WildPigABM Documentation and Motivation

Jason Holderieath

2017-06-08

Contents

1 Introduction

This vignette is a companion document to my dissertation essay *Pigs on the wrong side of the fence. An agent-based approach to evaluating the communication of marginal values.* This document contains much of the justification not needed for the story and code to show exactly how the functions work. The vignette is organized by major topic. The Overview, Design, Concepts, and Details section details the relevant features for other agent-based modelers. The subsequent sections document functions and ideas. There is substantially more capability in the model than is on display in the essay and this is apparent upon examination of the functions. The document is designed to fully explain everything about a section within each section. As such, there is substantial overlap between sections. Please cite the model as:

Holderieath, Jason J. 2016. "Public Agent-Based Model - Accounting for interactions and heterogeneity in feral swine policy analysis." doi:10.5281/zenodo.203239.

```
@misc{Holderieath,
author = {Holderieath, Jason J.},
doi = {10.5281/zenodo.203239},
title = {{Public Agent-Based Model - Accounting for interactions and heterogeneity in feral swine policy
url = {https://zenodo.org/record/203239}
}
```

1.1 License

Use of the software implies acceptance of the license.

```
cat(readLines('~/GitHub/Version 2/LICENSE'), sep = '\n')
```

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
```

to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of

the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special,

```
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "{}"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright 2016 Jason Holderieath

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

1.2 Instructions for use:

1.0 Download `NetLogo` Version 5.3.1 `WildPigABM` has not been tested with Version 6.0.1

2.0 Download Source Files

3.0 If your installation is exactly like mine, you can simply install the r package. The `NetLogo` model will be automatically placed in your `R` package library (i.e. `~\R\win-library\3.3\WildPigABM\data`). The `doOne` will know where to find the `NetLogo` model.

```
install.packages("~/WildPigABM_0.2.9.tar.gz", repos = NULL, type = "source")
```

3.1 It is quite likely you will need to modify the `doOne` function to update your generic output directory (`dirc`) and `NetLogo` application path (`nl.path`) must be updated. You may also wish to change other parts of the model. If that is the case, unzip the `tar.gz` file and modify. If you would like to collaborate please contact me through the site.

1.2.1 Use of The source files

If you chose option 3.1, extract `/WildPigABM_0.2.9.tar.gz`. The `/data` folder contains the `NetLogo` model. This model contains most of the functions that relate to the physical world. The `/doc` folder contains the documentation for the version of `RNetLogo` I was working with and documentation for a subset and replace operation that is not used in this version, but could be useful. The `/R` folder contains definitions of all `/R` functions. `/Test` contains scripts for testing this and previous versions. `/vignettes` contains the documentation. Please evaluate these files. If you decide that you would like to collaborate, contact me to *request* (I reserve the right to not grant access to *anyone*.) invitation as a collaborator on a private GitHub repository. Depending on the nature of your project, a version or a fork may be used to give you a private version to develop.

2 Documentation

2.1 Baseline Parameters

Table 1: Baseline Variable Levels

	Baseline Value
allowableOverCap	10
boarTerritoryDecay	1
brange	5
competingWildlife	FALSE
cont	FALSE
dam_Corn	0.02365
dam_CRP	0
dam_soy	0.01715
defaultLitter	0.8
dirc	C:/Users/zejas/Documents/GitHub/Version 2/WildPigABM/inst/examples/output
discount	0.05
effect_comm	0.9
effect_fee	0.8
effect_free	0.65
effect_govt	0.9
EGM_offFarm	100
generic_removal_price	35
gov_budget	1e+05
govLuck	0.9
govRemovalPrice	65
Govt_ON	TRUE
init_hh	45
init_pigs	100
map_height	49
map_width	49

	Baseline Value
max_cycles	10
max_pigs	1440
maxSounderN	100
mergeAccelerator	0.25
mergedistance	100
minPigsFee	10
minSounderN	4
multiYearLand	TRUE
nl.path	C:/Program Files/NetLogo 5.3.1/app
num_choice	12
P_corn	3.71
P_CRP	100
P_feePaying	300
P_soy	9.15
patchSize	10
pig_range	10
sdMult	0
sexRatio	0.25
SounderterritoryDecay	1
splitAccelerator	0.25
srange	5
stochastic	FALSE
swf	1
t_corn	0.5
t_CRP	2.22e-16
t_FEEPAYING	1
t_offfarm	1
t_PAIDREMOVAL	2.22e-16
t_soy	0.5
t_UNPAIDNOT	2.22e-16
t_UNPAIDPRES	1
varCost_corn	3740
varCost_Fee	100
varCost_free	2.22e-16
varCost_offFarm	2.22e-16
varCost_Paid	65
varCost_soy	2360
verboseR	TRUE
verCost_CRP	2.22e-16
wealth	5e+05
Yield_corn	1350
Yield_soy	450
model.path	system.file(“data/FromR_1.0.nlogo”,package = “WildPigABM”)
seed	28128868
init_sounder	5

2.2 Overview, Design Concepts, and Details

2.2.1 Purpose

The overarching purpose of the model is to provide a tool for analysis of feral swine management issues where interactions between people, pigs, and politicians could matter. In the "Core Paper" simulation we are specifically looking at the impacts of expectations forming in decision-making.

2.2.2 Entities, state variables, and scales

This model has a large number of moving parts. The agents are landowners and three types of feral swine. Landownership is defined prior to the simulation. Land is of varying quality. Landowners are characterized by their wealth and activities. Feral swine are characterized by their location, age, sex, and their status as living. Temporal scales are a multi-year simulation with explicit annual time steps. Spatial scales are They exist in a space a little over 6 miles long by 6 miles wide, made up of 2401 ten acre patches.

2.2.3 Process overview and scheduling

1. Initialize.
2. Landowners (or social planners) plan the year.

- Landowners assess their resource base.
- Use LP model to solve for their activities.
- Observer records most of the optimization data pieces.

3. Pig operations are conducted.

- Caginess is updated.
- They reproduce.
- Sounders split and merge.
- Assess their surroundings in preparation for movement.
- Move to a more desired location.

4. Removal operations are conducted.

- Census of feral swine.
- Calculation of patch intensity of effort.
- Calculation of inividual swine probability of removal.
- Death for those swine above the threshold.

5. Landowner year-end operations are conducted.

- Assess damage inflicted by remaining feral swine.
- Calculate updated net income from operations and off-farm work.
- Update wealth of the household.

6. Year-end inventories, clean-up, and save operations are conducted.
7. Simulation-end save operations are completed.

2.2.4 Design concepts

Design concepts have been summarized in the table below.

Design concepts applied to landowners	
Basic Principles	Landowners are complex.
Emergence	Management will depend on outlook
Adaptation	Regardless of the type of the manager, each year landowners reassess their situation.
Objectives	Gross margin maximization over a specified period of time
Learning	No explicit learning is modeled.
Prediction	Landowners are forward-looking into at least the current period, with old information.
Sensing	Landowners are aware of feral swine presence on their property.
Interaction	Landowners interact with feral swine by removing them. Landowners interact with each other indirectly by pursuing goals.
Stochasticity	Landowners are placed by stochastic processes.
Collectives	Only in the social planner simulations, in that case the collective planning process is imposed.
Observation	Wealth, income, decision-making criteria, and plans are retained.

Design concepts applied to government	
Basic Principles	Government is effectively absent for this set of simulations.
Emergence	The machinery is in place for future experiments.
Adaptation	
Objectives	
Learning	
Prediction	
Sensing	
Interaction	
Stochasticity	
Collectives	
Observation	

Design concepts applied to feral swine	
Basic Principles	Feral swine seek to live and reproduce.
Emergence	Movement will be sensitive to group dynamics.
Adaptation	Feral swine move in space away from crowding and removal pressure.
Objectives	Survival, reproduction.
Learning	No explicit learning is modeled.
Prediction	Feral swine are not forward looking.
Sensing	Feral swine are able to sense removal pressure and crowding.
Interaction	Feral swine interact with the other agents by moving away from removal pressure, and by causing damage.

Design concepts applied to feral swine	
Stochasticity	Initial placement is due to a stochastic process.
Collectives	Females are part of groups known as sounders.
Observation	A census of feral swine is kept each time period.

2.2.5 Initialization

Initialization is a fairly complex operation. In order:

- Inside the `doOne` function:
- The seed is specified. (If we had been looking to simulate across a number of different worlds we would have specified a stream of random number seeds in step 2).
- Static variables are defined.
- The `masterSchedule` function is called with the static variables, variables from the grid, and the RNG seed.
- Inside the `masterSchedule` function:
- Matrices and values are calculated for later use (e.g. a null and unit matrix).
- The clock is started for timing simulations.
- NetLogo is loaded into the `R` session using `RNetLogo::NLStart()` and the `NetLogo` portion of the model is loaded using `RNetLogo::NLLoadModel()`.
- If `gui = TRUE` the user is asked to press `[enter]` to continue. This gives the user an opportunity to troubleshoot the loading process.
- The seed is passed to `NetLogo`.
- Initial patch variables are created and passed to `NetLogo`
  - A uniform distribution is used to create initial values for each patch by generating a value in an array. The variables generated are `quality`, four effectiveness variables used in other simulations, and a memory variable defining the tendency of feral swine on a given patch to remember previous control efforts. `Patch_id` is assigned, and control effort in the period prior to the simulation is set to zero.
  - Those variables are passed to `NetLogo` using the `NL_pushToMap` function defined in "Processing Functions."
  - Variable costs, prices, time requirements, and baseline yields are assigned to each patch (non-stochastically) and passed using the `RNetLogo::NLSetPatches` function.
- A dataframe of variables used to define global variables is defined and sent to `NetLogo` using the `initialGlobalVariable` function.
- The `NetLogo` setup function is called.
  - A clean slate is created, along with empty lists for tracking, and default shapes set.
  - `setup-patches`
    - Max pigs per patch defined.
    - Set as ownerless and without pigs, previous removal effort, or income.
    - Yield is adjusted for quality of the land.
  - `setup-seededHH`
    - Randomly place `init_hh` number of households.
  - `setup-hh`
    - Farmstead is built, id number assigned.
  - `setup-landgrab`
    - `init_hh` number of patches with an owner claim an unclaimed neighbor. The process repeats until all patches are claimed.
  - `setup-roads`
    - Roads are created as a visual aid. They serve no other purpose.
  - `setup-boars`

        * A number of boars are created in a random location with age = 0.
    - `setup-sows`
        * A number of sows are created in a random location with age = 0.
        * Sows are then randomly assigned to a group (sounder).
- Result space matrices and lists are created.

2.2.6 Input data

No data files are read into the simulation as described by Railsback and Grimm (2011).

2.2.7 Submodels

Submodels are described in detail in the following sections of the vignette.

2.3 Core Paper Optimization Framework

2.3.1 Release Notes

The optimization process included in versions 0.2.9 and later is the most expansible yet. More emphaisis has been placed on modularity. This modularity is seen in the `masterSchedule` designation of type of price and landowner allowing mixed assignments, removal cost, damage functions, and OCPM functions can all be easily changed. As improvements are made in newer versions this vision of modularity will continue. Future versions will move more and more to a software package that a novice can use. Much of the effort that has gone into this package has been spent on landowner optimization.

