration of each of the zones.
18. **DummyInter**: This is an empty function that does nothing. However it is needed to prevent errors when calling the optional control functions.
19. **CullRing**: This function selects herds for pre-emptive depopulation. The function starts by parsing and initializing expressions to be used later as variables. It consists of a list of functions that are day and cleaniteration. cleaniteration is an empty functions. A switch function will allow the user to determine whether selection of herds for depopulation should be based on days following disease detection or number of diagnosed herds. In both cases and based on the inputs of the user, the function will select the newly diagnosed herds and measure the distance between each of the herds and the rest of the herds in the farmfile. Thereafter based on the defined radius of depopulation, the herds to be depopulated are selected. The selected herds must be within the selected radius, not depopulated, not detected and are eligible for culling. Thereafter the function updates information about the selected herds and sets them in the culling queue.
20. **SurvDead**: This function implements a strategy in which samples are collected from a user defined number of dead animals with a user defined frequency (number of days) from herds either in the protection zone or the protection and surveillance zones. The samples are then sent to the laboratory for ELISA and PCR analyses. It consists of a list of functions that are day and cleaniteration. cleaniteration is an empty functions. Theday function includes a switch function to choose the samples collection from either the protection zone only or the protection and surveillance zones. By default, it selects from the protection zone only. Whether the selection is from the protection zone only or from the protection or surveillance zones, the process is similar. It starts by daily checking of the herds that should be assigned a submission day, in which in this day, samples collected during the week must be submitted. And then it assign a submission day for the herds that meets the submission criteria, which is herds is the defined zone, not detected and not culled, and do not have a scheduled submission date. Then it assigns a submission day for these herds. Thereafter it checks the herds that had submitted samples and were detected and should be culled and includes them to the culling queue system and update information about these herds in the aHerd matrix. Thereafter it checks which herds should submit samples today. Then it determines the normal mortality in these herds including the infected herds. Thereafter it determines the infected herds that will submit samples and if there are infected herds among these herds, it will determine the number of animals that died during the week because of ASF. Then it will estimate the probability of detection given the number of samples, the total number of dead animals and the number of dead animals due to ASF. Herds that with deaths due to ASF and that had sample size equal to the total number of deaths will have 100% probability of detection. The samples will arrive after a specific delay time (defined by the user), which reflects the way the samples are submitted. The samples will then be tested and the diagnosed herds (random process given the probability of detection) will then be assigned a day to disease confirmation and hence set for culling. This delay time is actually used to inform when the detected herds should be set for culling. The model will actually make the diagnosis at the same day of submission, but will introduce a delay of the action of culling that should reflect a delay on the diagnosis. This manipulation is just a small trick to avoid too much unnecessary coding ;-). Thereafter the model updates the count of the number of tested animals.
21. **limitMovements**: This function decides on detection of clinical herds. It includes 2 elements, the first represents the first detection of the epidemic in the country, while the second represents the basic detection (farmers call the vet who calls the official vets who may diagnose the problem). For the first detection, if the first case is not detected, then it starts by getting all clinical herds and the time they started to show clinical signs. Then it estimates the expected mortality level in the herd given the type of the herd. Thereafter if the mortality level in the herd has increased by more than a user defined value and at least a user defined number of animals have died and the increase in mortality has exceeded a user defined percentage of the herd. This percentage was added based on actual outbreak data from a large herd with good records from Eastern Europe. Thereafter one herd will be selected for first detection of the epidemic and then first detection and detection day are updated. The second part represents basic detection, which is activated only after first detection. The model determines first whether first detection has occurred or not, then if gets all clinical animals and the time they became clinical. Then it determines the expected mortality level and determines the herds that they may be detected. This is dependent on the same inputs as for first detection, but the minimum number of animals to have had died maybe different. Then the selected herds maybe diagnosed with a user defined probability that is set to 1 by default. Thereafter the model sets the detected herds in the depopulation queue and updates information about these herds.
22. **constructDist**: this function creates a distance matrix between a selected herd(s) and the rest of the herds in the country. The function starts by initializing a matrix consisting of 200 columns with a defined number of rows reflecting the number of herds in the dataset. Then it initializes factors that are used to convert degrees to radius and km to miles. A switch function is included to define the distance calculation to be used. calcD is a function that calculate the distance. It includes a switch function that allow the choice whether to calculate the distance based on 1) Spherical (north=latitude, east=longitude), 2) Euclidean using latitude and longitude, and 3) Euclidian using UTM (rectangular) coordinates. The distances are then calculated based on standard method for each of the distance calculation way. addIDs is a function within constructDist that calculate the distances from specified herd IDs to all other locations and put them into the distance matrix. constructDist includes as well a list of functions that are being called from outside. These functions are: get, getNrow, getNcol and wipe. Function get provides the distance between selected herds and the other herds, function getNrow provides the number of rows, function getNcol provides the number of columns and function wipe clears the distance matrix.
23. **ConstpMat**: This function creates the distance-probability matrixes that are used to select newly infected herds by the infection functions; as described above. The function starts by building up the pMat (probability matrix), which consists of a defined number of columns and a number of rows reflecting the number of all herds. It contains the fillCat function which categories the distances into specific categories given by the user. The function contains a list of functions that are: fillProb, addIDs, get and wipe. fillProb starts by calling fillCat to categorize distances in a distance matrix. Thereafter it looks up the probability that corresponds to each distance category and for each herd in the matrix. This is done based on the region and the herd category. addIDs is a function that expands the pMat by first checking whether there are enough columns or not; if not, it will add columns, and then it will add the new IDs to the matrix. Function get gives the distance-probabilities for defined IDs and function wipe clears the matrix.
24. **constructChronicle**: This function stores specific information about changes that occurs to herds during the simulated time and for all iterations. The changes are related to the state of the herd, infection mode, source of infection, herd size and type, day and iteration numbers. The function starts by creating variables that are needed to communicate with the file that will contain the outputs. Thereafter it creates expressions and lists that are needed to store the output data. It contains 2 functions (zero and dump) and a list of functions that are to be called from outside this function (addEntry and wrapup). Function zero creates lists and arrays where outputs will be stored and is called at the initialization stage. Function dump saves the outputs into the defined file, where the data should be saved, and calls function zero. addEntry adds information about changes to the herds into the lists and arrays. This function is being called frequently by other functions, such as new infection functions. The function has basic elements for tracing, but they are not actively used in this version of the model. Function wrapup gathers all information that has been stored by storing them in new expressions, to avoid overwriting, and then puts them in a list called idx, and then save the list as a .bix file.
25. **sumTh**: This function creates the epidemiological and economic outputs of the model. It consists of a list containing 4 functions: init, day, iter and final. Init is a function that is called during the initialization process. It creates expressions of variables and matrixes that should be filled by outcomes either on daily or iteration basis. Function day creates outputs per day. Currently only a matrix of the total number of herds queuing for surveillance is created, when detailed output is requested. This is deactivated in the default options. Function iter creates epidemiological outputs per iteration.