Please contact me if you would like to collaborate. The following issues are currently outstanding in a private repository.

- Add support for commercial solver packages In particular, IBM's CPLEX, Gurobi, GAMS, and Mosek solvers all have strengths that could be useful.
- Add GPU Solver Option A GPU solver option would speed up computation greatly - even if it does not speed up an individual optimization, it would allow developers to move many of the calculations off of the CPU and to the GPU speeding up those operations.
- Add support for Non-linear Optimization and Dynamic Programming Improving granularity in the decision-maker's choice space could improve the model. It could slow it down though.

2.3.2 The land-manager's decision

Each type of decision maker uses the same pieces of information to make his decision. Landowners are aware of the presence of feral swine ($pigTerritoryStrength$) and are able to assess associated likely damage, as well as their expected impact for no removal effort (pop_if_none), light removal effort of attempting to remove 25% of the population (pop_if_light), and heavy removal effort of attempting to remove 90% of the population (pop_if_heavy).

2.3.3 Implementation

Implementation between each type of decision is fairly similar. From the `masterSchedule`, `if` statements choose the appropriate optimization function.

```
if(RemovalPriceMethod=="random"){
  a<-c("marginal","minus","linear")
  removalPrFnVec <- sample(a, init_hh, replace=TRUE)
```

```
}
if(RemovalPriceMethod=="linear"){
  removalPrFnVec <- rep_len("linear",length.out = init_hh)
}
if(RemovalPriceMethod=="marginal"){
  removalPrFnVec <- rep_len("marginal",length.out = init_hh)
}
if(RemovalPriceMethod=="minus"){
  removalPrFnVec <- rep_len("minus",length.out = init_hh)
}
if(UserCostMethod=="random"){
  a<-c("myopic","minus","linear")
  userCostFnVec <- sample(a, init_hh, replace=TRUE)
}
if(UserCostMethod=="myopic"){
  userCostFnVec <- rep_len("myopic",length.out = init_hh)
}
if(UserCostMethod=="minus"){
  userCostFnVec <- rep_len("minus",length.out = init_hh)
}
if(UserCostMethod=="linear"){
  userCostFnVec <- rep_len("linear",length.out = init_hh)
}
```

Then inside the time loop.

```
    plan[t] <- RevisedDynamicISH(verboseR, t, now,today,init_hh,
                                             P_corn,P_soy,P_CRP,
                                             varCost_corn, varCost_soy,verCost_CRP,
                                             removalPrFnVec,userCostFnVec,typeDecision,
                                              bounty,
                                             ...)
```

```
#' @rdname WildPigABM
#' @export
removalPrice <- function(x){
  35.404 + (1464)/(1 + (x/3.299004)^1.843566)
}

#' @rdname WildPigABM
#' @export
pctDamFunc <- function(x){ 0.9 + (-0.899)/(1 + (x/14.21)^5)}

#' @rdname WildPigABM
#' @export
incrRemovalPrice <- function(x,y,...){
sum(removalPrice(x:y))
}

#' @rdname WildPigABM
#' @export
minusUserCost <- function(pop_if_none,pop_if,...){
  x <- pop_if_none
  y <- pop_if
  z = x-y
```

```
  (userCostf(y)-userCostf(x))*z
}


#' @rdname WildPigABM
#' @export
estGM <- function(pop_if_none, pop_if_light, pop_if_heavy ,
                  dam_corn,dam_soy,dam_crp,
                  adjYieldCorn, adjYieldSoy,
                  P_corn, p_soy, p_crp,
                  varCost_corn,varCost_soy,varCost_crp,
                  dFnForm,Removal_Price,User_Cost,bounty,...){
  P_crp=P_CRP
  varCost_crp=verCost_CRP
  adjYieldCRP=10
# linDam <- ifelse(dFnForm=="linear",1,0)
# SigDam <- ifelse(dFnForm=="sigmoidal",1,0)
linear <- ifelse(Removal_Price=="linear",1,0)
marginal <- ifelse(Removal_Price=="marginal",1,0)
minus <- ifelse(Removal_Price=="minus",1,0)
ucminus <- ifelse(User_Cost=="minus",1,0)
ucmyopic <- 0
uclinear <- ifelse(User_Cost=="linear",1,0)


K = list("Corn","Soy","CRP")
KK = list("none","light","heavy")
for (k in 1:length(K)){
  for (kk in 1:length(KK)){
    p<-get(paste0("P_",tolower(K[[k]])))
    y<-get(paste0("adjYield",K[[k]]))
    d<-get(paste0("pop_if_",KK[[kk]]))
    vc<-get(paste0("varCost_",tolower(K[[k]])))

    gm <- p*(y-y*pctDamFunc(d))-vc-   #Net Income
      linear*(removalPrice(d)*(pop_if_none-d)) - # linear removal pricing
      marginal*(incrRemovalPrice(pop_if_none,d))- # removal price evaluated at each animal removed
      minus*((removalPrice(d)-removalPrice(pop_if_none))*(pop_if_none-d))+ #removal price is difference
      ucminus*(minusUserCost(pop_if_none,d))-
      ucmyopic*(d)-
      uclinear*(userCostf(d)*(pop_if_none-d))+
      bounty*(pop_if_none-d)

    nam <- paste0("gm_",tolower(K[[k]]),"_",KK[[kk]])

    assign(nam,gm)
  }
}
return(c(gm_corn_none,
         gm_corn_light,
         gm_corn_heavy,
         gm_soy_none,
         gm_soy_light,
         gm_soy_heavy,
```

```
         gm_crp_none,
         gm_crp_light,
         gm_crp_heavy ))
}



#=============================================================================
# All of the landowner decision making functions use the same framework.
# The only difference is the forumulation of the objective function.
# The revised dynamicish function realizes this and provides flexibility for
# mixed levels of rationality.

#' @rdname WildPigABM
#' @export
RevisedDynamicISH <- function(verboseR, t, now,today,
                        init_hh,
                        P_corn,P_soy,P_CRP,
                        varCost_corn, varCost_soy,verCost_CRP,
                        removalPrFnVec,userCostFnVec,typeDecision,bounty,
                        ...) {
    ## NL has all it needs
    RNetLogo::NLCommand('dynamic-prep-step1')
    Sys.sleep(2)
    landownersolution <- list()
    #start looping
    for (j in 1:init_hh) {
      hh <- j - 1
      if (verboseR == TRUE) {print(paste("hh=", hh))}
      patchset <- paste("patches with [owner = ",hh,"]")
      dat <- RNetLogo::NLGetPatches(c(
        "pxcor", "pycor",'pop_if_none',
        'pop_if_light',
        'pop_if_heavy',
        'adjYieldCorn',
        'adjYieldSoy',
        "pigTerritoryStrength"),
        patchset)
      Sys.sleep(2)

      pigTerritoryStrength <- as.vector(dat$pigTerritoryStrength)
      pop_if_none <- as.vector(dat$pop_if_none)
      pop_if_light <- as.vector(dat$pop_if_light)
      pop_if_heavy <- as.vector(dat$pop_if_heavy)
      adjYieldCorn <- as.vector(dat$adjYieldCorn)
      adjYieldSoy <- as.vector(dat$adjYieldSoy)

      #========== eval marginal cost at current, and cost at proposed population subtract
      #fun.6
      #usercost
      #f
      #set gross margins

      obj <- estGM(pop_if_none=pop_if_none,
```

```
              pop_if_light=pop_if_light,
              pop_if_heavy=pop_if_heavy,
              dam_corn=dam_Corn,
              dam_soy=dam_soy,
              dam_crp=dam_CRP,
              adjYieldCorn=adjYieldCorn,
              adjYieldSoy=adjYieldSoy,
              P_corn=P_corn,
              P_soy=P_soy,
              P_crp=P_CRP,
              varCost_corn=varCost_corn,
              varCost_soy=varCost_soy,
              varCost_crp=verCost_CRP,
              dFnForm=NULL,
              Removal_Price=removalPrFnVec[j],
              User_Cost=userCostFnVec[j],
              bounty=bounty)
pc <- nrow(dat)

#Build patch use restriction matrix
#Looks like [I|I...]
pu  <- cbind(
  diag(pc),
  diag(pc),
  diag(pc),

  diag(pc),
  diag(pc),
  diag(pc),

  diag(pc),
  diag(pc),
  diag(pc))
#Overkill at this point, previous versions needed to put together many submatrices
#into a single large matrix
mat <- rbind(pu)
#RHS is all 1s
rhs <- c(rep_len(1,  length.out = pc))
#all eq. are stated as <=
dir <- c(rep_len('<=', length.out = pc))
#all choice variables are continuious
types <- c(rep_len('C', length.out = pc * 9))

indd <- c(1:(pc * 9))
bounds <- list(lower = list(ind=indd,val=c(rep_len(0, length.out = (pc * 9)))),
               upper = list(ind=indd,val=c(rep_len(1, length.out = (pc * 9))))
)

soln <- Rglpk::Rglpk_solve_LP(obj, mat=mat, dir, rhs,
                              bounds = bounds,
                              types = types,
                              max = TRUE,
                              control =
                                list("presolve" = TRUE,
```

```
                                    "tm_limit" = 120000,
                                    verbose = TRUE,
                                    canonicalize_status = FALSE))
solution <- soln$solution
optimum <- soln$optimum
status <- soln$status
spaceFrom <- seq(1,pc*8+1,pc)
spaceTo <- seq(pc,pc*9,pc)
Sys.sleep(1)

dat$a_corn_none <- solution[spaceFrom[1]:spaceTo[1]]
myvars <- c("pxcor", "pycor", "a_corn_none")

transfer <- dat[myvars]
RNetLogo::NLSetPatchSet('a_corn_none',transfer)
Sys.sleep(1)

dat$a_corn_light <- solution[spaceFrom[1]:spaceTo[1]]
transfer <- dat[c("pxcor","pycor","a_corn_light")]
RNetLogo::NLSetPatchSet('a_corn_light',transfer)
Sys.sleep(1)

dat$a_corn_heavy <- solution[spaceFrom[2]:spaceTo[2]]
transfer <- dat[c("pxcor","pycor","a_corn_heavy")]
RNetLogo::NLSetPatchSet('a_corn_heavy',transfer)
Sys.sleep(1)

dat$a_soy_none <- solution[spaceFrom[3]:spaceTo[3]]
transfer <- dat[c("pxcor","pycor","a_soy_none")]
RNetLogo::NLSetPatchSet('a_soy_none',transfer)
Sys.sleep(1)

dat$a_soy_light <- solution[spaceFrom[4]:spaceTo[4]]
transfer <- dat[c("pxcor","pycor","a_soy_light")]
RNetLogo::NLSetPatchSet('a_soy_light',transfer)
Sys.sleep(1)

dat$a_soy_heavy <- solution[spaceFrom[5]:spaceTo[5]]
transfer <- dat[c("pxcor","pycor","a_soy_heavy")]
RNetLogo::NLSetPatchSet('a_soy_heavy',transfer)
Sys.sleep(1)

dat$a_CRP_none <- solution[spaceFrom[6]:spaceTo[6]]
transfer <- dat[c("pxcor","pycor","a_CRP_none")]
RNetLogo::NLSetPatchSet('a_CRP_none',transfer)
Sys.sleep(1)

dat$a_CRP_light <- solution[spaceFrom[7]:spaceTo[7]]
transfer <- dat[c("pxcor","pycor","a_CRP_light")]
RNetLogo::NLSetPatchSet('a_CRP_light',transfer)
Sys.sleep(1)

dat$a_CRP_heavy <- solution[spaceFrom[8]:spaceTo[8]]
transfer <- dat[c("pxcor","pycor","a_CRP_heavy")]
```

```
    RNetLogo::NLSetPatchSet('a_CRP_heavy',transfer)
    Sys.sleep(1)



    if (verboseR == TRUE) {print(paste("LU pushed in period", t, "for HH",j))}
    print(dim(list))
    print(dim(solution))
    print(dim(optimum))
    print(dim(status))
    print(dim(obj))
    print(dim(mat))
    print(dim(dir))
    print(dim(rhs))


    landownersolution[j] <- list(solution=solution,
                                 dat=dat,
                                 optimum=optimum,
                                 status=status,
                                 obj=obj,
                                 mat=mat,
                                 dir=dir,
                                 rhs=rhs)
  }
  mainDir <- "C:/Users/zejas/OneDrive - Colostate/DissertationTopics/ABM/Intermediate_output/"
  subDir <- format(today, format="%Y%b%d")
  dirc <- paste(mainDir,subDir,"/",sep = "")
  dir.create(file.path(dirc), showWarnings = FALSE, recursive = FALSE)
  nam <- paste(dirc,"decisionReturn",t,j,"_",typeDecision,".rda", sep = "")
  save(landownersolution,file = nam)
  return(landownersolution)
}
```