The epidemiological outputs are:

* 1. Epidemic duration **(gEpiDur):** Number of days between first detection and last detected herd is culled.
  2. Number of detected herds (**NumDet**).
  3. Number of infected herds (**NumInf**).
  4. Number of culled herds (**NumCulled**).
  5. Number of recovered herds (**Recovered**).
  6. Number of culled animals (**NumCullAnim**).
  7. Number of Surveyed dead animals (**NumSurvDeadAnim**).
  8. Number of surveyed herds with dead animals (**NumSurvDeadHerd**).
  9. Number of detected herds from surveillance of dead animals (**DetFromSurvDead**).
  10. Number of clinically surveillance visited herds (**ClVisCount**).
  11. Number of serology surveillance visited herds (**SerVisHerdCount**).
  12. Number of PCR surveillance visited herds (**PCRVisHerdCount**).
  13. Number of serology surveyed animals (**SerVisAnimCount**).
  14. Number of PCR surveyed animals (**PCRVisAnimCount**).
  15. Number of herds diagnosed through surveillance (**DiagHerdsSurv**).
  16. First detection day (**FirstEpiDet**).
  17. Extra delay to regain the free status due to delay on OIE meeting (**MeetDelay**).

The function provides as well a dataset including the infected herds, together with their infection time, diagnosis time (if never diagnosed, it gives 0), and the infection mode (the process that resulted in the herd’s infection).

The Economic outputs are divided into direct costs and export losses:

Direct costs are:

1. Surveillance costs is divided to surveillance costs from clinical visits (**VisitsCostsCl**) which was calculated as the sum of the product of the total number of clinical surveillance visits and the costs of a clinical surveillance visit, surveillance costs from serology and/or PCR testing (**VisitCostsTes**). The costs were calculated as the sum of the number of tests multiplied by the costs per testing, plus the sum of the number of visits multiplied by the costs of clinical surveillance, as animals would also be clinically surveyed.
2. Costs of surveillance of dead animals (**SurvCosDead**) were calculated as the sum of the product of multiplying the number of tested samples by the costs of PCR and serology testing plus the sum of the times samples were submitted by the costs of samples submission.
3. Depopulation costs (included costs of logistics, testing and destruction; **logCostsSw**), cleaning and disinfection costs (**CCostsPigFS**, **CCostsPigInd**) were calculated as the sum of the product of multiplying the number of detected herds by the depopulation or cleaning and disinfection costs. Compensation costs (**CompCosts**) are calculated for each detected herd based on the gender and age group of the animals within the herds and the compensation price of each of these animals.
4. Costs of empty stables (**ESCosts**) were dependent on the number of animals per age group in the depopulated herds and the period the herd was empty following depopulation. The period each herd was empty was calculated based on the number of days a herd was in a zone.
5. Costs of welfare slaughter were calculated for herds in the surveillance and protection zones for finishers (**cosWSFin)** and for weaners (**cosWSWean**). The costs were calculated as the number of sows multiplied by a factor (represent the number of produced finishers and weaners per sow per day) multiplied by the duration of time in the zones minus 14 days and multiplied by the value of a finisher or weaner.
6. Costs of the 3 days standstill (**CosSS**) were calculated as a random pert distribution with a minimum, mode and maximum of €4, €5 and €7 x 106, respectively.

Export losses:

Indirect costs included losses due to export bans on livestock and livestock products to EU and non-EU countries. The losses were calculated for live pigs as well as pig products. Losses due to export ban on live pigs and pig products were divided into losses from export ban on: 1) live swine to EU countries, 2) swine products to EU countries, and 3) live swine and swine products to non-EU countries.

1. A total ban on export of live swine to EU countries (**LoExpLivSwEU**) was assumed from the first detection until the last surveillance zone was lifted.
2. Losses due to an export ban on swine products to EU countries were assumed to apply to herds in protection and surveillance zones for the period as long as they were in a zone (**LoExpSwProdEUIZ**). The value of lost export to EU countries was calculated as the percentage of herd-days within zones multiplied by the total daily export values within EU. We assumed that export could be obtained again as soon as the last zone was lifted.
3. Export of live swine and swine products to non-EU countries (**LoExpSwNEU**) was assumed to be ceased from all Danish herds. The value of lost export to non-EU countries was calculated similar to lost export to EU countries, however, with a 3 months delay after culling of the last diagnosed herd and an extra delay to regain freedom of ASF-status. This extra delay is until the OIE committee meets to declare Denmark free from ASF. A random pert distribution with minimum, mode and maximum of 30, 60 and 90 days respectively was used to model the extra delay. Furthermore, a loss of only 25% of the export value was used, as we assumed that these products would be sold locally and on the EU market.

By the end of the function, a list including all outputs prepared by iter function are collected and printed in the working directory with the same ID that has been given to the run.

1. **RandContacts**: This function determines based on a binomial process whether a contact would have happened.
2. **rpert**: This is a random PERT function.

**Appendix I: Input dataset files:**

Several excel (.csv) sheets were added. The sheets and the information included are:

1. MovAb: These files list the probability of a specific herd category to receive an abattoir contact from another herd.
2. SwMovAbProb,: this file represents the distance based movement probability through an abattoir movement.
3. MovSwProb, MovWeProb: these files list the distance probability based risk of a direct animal movement from one herd to another for, sows and weaners, movements; respectively.
4. MovMatAll and MovMatWean: These 2 files list the probabilities of animal movements from a herd type to the other types.
5. MedRiskMovProb and LowRiskMovProb: The 2 files lists the distance based probabilities of the medium and low risk animal movements; respectively, from one herd to another.
6. IndexHerds: this csv file includes the index herds. They are pre-selected based on any criteria the user chooses. These herds are organized in different columns; in which each column represent a criterion of selection. Each herd will be used one time as an index herd in the different runs.
7. DataDADSASF: this file contains all herds’ data. It includes the following columns:
   * 1. ID: represent the herd ID.
     2. BES\_NR: herd number.
     3. CHR: the central herd registration number.
     4. Sows: is the number of sows in the herd.
     5. Finishers: is the number of finishers in the herd.
     6. Weaners: is the number of weaners in the herd.
     7. herdSize: represent the number of animals per herd.
     8. Type: is the type of the herd (text).
     9. herdType: represent the type of the herds,
     10. east: is the easting coordinate of the herd
     11. north: is the northing coordinate of the herd.
     12. LamAll: represent the sow herd-specific lambda.
     13. LamWea: represent the weaner herd-specific lambda.
     14. Region: a categorical variable represents the location of the herd. 1: non-Bornholm herd, 2: Bornholm herds.
     15. herdSizeCull: a variable to represent herd size and will be used when culling of herds is implemented.
     16. herdSizeCat: represent the category of herd size. The categories were selected based on the 25th, 50th and 75th percentiles and the minimum and maximum values of the herd sizes.
     17. ExpMortality: respresents the expected mortality within the herd per day.
     18. NumSamp: the number of animals that will be sampled in case the herd should be sampled for serology and/or PCR.
8. typesfile: this file includes data that will be distributed per herd type to all herds in the DataDADSASF list. It contains the following columns:
   1. herdTypeID: is the ID (value) of the herd type.
   2. herdTypeName: is the explaining text of the herd type IDs.
   3. latDurFreq: is a variable that contains the input distribution of the duration of latent period for the individual animals.
   4. SubDurFreq: is a variable that contains the input distribution of the duration of subclinical period for the individual animals.
   5. CliDurFreq: is a variable that contains the input distribution of the duration of clinical period for the individual animals.
   6. K: is a variable representing the transmission rate of ASF virus between animals within the herd.
   7. RelSusceptibility: is the relative susceptibility of the herds.
   8. RiskDC: is the risk of disease transmission through direct contact.
   9. RiskAb: is the risk of disease transmission through abattoir contacts.
   10. LamMRC: is the lambda of the medium risk contacts (Vets, technicians and AI).
   11. RiskMRC: is the risk of disease transmission through medium risk contact.
   12. LamLRC: is the lambda of low risk contact.
   13. RiskLRC: is the risk of disease transmission through low risk contact.

**Appendix II: Model output files:**

The model produces a standard outputs based on the above summaries. These outputs are:

* + - 1. Summaries per iteration (file name: runID-ASF.txt): runID is a variable used in ASFoptions file to assign an ID for the run. This summaries file includes the following variables:
         1. Column 1: FirstEpiDet: day of first detection.
         2. Column 2: gEpiDur: the true epidemic duration.
         3. Column 3: ObsEpiDur: the observed epidemic duration.
         4. Column 4: NumInf: number of infected herds.
         5. Column 5: NumDet: number of detected herds.
         6. Column 6: NumCulled: number of culled herds.
         7. Column 7: diagHerdsSurv: number of herds diagnosed through surveillance.
         8. Column 8: DetFromSurvDead: number of detected herds through surveillance of dead animals
         9. Column 9: Recovered: number of infected but undetected herds. Became susceptible again.
         10. Column 10: NumCullAnim: number of culled animals.
         11. Column 11: ClVisCount: number of clinical surveillance visits.
         12. Column 12: SerVisHerdCount: number of serology samplings (herd level).
         13. Column 13: SerVisAnimCount: number of animals sampled for serology.
         14. Column 14: PCRVisHerdCount: number of PCR samplings (herd level).
         15. Column 15: PCRVisAnimCount: number of animals sampled for PCR.
         16. Column 16: NumSurvDeadHerd: number of death surveillance events (herd level).
         17. Column 17: NumSurvDeadAnim: number of surveyed dead animals.
         18. Column 18: VisitCostsCl: costs of clinical surveillance.
         19. Column 19: VisitCostsTes: costs of serology and/or PCR surveillance.
         20. Column 20: SurvCosDead: costs of surveillance of dead animals.
         21. Column 21: logCostsSw: logistic costs.
         22. Column 22: CCostsPigFS: costs of cleaning and disinfection.
         23. Column 23: ESCosts: empty stable costs.
         24. Column 24: cosWSFin: costs of welfare slaughter of finishers.
         25. Column 25: cosWSWean: costs of welfare slaughter of weaners.
         26. Column 26: CompCosts: compensation costs.
         27. Column 27: CosSS: costs of national standstill for 3 days.
         28. Column 28: LoExpSwProdEU: export losses of swine products to the EU market.
         29. Column 29: LoExpLivSwEU: export losses of live swine to the EU market.
         30. Column 30: LoExpSwNEU: export losses of live swine and swine products to the non-EU market.
         31. Column 31: MeetDelay: the delay to the meeting of the OIE committee to declare freedom following application.
      2. Details about infected herds (file name: runID-AllInfHerds.txt): This file includes more detailed outputs. It is a data file with 7 columns (undefined names) that are:
         1. Column 1: iteration number.
         2. Column 2: the diagnosis time of the herd, if diagnosed, otherwise an Inf value, indicating that the herd was never diagnosed.
         3. Column 3: the time the herd became immune, if it had become immune, otherwise a value 0, indicating that the herd did not become immune.
         4. Column 4: the ID of the infected herd.
         5. Column 5: the time the herd became infected.
         6. Column 6: the infection mode of the herd, which is the mechanism by which the herd became infected. 1 = Animal movements, 2 = Abattoir movements, 3 = Medium risk contacts, 4 = Low risk contacts, and 5 = Local spread.
         7. Column 7: the infection source, which is the ID of the herd that infected this herd.

The model can provide several datasets that include very detailed outputs per day and iteration. These datasets can be obtained by activating the variable “Detailed" in the ASFoptions file. The files are:

* + - * 1. runID-SurvHerds.txt: This file provides the herds that were in the surveillance zones per iteration and day. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that are in a surveillance zone (not in protection zone).
        2. runID-ProtHerds.txt: This file provides the herds that were in the protection zones per iteration and day. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that are in a protection zone.
        3. runID-DepopHerds: This file provides the depopulated herds. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the depopulated herds
        4. runID-TraceDC.txt: This file provides traced herds for direct contacts. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that are traced for direct contacts.
        5. runID-TraceIDC.txt: This file provides traced herds for indirect contacts. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that are traced for indirect contacts.
        6. runID-PreEmpHerds.txt: This file provides depopulated herds from pre-emptive depopulation. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the depopulated herds from pre-emptive depopulation.
        7. runID-ClSurveyedHerds.txt: This file provides the clinically surveyed herds. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that have been clinically surveyed.
        8. runID-SerSurveyedHerds.txt: This file provides the serologically surveyed herds. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that have been serologically surveyed.
        9. runID-PCRSurveyedHerds.txt: This file provides the PCR surveyed herds. It includes 3 columns, the first is iteration number, the second is the day (time), and the last is the IDs of the herds that have been PCR surveyed.