2.4 Opportunity Cost of Postponement of Management (OCPM)

A technique needs to be developed to account for costs imposed by an animal surviving into the next planning period. Expectations about prices have been handled a variety of ways (Shi and Irwin 2005; Thijssen 1996), however feral swine are living and reproducing organisms making forming expectations of their impact more nuanced. Feral swine move, reproduce, die, cause damage according to a non-linear function, and are increasingly expensive to remove as the population is driven to zero.

A simple example of this phenomena near to the hearts of spring residents of Northern Colorado, is spring allergies. Many of us suffer from spring allergies, and maintain a tolerable level of inflamation with some over the counter antihistimine. The antihistimine is relatively low cost and maintains adequate control. We could get shots and cure ourselves – that would be very costly in terms of both time and money and the discomfort of recieving injections. If we fail to control our allergies there can be much more severe and costly conditions, such as pnemonia, that can arise. It is also concievable that our willingness to pay for injections is constrained by some factor currently, and in future periods injections may be the optimal form of management.

Like the allergies, feral swine impact producers differently as a function of population. If the population gets out of hand, it can be very damaging and very costly to bring back into an acceptable level of damage. Differing levels of control can be optimal given different circumstances, and an optimal management plan

may include letting a given animal live another period. The added level of complexity that feral swine carry is that they reproduce, and progeny will very quickly be inflicting damage. The following paragraphs will explore the necessity, mathematics, and estimation of what we will coin, "Opportunity Cost of Postponement of Management (OCPM)". We draw inspriation from, but do not necessarily emulate the concept of "user cost" from the natural resources literature relating to extraction of a resource (see Berck and Helfand 2011, ch. 16 & Field (2001), ch. 10). In that literature, the manager is concerned with not over- or under-extracting a resource in order to balance costs (especially opportunity costs) and benefits across time.

If one were to simply find the net present value or discounted sum of future damages (see Moss 2013, ch. 5), the effect of progeny would be missed. If damage was inflicted linearly (such as in Zivin, Hueth, and Zilberman (2000)), we could use a formula for present value of a perpetuity, $PV = \frac{PMT}{r-g}$, where PV is present value, PMT is the annual cash flow inflicted by feral swine, r is the discount rate, and g is the annual population growth rate. However, while the exact functional form is unknown, a linear damage function does not follow the narrative that we hear from producers and does not make logical sense.

A non-linear damage function fits the known behavior of feral swine. First, the damage function must be able to handle populations greater than estimated carrying capacity. In Zivin, Hueth, and Zilberman (2000) the function is based on full damage of 90% being carried out by a population of 15 animals.

> Damage to rangeland, measured as a percentage loss of total revenue, is modeled as a linear function of pig population density (Barrett [16]):
>
> $$D(N) = cN$$
>
> Data and information on rangeland damage were collected from a number of studies (Barrett et al. [10], Sterner and Barrett [12], Barrett [17]). Based on this information, and through conversations with Barrett [16], c was set equal to 0.06. This implies that at full carrying capacity (15 pigs/km2), total rangeland damage is 90%. Because there is considerable uncertainty associated with this parameter, we vary this coefficient and examine its impact on steady-state results. (Zivin, Hueth, and Zilberman 2000, 197–98)

It is also known that based on a regression of change in damage by Hone (1995) that a non-linear damage function fits a change in damage measure. This approach should be parameterized for the United States, but certainly demonstrates the shape of what the damage function should be in a more complex representation. Hone (1995) found that there is a curvilinear relationship between the extent of rooting and freqency of rooting and feral swine density. The plots below demonstrate the curvature found when a measure of pig density (x) with one more measure for each line is regressed on the change in frequency of rooting.

Additionaly, a linear reduction in population is almost certainly not optimal from an optimal management perspective. This is what is found here, discussed later in the estimation portion of this discussion, and is typical in these types of problems. A reasonable followup question to this statement is, why not just use a planning technique that will produce optimal management?

We want to abstract away from an optimal management scenario to test differing levels of adherance to expert opinion. For instance, if one were to hire a consultant or ask for assistance from an extension agent about their feral swine problem, they would likely not be presented with a fully optimal management plan that accounts for movement, reproduction, resistance to control measures, the particular damage function of that farm, etc. The producer would be presented with a rule of thumb or may be presented something more complex and then act on it inappropriately. As discussed earlier, one person's mismanagement or refusal to cooperate will have consequences beyond their property borders. If we can condence the essence of an optimal management plan and then test different implementations of that management plan we can address this communication issue.

2.4.1 Presentation of the Mathematics

The central idea behind optimal management of feral swine is that across both time and space producers will choose to apply the input (control) to the point at which the value of its marginal product is just equal to

its per-unit market price. The input is control, or a desired remaining population. The value of the output of this input is avoided damage - from this animal and its progeny.

2.4.1.1 Control Costs

As mentioned earlier, the removal of feral swine is increasingly difficult as the population approaches zero. The most extreme statement of this is:

$$\lim_{n \to 0} w_{control}(n) \ \to L.$$

There have been successful eradication campaigns, so the price of removal to zero is not infinite. Saunders and Bryant (1988) proposed a removal cost function based on flight time used in an exercise in Australia,

> p = 1 - 1.177exp(0.001375t). With one helicopter we can assume an average of 400 min flying time per day. To remove 95% of the population would have therefore taken the exercise to 7 days (2297 min). Similarly, a 99% reduction would take 9 days (3468 min). (Saunders and Bryant 1988, 78)

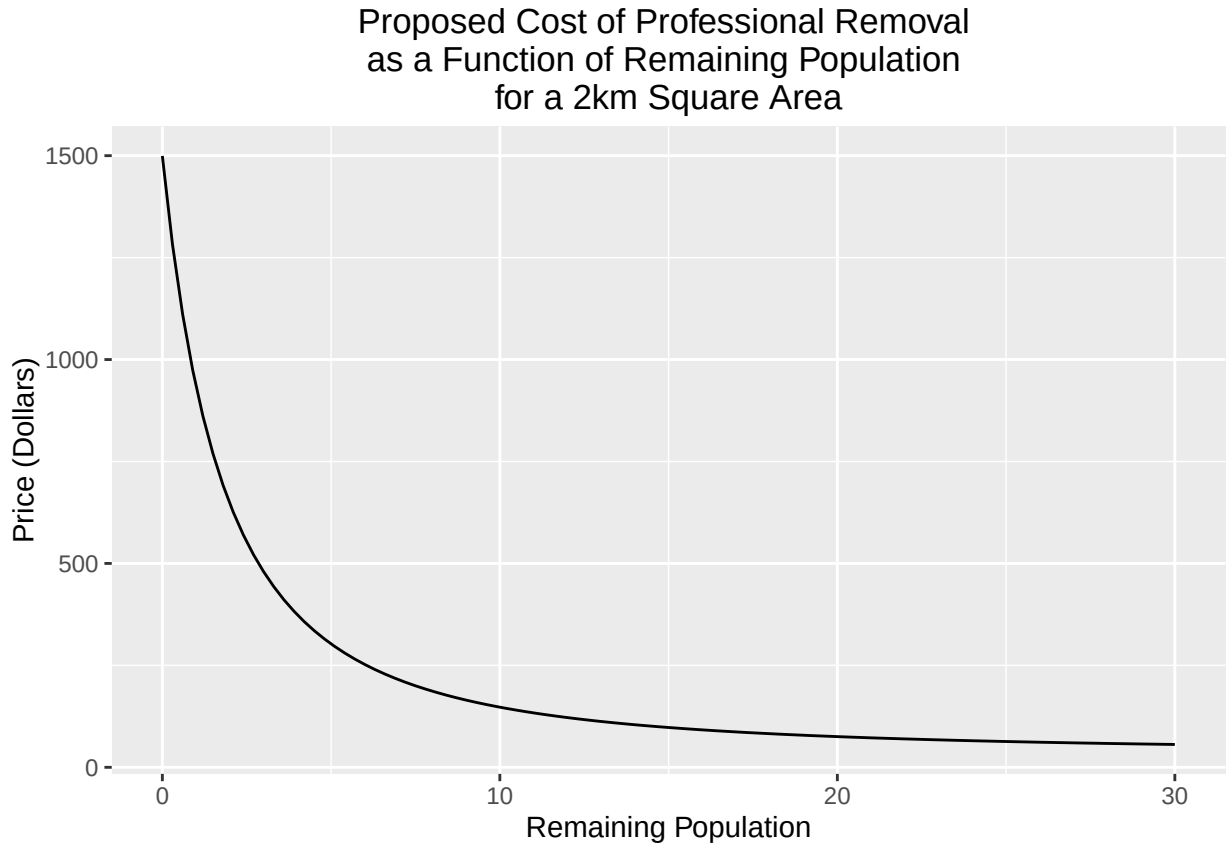
Wildlife managers understand the increasing cost of removal, but they also understand the need to communicate average costs. That is, we know that we are currently removing something close to 20-30% of the feral swine in Texas (Carson 2013) and mixed removal actions in Texas cost approximately $65/head (Bodenchuk 2014). If we solve the equation proposed by Saunders and Bryant (1988) for t, we find:

$$t = 8000/11 \times ln(-100/177 \times (p-1)) \ for \ p < 1.$$

However, what we really need is a pricing mechanism for professional removal that reflects the increasing difficulty of removing as the population reaches zero. A function like the one below reflects the price we currently face with abundant populations and approaches the theoretical price curve proposed by Saunders and Bryant (1988).

$$w_{control} = 35.404 + (1464)/(1 + (x)/3.299004)^{1.843566})$$

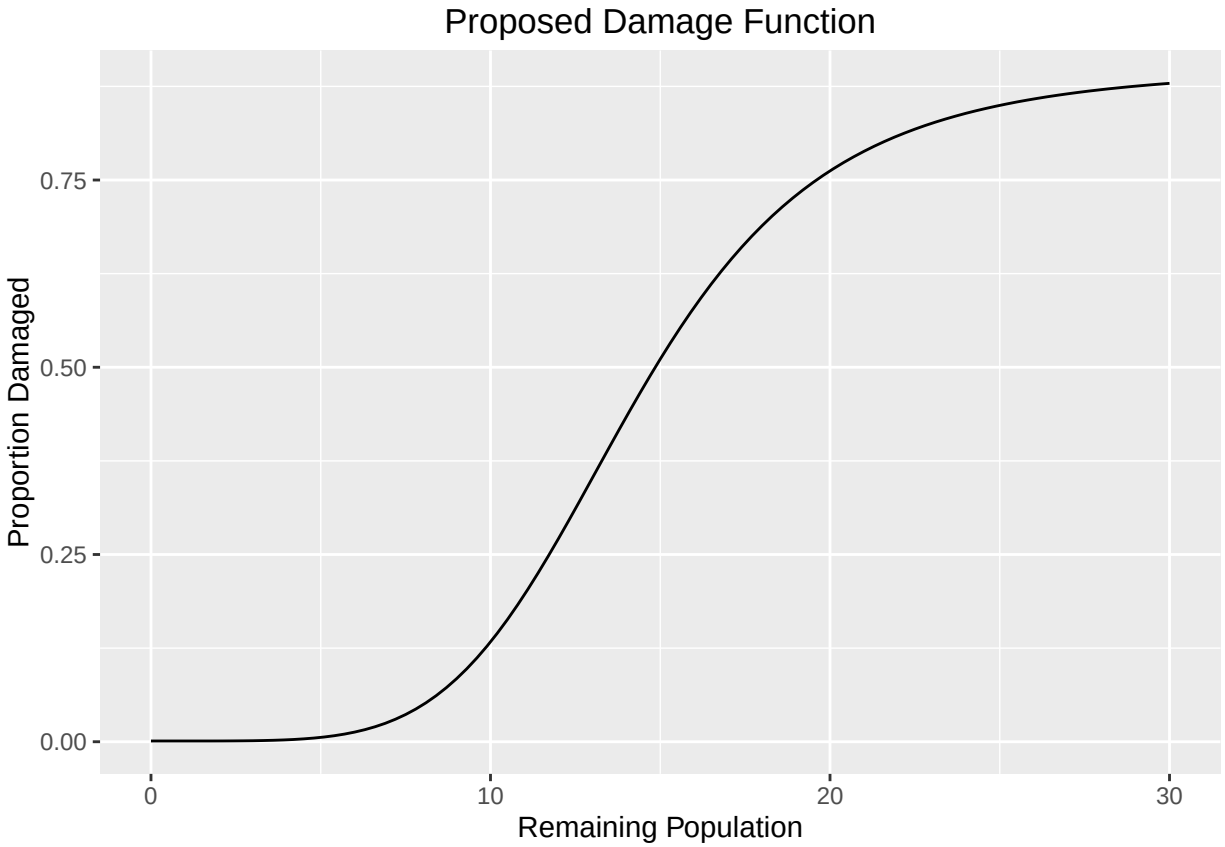
Plotted, this function retains the shape of Saunders and Bryant (1988) but instead works off of the expected remaining population and calibrates to approximate twenty percent removal at approximately $65/head and removal of the last animal at approximately $1500.



2.4.1.2 Damage

As discussed earlier, there is a maximum level of damage and that when the population changes, damage changes non-linearly. These ideas can be fitted to a sigmoidal curve. A symmetric sigmoidal curve set to be very near zero at a population of zero and to max out at 90% damage at a full population of 30 head in the 2km^2 area.

$$y = 0.9 + (-0.899)/(1 + (x/14.21)^5)$$



2.4.1.3 Marginal Conditions

Both control and damage are a function of remaining population, rather than control effort. In addition, benefit is an avoided cost. They are two sides of the same coin, however, a subtle shift in interpretation of the marginal conditions is required. First, find the marginal damage for corn and soybeans.

$$\text{differentiate } damage_{corn} = (0.9 - 0.899/(1 + (x/14.21)^5)) \times 1350 \times 3.71 \text{ w.r.t. x}$$

$$damage'_{corn}(x) = (1.30439 \text{»} 10^{10} \times x^4)/(x^5 + 579389)^2$$

$$\text{differentiate } damage_{soy} = (0.9 - 0.899/(1 + (x/14.21)^5)) \times 450 \times 9.15 \text{ w.r.t. x}$$

$$damage'_{soy}(x) = (1.07234 \text{»} 10^{10} \times x^4)/(x^5 + 579389)^2$$

Then the cost of achieving a remaining population. There are more benefits to removal than the single season. We should include future benefits. It is unclear how long a decision-maker will plan, however, a perfect planner would look forward infinitely. The present value of a perpetuity is:

$$PV = PMT/r$$

where PV is present value, PMT is a payment, and r is a dicount rate.

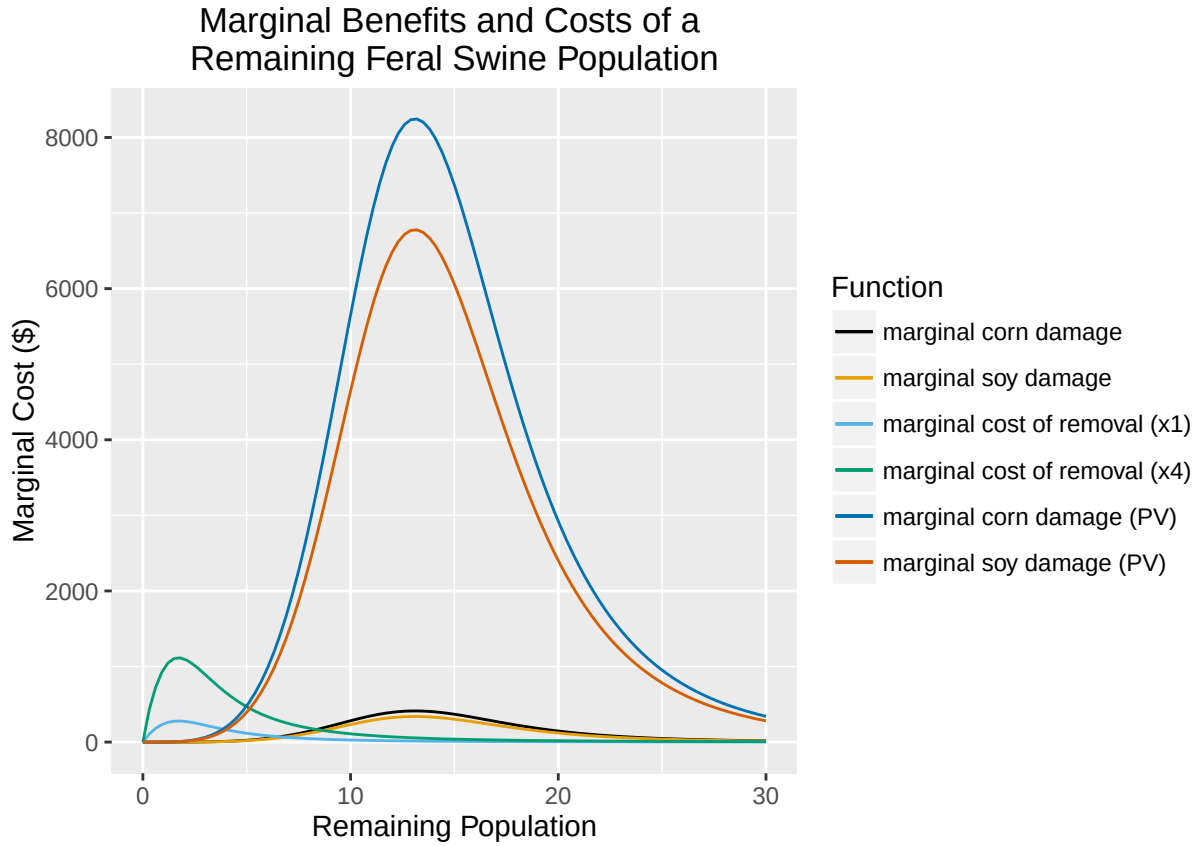
$$damage'_{corn,perpetuity}(x) = (2.60878x10^{11}x^4)/(x^5 + 579389)^2$$

$$damage'_{soy,perpetuity}(x) = (2.14469x10^{11}x^4)/(x^5 + 579389)^2$$

The cost of removal function can be differentiated:

$$removal'(1x) = (24371 \times x^{0.843566})/(x^{1.84357} + 9.02969)^2$$

$$removal'(4x) = (97483.9 \times x^{0.843566})/(x^{1.84357} + 9.02969)^2.$$



The plotted functions show marginal cost of a remaining population. Again, this graph does not show the demand for or the cost of a given unit of control. It shows the costs and and benefit of a remaining population. Clearly, the myopic planner and a forward looking planner will arrive at different outcomes. Even this much higher function might be an understatement. These marginal present value functions do not consider reproductive capacity of a single animal. The dynamic model presented in the next section will account for the progeny of an animal.

2.4.2 Estimation

To estimate the essence of an expert we simplify the problem to dimensions that can be understood on a per animal basis. A non-linear programming (NLP) routine is used to find optimal management of a smaller

problem. This optimization routine was implemented in `GAMS`. The cost of an additional animal in future periods is then distilled to a function for use in the agent-based model.

The model philosophy was changed slightly to reflect a move toward (but not completely achieving) optimal management. A single decision-maker maximizes gross margin for a subset of the larger map. The world is reduced to the assumed range of an animal of approximately 2 km^2 (see Schlichting et al. 2016). The 490 acres of this world are managed as one unit, with no movement into or out of the area. This implicitly assumes that everyone will engage in the same actions. The model is estimated for a 75 year planning horizon, sufficiently long to make terminal conditions irrelevant.

The same crop prices, yields, and variable costs are used. Quality is the average for the subset of patches included.

Initial conditions are established, creating a state with an initial population ,as well as no removal, no land use, or damage in the initial period. Initial population, set initial population at a variable level for analysis.

$$initPigs = rP_{n1,t} \; if \; ord_t = 1$$

Intitial removal efforts are restricted to the level of the initial number of animals.

$$q_{n1,t} < initPigsif \; ord_t = 1$$

To avoid complete removal at initialization, and preventing the harvest of a marginal value of a remaining animal, Initial removal price is set very high.

$$P\_removal_{n1,t} = 10000 \; if \; ord_t = 1$$

Population dynmaics are essentially previous population plus growth minus removed equals ending population.

$$rP_{n1,t-1} + (rP_{n1,t-1} \times gamma \times (1 - (rP_{n1,t-1} maxD^{-1}))) - q_{n1,t} = rP_{n1,t} \; \forall \; t \; if \; ord_t \neq 1$$

Removal is restricted to the number of animals present at the beginning of a time-step.

$$q_{n1,t} \leq rP_{n1,t-1} \; \forall \; t \; if \; ord_t \neq 1$$

A symmetric sigmoidal curve set to be very near zero at a population of zero and to max out at 90% damage at a full population of 30 head in the 2km^2 area.

$$dam_{n1,t} = 0.9 + (-0.899)/(1 + (rP_{n1,t}/14.21)^5)$$

Price of removal is increasing at an increasing rate as the remaining population approaches zero and is an implementation of the removal function developed above.

$$P\_removal_{n1,t} = 35.404 + (1464)/(1 + (rP_{n1,t}/3.299004)^{1.843566})$$

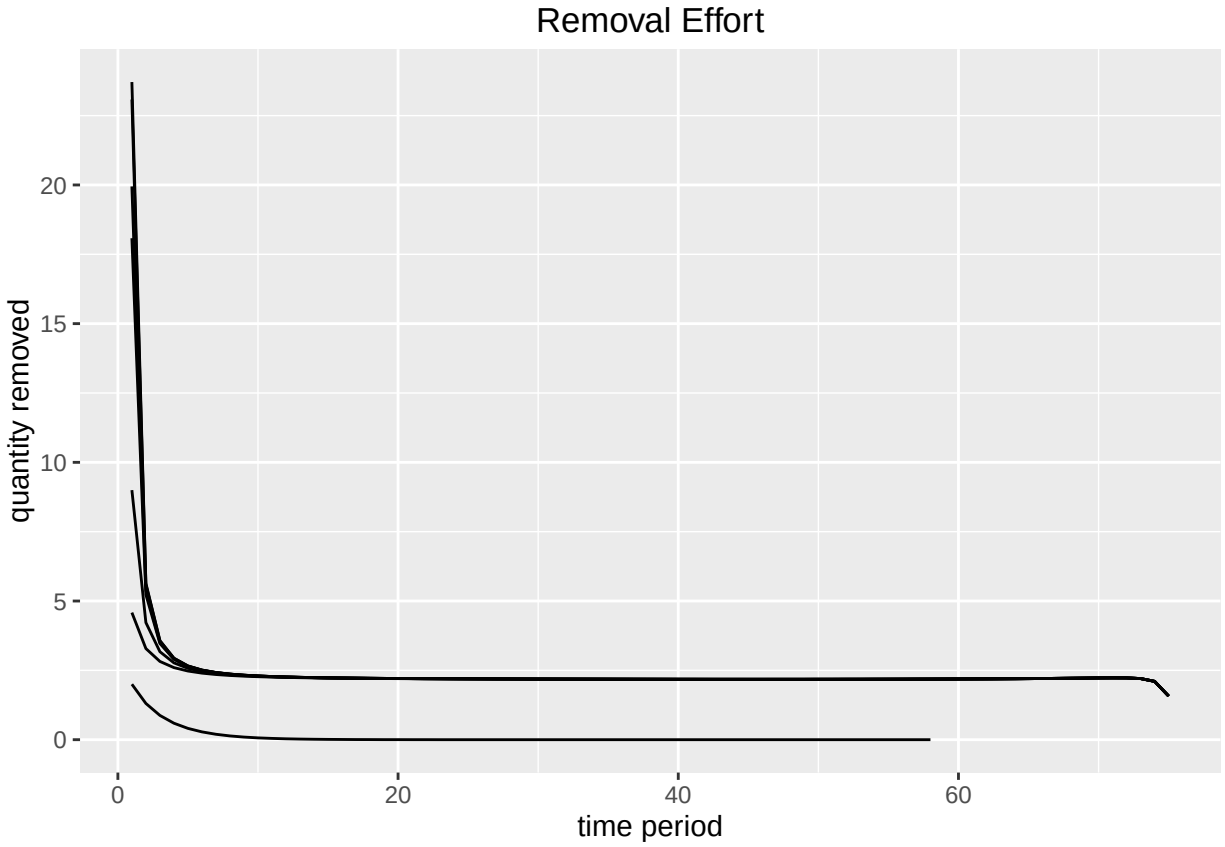
Patches are restricted to a single use in a given season.

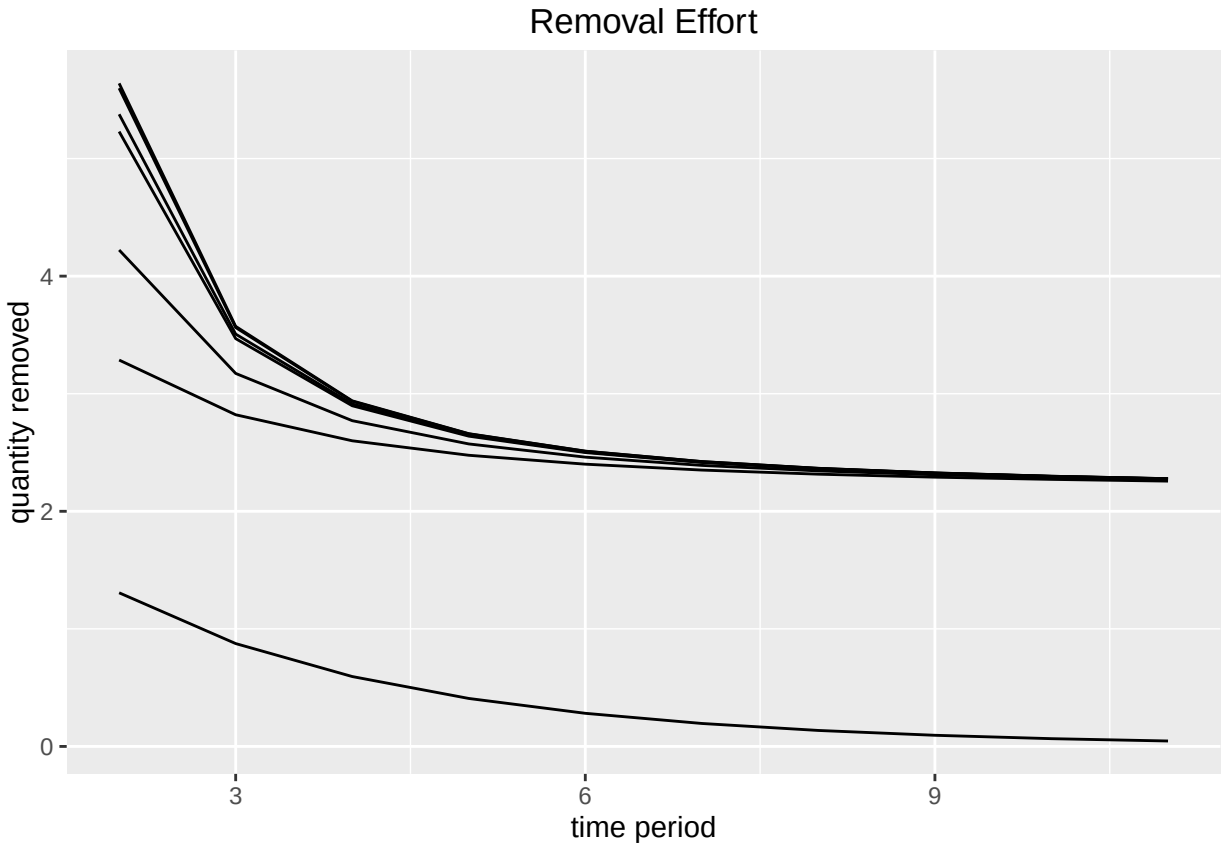
$$\sum_k (use_{k,n1,t}) < 1 \; \forall \; t \; if \; ord_t \neq 1$$

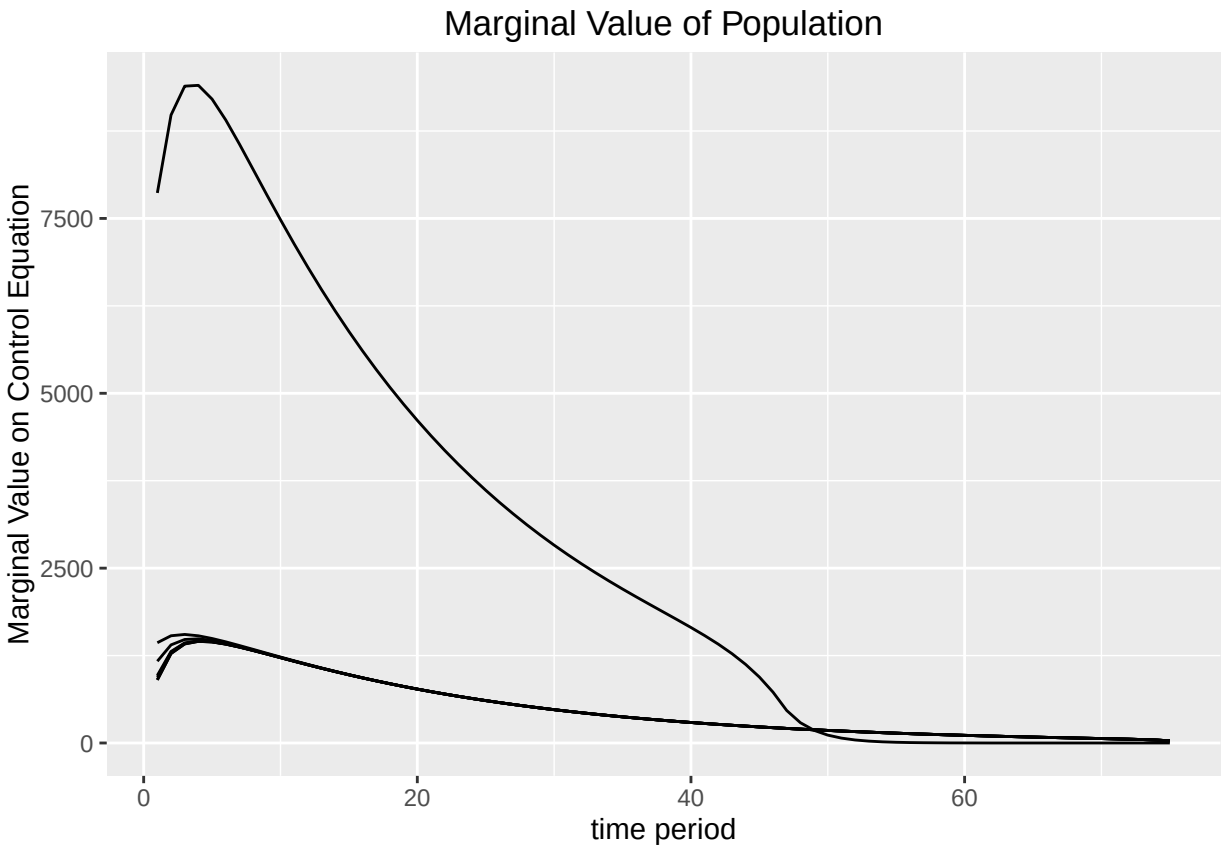
Finally, the objective function to be maximized is a total discounted benefits (z) function.

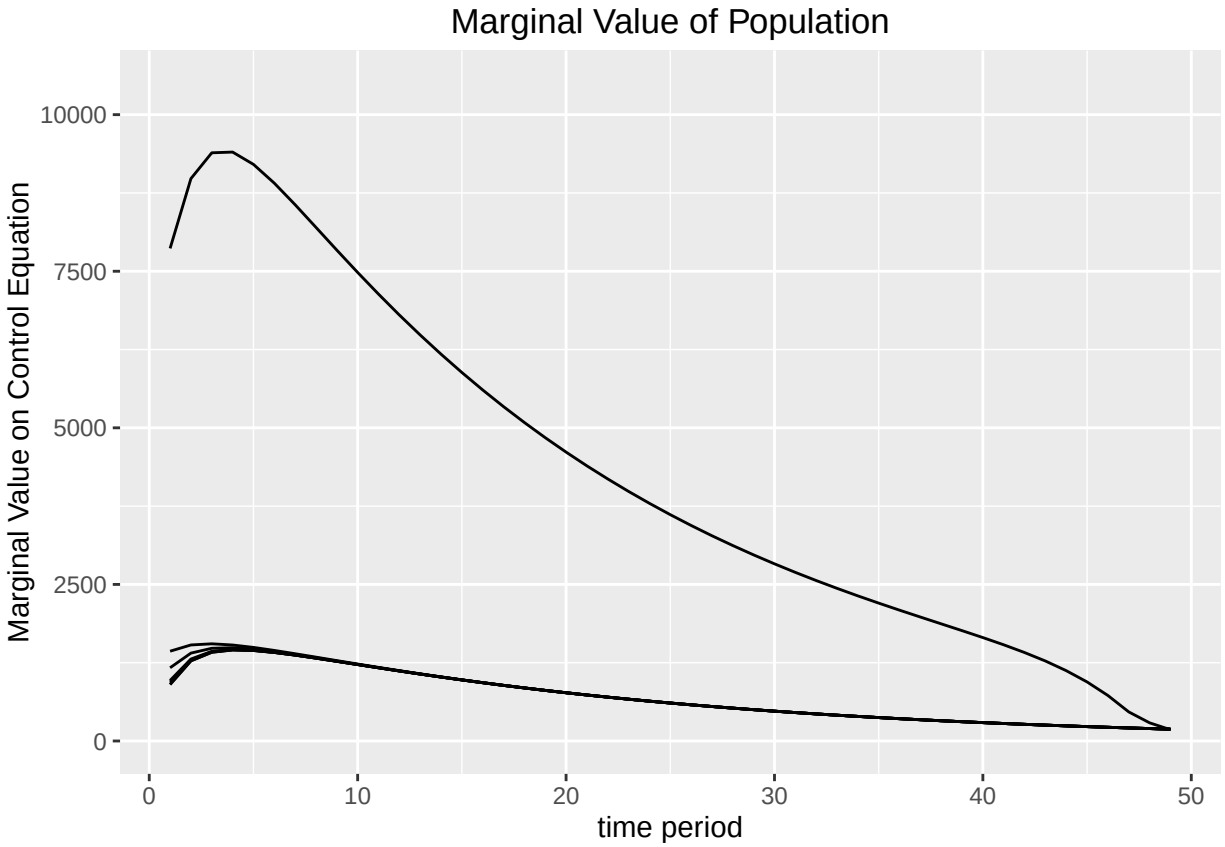
$$z = \sum_{k,n1,t} ((P_k \times Yield_k \times Quality_{n1} - VC_k) \times use_{k,n1,t} - (P_k \times Yield_k \times Quality_{n1} \times dam_{n1,t}) \times use_{k,n1,t} - (P\_removal_{n1,t} \times q_{n1,t})) \times$$

Results from the non-linear optimization show a typical response to a species that inflicts damage and is costly to remove. Removal effort is characterized by strong, but incomplete, removal effort to an approximate steady state population that is above zero and ending with a relaxation of removal effort. The figures below show that regardless of initial population, removal efforts and population converged to a near same path within approximately ten years.




Removal Effort
quantity removed
4
2
0
3
6
9
time period

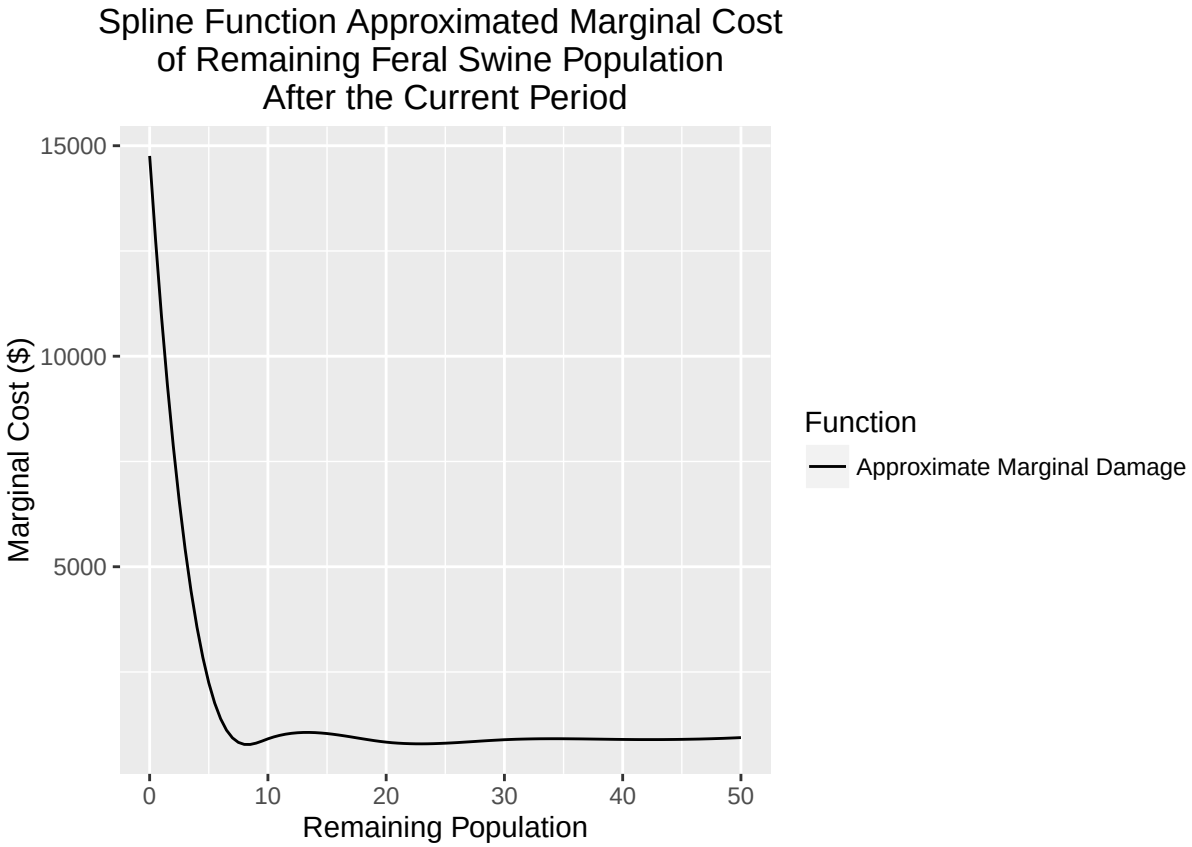

Marginal Value of Population
Marginal Value on Control Equation
7500
5000
2500
0
0
20
40
60
time period



The point of this non-linear model was to approximate the cost of leaving an animal on the landscape. To that end we need to do some calculations. First, we need to think of the decision-maker in the agent-based model. In each time period that decision maker decides what to grow and how many animals to remove. That is more analagous to our time period one in the dynamic model than time zero. Time zero is more of a spin up to allow for referencing previous periods. We are arguing that the agent-based decision-maker should be concerned about what leaving one more animal on the landscape will mean for future periods. Again, with the population `control` equation, the timing is reconciling the population after reproduction, after removal efforts, prior to damage, i.e. the same schedule as the agent-based model. The agent-based decision-maker is already considering damage in the present, so we need to remove damage in the present from the model's accounting of future costs imposed by an animal found in the `control_marginal`output of `gams`. The current period damage can be found from the derivative above and subtracted from the first period marginal obtained from the dynamic model to find the future periods damage.

$$y = Control\_Marginal(n)_{t=1} - damage'(n)$$

Corn and soybean damage marginal functions are very close at the tails and fairly close at the function's maximum. The spirit of the test is a simplified view of future damage, so these two functions will be averaged into a single. Using `R`'s spline interpolation functionality we are able do develop a function that will find the marginal value of a single animal left alive *after* the first period. This function is then added to the Agent-Based Simulation Framework.



2.4.3 Implementation in the Agent-Based Framework

Departing from rational expectations is challenging because there are infinitely many ways to model decision-making irrationally. Therefore, we will focus on a few that fit the narrative heard when talking to producers and extension professionals and common heuristic devices.

The agents make land use choices in a linear programming framework. The agent-based framework developed here gives substanially more granularity to spatial decisions and individual personalities, however gives up granularity in choice of actions. The same three crops are choices for the agents in the agent-based model, but the agent can only choose to not control, control lightly (around 25% reduction), or control heavily (around a 90% reduction). As such, we are able to pass a truely non-linear problem as linear. The key difference between the different implementations of OCPM is in construction of the objective function.

A technical detail that is necessary to understand the forthcoming equations is that `R` is particularly adept at processing functions across vectors and less adept at for-loops and similar structures commonly used in other languages. As such, the parameters and variables are vectors, not scalars.

The implementations in the agent-based framework are as follows. We define two dimensions of producer implementation. First, "removal price method", can be "marginal","minus",or "linear." The second is the "OCPM method." Code implementation often refers to this as "user cost" simply because of its succinct and clear (although not totally correct) meaning. OCPM method can take values of "myopic", "minus", or "linear."

A marginal removal price method implies that the decision maker evaluates the cost of removal at every increment. For example, the landowner chooses to remove the tenth animal, re-evaluates, and so on (`marginal*(incrRemovalPrice(pop_if_none,d))`) where `pop_if_none` is the current population and `d` is the proposed population. The minus removal price method implies that the producer is concerned about

the difference in prices between different remaining populations. The landowner evaluates the price of removal at the current and proposed population and then multiplies the difference by the number of animals removed that implies (`minus*((removalPrice(d)-removalPrice(pop_if_none))*(pop_if_none-d))`). The linear removal price method implies that the producer evaluates the removal cost function at proposed population levels and then multiplies that price by the implied number of animals removed (`linear*(removalPrice(d)*(pop_if_none-d))`).

A myopic approach to OCPM is simply setting the OCPM to zero (`ucmyopic*(d)`). A minus approach evaluates the OCPM function at current and proposed population levels and multiplies the difference by the proposed population change (`+ucminus*(minusUserCost(pop_if_none,d))`) and views this value as a benefit to removal. Finally, a linear approach applies the value as a cost imposed by a remaining population linearly (`-uclinear*(userCostf(d)*(pop_if_none-d))`) by evaluating the OCPM at the proposed population and then multiplying it by the difference in proposed and current populations.

Each agent is assigned removal price method and an OCPM method.

```
if(RemovalPriceMethod=="random"){
  a<-c("marginal","minus","linear")
  removalPrFnVec <- sample(a, init_hh, replace=TRUE)
}
if(RemovalPriceMethod=="linear"){
  removalPrFnVec <- rep_len("linear",length.out = init_hh)
}
if(RemovalPriceMethod=="marginal"){
  removalPrFnVec <- rep_len("marginal",length.out = init_hh)
}
if(RemovalPriceMethod=="minus"){
  removalPrFnVec <- rep_len("minus",length.out = init_hh)
}
if(UserCostMethod=="random"){
  a<-c("myopic","minus","linear")
  userCostFnVec <- sample(a, init_hh, replace=TRUE)
}
if(UserCostMethod=="myopic"){
  userCostFnVec <- rep_len("myopic",length.out = init_hh)
}
if(UserCostMethod=="minus"){
  userCostFnVec <- rep_len("minus",length.out = init_hh)
}
if(UserCostMethod=="linear"){
  userCostFnVec <- rep_len("linear",length.out = init_hh)
}
```

The damage function described above is implemented.

```
#' @rdname WildPigABM
#' @export
pctDamFunc <- function(x){0.9 + (-0.899)/(1 + (x/14.21)^5)}
```

A removal price function is defined as described above as well as a incremental removal price function used in the case of the marginal removal price.

```
#' @rdname WildPigABM
#' @export
removalPrice <- function(x){
  35.404 + (1464)/(1 + (x/3.299004)^1.843566)
}
```

```
#' @rdname WildPigABM
#' @export
incrRemovalPrice <- function(x,y,...){
sum(removalPrice(x:y))
}
```

The OCPM function is defined through the spline functionality of R described above. The minus method of approaching OCPM is done by first defining a function to difference the evaluated OCPM functions and multiply this difference by the difference in population levels.

```
#' @rdname WildPigABM
#' @export
minusUserCost <- function(pop_if_none,pop_if,...){
  x <- pop_if_none
  y <- pop_if
  z = x-y
  (userCostf(y)-userCostf(x))*z
}
```

These, smaller, defined functions are then used in a function to develop the gross margin for each activity. Mathematically, the gross margin is defined:

$$gm = p \times (y - y \times pctDamFunc(d)) - vc - linear \times (removalPrice(d) \times (pop\_if\_none - d)) - marginal \times (incrRemovalPrice(pop$$

In R the objective function generating function is defined.

```
#' @rdname WildPigABM
#' @export
estGM <- function(pop_if_none, pop_if_light, pop_if_heavy ,
                  dam_corn,dam_soy,dam_crp,
                  adjYieldCorn, adjYieldSoy,
                  P_corn, p_soy, p_crp,
                  varCost_corn,varCost_soy,varCost_crp,
                  dFnForm,Removal_Price,User_Cost,...){
  P_crp=P_CRP
  varCost_crp=verCost_CRP
  adjYieldCRP=10
# linDam <- ifelse(dFnForm=="linear",1,0) ## NOT IMPLEMENTED
# SigDam <- ifelse(dFnForm=="sigmoidal",1,0) ## NOT IMPLEMENTED
linear <- ifelse(Removal_Price=="linear",1,0)
marginal <- ifelse(Removal_Price=="marginal",1,0)
minus <- ifelse(Removal_Price=="minus",1,0)
ucminus <- ifelse(User_Cost=="minus",1,0)
ucmyopic <- 0
uclinear <- ifelse(User_Cost=="linear",1,0)


K = list("Corn","Soy","CRP")
KK = list("none","light","heavy")
for (k in 1:length(K)){
  for (kk in 1:length(KK)){
    p<-get(paste0("P_",tolower(K[[k]])))
    y<-get(paste0("adjYield",K[[k]]))
    d<-get(paste0("pop_if_",KK[[kk]]))
    vc<-get(paste0("varCost_",tolower(K[[k]])))
```

```
    gm <- p*(y-y*pctDamFunc(d))-vc-   #Net Income
      linear*(removalPrice(d)*(pop_if_none-d)) - # linear removal pricing
      marginal*(incrRemovalPrice(pop_if_none,d))- # removal price evaluated at each animal removed
      minus*((removalPrice(d)-removalPrice(pop_if_none))*(pop_if_none-d))+ #removal price is difference
      ucminus*(minusUserCost(pop_if_none,d))-
      ucmyopic*(d)-
      uclinear*(userCostf(d)*(pop_if_none-d))

    nam <- paste0("gm_",tolower(K[[k]]),"_",KK[[kk]])

    assign(nam,gm)
  }
}
return(c(gm_corn_none,
         gm_corn_light,
         gm_corn_heavy,
         gm_soy_none,
         gm_soy_light,
         gm_soy_heavy,
         gm_crp_none,
         gm_crp_light,
         gm_crp_heavy ))
}
```

This gross margin generation function is used within the context of the linear programming evaluation process. The implementation of this function begins by gathering the data it needs from `NetLogo` and the global environment of `R` and then generates a vector of gross margins.

```
#============================================================================
# All of the landowner decision making functions use the same framework.
# The only difference is the forumulation of the objective function.
# The revised dynamicish function realizes this and provides flexibility for
# mixed levels of rationality.

#' @rdname WildPigABM
#' @export
RevisedDynamicISH <- function(verboseR, t, now,today,
                        init_hh,
                        P_corn,P_soy,P_CRP,
                        varCost_corn, varCost_soy,verCost_CRP,
                        removalPrFnVec,userCostFnVec,typeDecision,
                        ...) {
    ## NL has all it needs
    RNetLogo::NLCommand('dynamic-prep-step1')
    Sys.sleep(2)
    landownersolution <- list()
    #start looping
    for (j in 1:init_hh) {
      hh <- j - 1
      if (verboseR == TRUE) {print(paste("hh=", hh))}
      patchset <- paste("patches with [owner = ",hh,"]")
      dat <- RNetLogo::NLGetPatches(c(
        "pxcor", "pycor",'pop_if_none',
        'pop_if_light',
```

```
  'pop_if_heavy',
  'adjYieldCorn',
  'adjYieldSoy',
  "pigTerritoryStrength"),
  patchset)
Sys.sleep(2)

pigTerritoryStrength <- as.vector(dat$pigTerritoryStrength)
pop_if_none <- as.vector(dat$pop_if_none)
pop_if_light <- as.vector(dat$pop_if_light)
pop_if_heavy <- as.vector(dat$pop_if_heavy)
adjYieldCorn <- as.vector(dat$adjYieldCorn)
adjYieldSoy <- as.vector(dat$adjYieldSoy)

#========== eval marginal cost at current, and cost at proposed population subtract
#fun.6
#usercost
#f
#set gross margins

obj <- estGM(pop_if_none=pop_if_none,
             pop_if_light=pop_if_light,
             pop_if_heavy=pop_if_heavy,
             dam_corn=dam_Corn,
             dam_soy=dam_soy,
             dam_crp=dam_CRP,
             adjYieldCorn=adjYieldCorn,
             adjYieldSoy=adjYieldSoy,
             P_corn=P_corn,
             P_soy=P_soy,
             P_crp=P_CRP,
             varCost_corn=varCost_corn,
             varCost_soy=varCost_soy,
             varCost_crp=verCost_CRP,
             dFnForm=NULL,
             Removal_Price=removalPrFnVec[j],
             User_Cost=userCostFnVec[j])
```

2.5 Landowner Income and Wealth

Landowners income and wealth are tracked over the course of the simulation. Income is calculated:

$$Income_j = \sum_i \sum_k GM_{i,k} \times use_{i,k}.$$

Note that the landowner only recieves income on what actually happened. Many calculations preceed this to reflect the state of feral swine. From the `masterSchedule`:

```
RNetLogo::NLCommand('go-wrap-up-myopic')
```

In spite of the command's name, it is used for all four decision types. This function calls a number of `NetLogo` functions.

```
to go-wrap-up-myopic
  ;myopic-income
```

```
  real-income
  calculateHHIncome
end
```

The first operation (`count_and_claim`) reassesses the presence of feral swine. The second operation calculates income on a per patch basis and the third totals the patch and other income by individual. In the second operation, damage and removal prices are reassessed and then a series of `if` statements determine what the land was used for and how much that was worth under the final conditions.

```
to real-income
    count_and_claim
  ask patches [
    let none  0
    let light (pigTerritoryStrength * 0.25)
    let heavy (pigTerritoryStrength * 0.90)
    set tempCornDam    0.9 + (-0.899) / (1 + (pigTerritoryStrength / 14.21) ^ 5)
    if tempCornDam > 0.999 [set tempCornDam 0.999]
    set tempSoyDam     0.9 + (-0.899) / (1 + (pigTerritoryStrength / 14.21) ^ 5)
    if tempSoyDam > 0.999 [set tempSoyDam 0.999]
    set removal_none   0
    set removal_light  (35.404 + (1464) / (1 + (pigTerritoryStrength / 3.299004) ^ 1.843566))  * light
    set removal_heavy  (35.404 + (1464) / (1 + (pigTerritoryStrength / 3.299004) ^ 1.843566))  * heavy

    set tkcount count tk

    if a_corn_none > 0  [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_none + tkcount *
      ]
    if a_corn_light > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_light + tkcount
      ]
    if a_corn_heavy > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_heavy + tkcount
      ]

    if a_soy_none > 0  [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_none + tkcount * bou
      ]
    if a_soy_light > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_light + tkcount * bo
      ]
    if a_soy_heavy > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_heavy + tkcount * bo
      ]

    if a_crp_none > 0  [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_none + tkcount * bounty
      ]
    if a_crp_light > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_light + tkcount * bounty
      ]
    if a_crp_heavy > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_heavy + tkcount * bounty
      ]
    output-show (sentence "parcel_NI:" parcel_NI
```

```
                              "pigTerritoryStrength" pigTerritoryStrength
                              "removal_none" removal_none
                              "removal_light" removal_light
                              "removal_heavy" removal_heavy
                              "tempCornDam" tempCornDam
                              "tempSoyDam" tempSoyDam
                              "a_corn_none" a_corn_none
                              "a_corn_light" a_corn_light
                              "a_corn_heavy" a_corn_heavy
                              "a_soy_none" a_soy_none
                              "a_soy_light" a_soy_light
                              "a_soy_heavy" a_soy_heavy
                              "a_crp_none" a_crp_none
                              "a_crp_light" a_crp_light
                              "a_crp_heavy" a_crp_heavy)

  ]
end
```

The third operation collects income on patches owned by each individual.

```
to calculateHHIncome
  ask households [
    let aa sum [parcel_NI] of patches with [owner = [hh_num] of myself]
    set HH_income aa
    set wealthi wealthi + HH_income
    show (sentence "HH_income" HH_income "Weath" wealthi)
    ]

end
```

This income information is sent to `R` for reporting.

```
wealthi <- cbind(wealthi,RNetLogo::NLGetAgentSet('wealthi', 'households', as.data.frame = TRUE))
...
names(wealthi) <- paste0("t",0:max_cycles)
  wealthi$id <- 1:45
  wealthi <- reshape::melt(wealthi,id.vars="id")
  wealthi$sim <- paste("sim",typeDecision,generic_removal_price,realistic,sep = "_")
```

2.6 Feral Swine Damage

If one were to simply find the net present value or discounted sum of future damages (see Moss 2013, ch. 5), the effect of progeny would be missed. If damage was inflicted linearly (such as in Zivin, Hueth, and Zilberman (2000)), we could use a formula for present value of a perpetuity, $PV = \frac{PMT}{r-g}$, where PV is present value, PMT is the annual cash flow inflicted by feral swine, r is the discount rate, and g is the annual population growth rate. However, while the exact functional form is unknown, a linear damage function does not follow the narrative that we hear from producers and does not make logical sense.

A non-linear damage function fits the known behavior of feral swine. First, the damage function must be able to handle populations greater than estimated carrying capacity. In Zivin, Hueth, and Zilberman (2000) the function is based on full damage of 90% being carried out by a population of 15 animals.

> Damage to rangeland, measured as a percentage loss of total revenue, is modeled as a linear function of pig population density (Barrett [16]):
>
> $$D(N) = cN$$

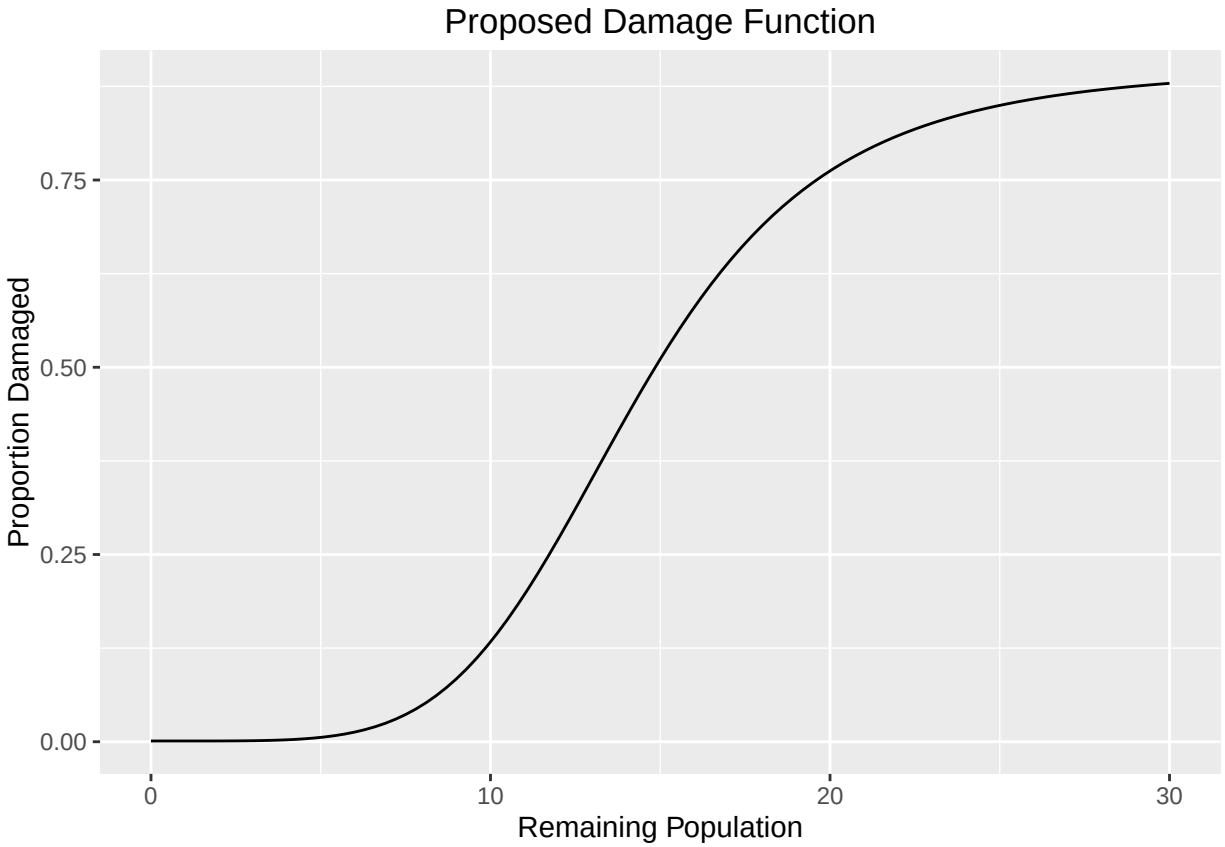
> Data and information on rangeland damage were collected from a number of studies (Barrett et al. [10], Sterner and Barrett [12], Barrett [17]). Based on this information, and through conversations with Barrett [16], c was set equal to 0.06. This implies that at full carrying capacity (15 pigs/km2), total rangeland damage is 90%. Because there is considerable uncertainty associated with this parameter, we vary this coefficient and examine its impact on steady-state results. (Zivin, Hueth, and Zilberman 2000, 197–98)

It is also known that based on a regression of change in damage by Hone (1995) that a non-linear damage function fits a change in damage measure. This approach should be parameterized for the United States, but certainly demonstrates the shape of what the damage function should be in a more complex representation. Hone (1995) found that there is a curvilinear relationship between the extent of rooting and freqency of rooting and feral swine density. The plots below demonstrate the curvature found when a measure of pig density (x) with one more measure for each line is regressed on the change in frequency of rooting.

Additionaly, a linear reduction in population is almost certainly not optimal from an optimal management perspective. This is what is found here, discussed later in the estimation portion of this discussion, and is typical in these types of problems. A reasonable followup question to this statement is, why not just use a planning technique that will produce optimal management?

We want to abstract away from an optimal management scenario to test differing levels of adherance to expert opinion. For instance, if one were to hire a consultant or ask for assistance from an extension agent about their feral swine problem, they would likely not be presented with a fully optimal management plan that accounts for movement, reproduction, resistance to control measures, the particular damage function of that farm, etc. The producer would be presented with a rule of thumb or may be presented something more complex and then act on it inappropriately. As discussed earlier, one person's mismanagement or refusal to cooperate will have consequences beyond their property borders. If we can condence the essence of an optimal management plan and then test different implementations of that management plan we can address this communication issue. As discussed earlier, there is a maximum level of damage and that when the population changes, damage changes non-linearly. These ideas can be fitted to a sigmoidal curve. A symmetric sigmoidal curve set to be very near zero at a population of zero and to max out at 90% damage at a full population of 30 head in the 2km^2 area.

$$y = 0.9 + (-0.899)/(1 + (x/14.21)^5)$$



2.6.0.1 Marginal Conditions

Both control and damage are a function of remaining population, rather than control effort. In addition, benefit is an avoided cost. They are two sides of the same coin, however, a subtle shift in interpretation of the marginal conditions is required. First, find the marginal damage for corn and soybeans.

$$\text{differentiate } damage_{corn} = (0.9 - 0.899/(1 + (x/14.21)^5)) \times 1350 \times 3.71 \text{ w.r.t. x}$$

$$damage'_{corn}(x) = (1.30439»10^{10} \times x^4)/(x^5 + 579389)^2$$

$$\text{differentiate } damage_{soy} = (0.9 - 0.899/(1 + (x/14.21)^5)) \times 450 \times 9.15 \text{ w.r.t. x}$$

$$damage'_{soy}(x) = (1.07234»10^{10} \times x^4)/(x^5 + 579389)^2$$

Then the cost of achieving a remaining population. There are more benefits to removal than the single season. We should include future benefits. It is unclear how long a decision-maker will plan, however, a perfect planner would look forward infinitely. The present value of a perpetuity is:

$$PV = PMT/r$$

where PV is present value, PMT is a payment, and r is a dicount rate.

$$damage'_{corn,perpetuity}(x) = (2.60878x10^{11}x^4)/(x^5 + 579389)^2$$

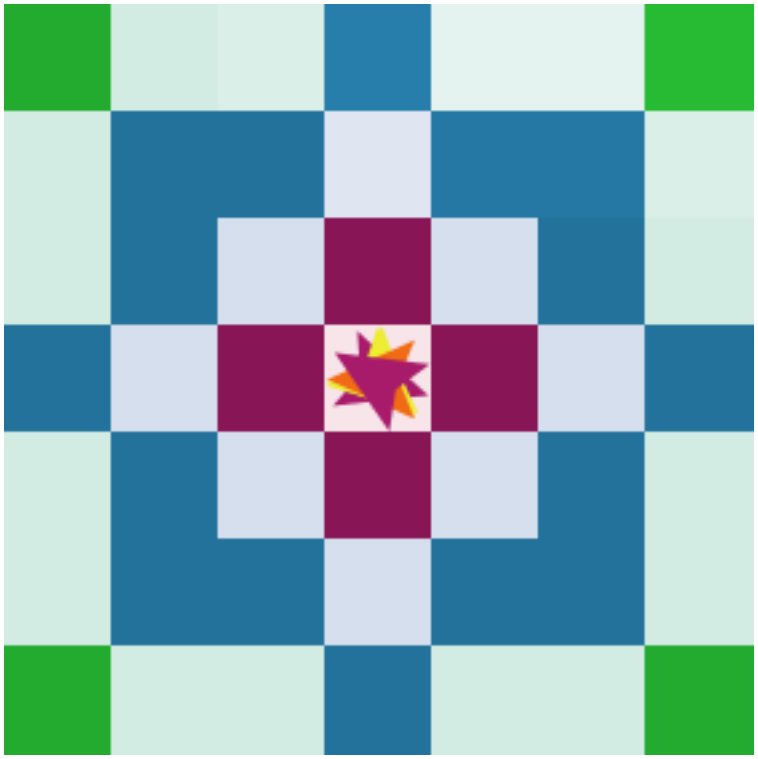


Figure 1: Example Feral Swine Presence

$$damage'_{soy,perpetuity}(x) = (2.14469x10^{11}x^4)/(x^5 + 579389)^2$$

Damage is calculated both before during the landowner planning phase (`dynamic-prep-step1`) and at the end of the period (`myopic-income`). Each time damage is a function of feral swine presence. Presence is a essentially a count of full and partial days of pig presence. Illustrated in the figure below, we see a center patch with several feral swine located on it. That patch should have a presence of $n \times 1$, the four red patches around that will have a presence of $n \times 1/4$, the eight light blue patches outside of that have a presence of $n \times 1/8$, the 16 patches outside of that have a presence of $n \times 1/16$, the lighter blue patches outside of those have a presence of $n \times 1/32$, and the green patches in the corners are untouched. If the range is increased, the impact continues to halve each distance outward.

Applying this idea (with different colors) to the entire map in the figure below we see how presence compounds between animals across space. Crop damage varies greatly by state and likely by specific areas within states. Anderson et al. (2016) found between 0.83% and 4.73% of corn and between 0.02% and 3.43% of soybeans are lost to feral swine damage. It is quite likely that damage is highly localized, meaning that where damage occurs it is intense, but there are vast areas without substantial damage as well. It is unlikely that damages would be this low if feral swine presence was at 100% of capacity.

Zivin, Hueth, and Zilberman (2000) implies that each pig creates 6% damage on an area of 1 km^2. On that scale, this may be a high estimate on a per-pig basis. For the purposes of this we will assume that 1 pig spending a whole year on a ten acre plot will cause 2.37% damage to corn and 1.71% damage to soybeans. This is a more conservative estimate when evaluated on the same scale as Zivin, Hueth, and Zilberman (2000).

```
## Warning: Removed 67 rows containing missing values (geom_path).
```

```
## Warning: Removed 16 rows containing missing values (geom_path).
```

Alternatively, damage could be modeled based on Hone (1995). This approach should be parameterized for

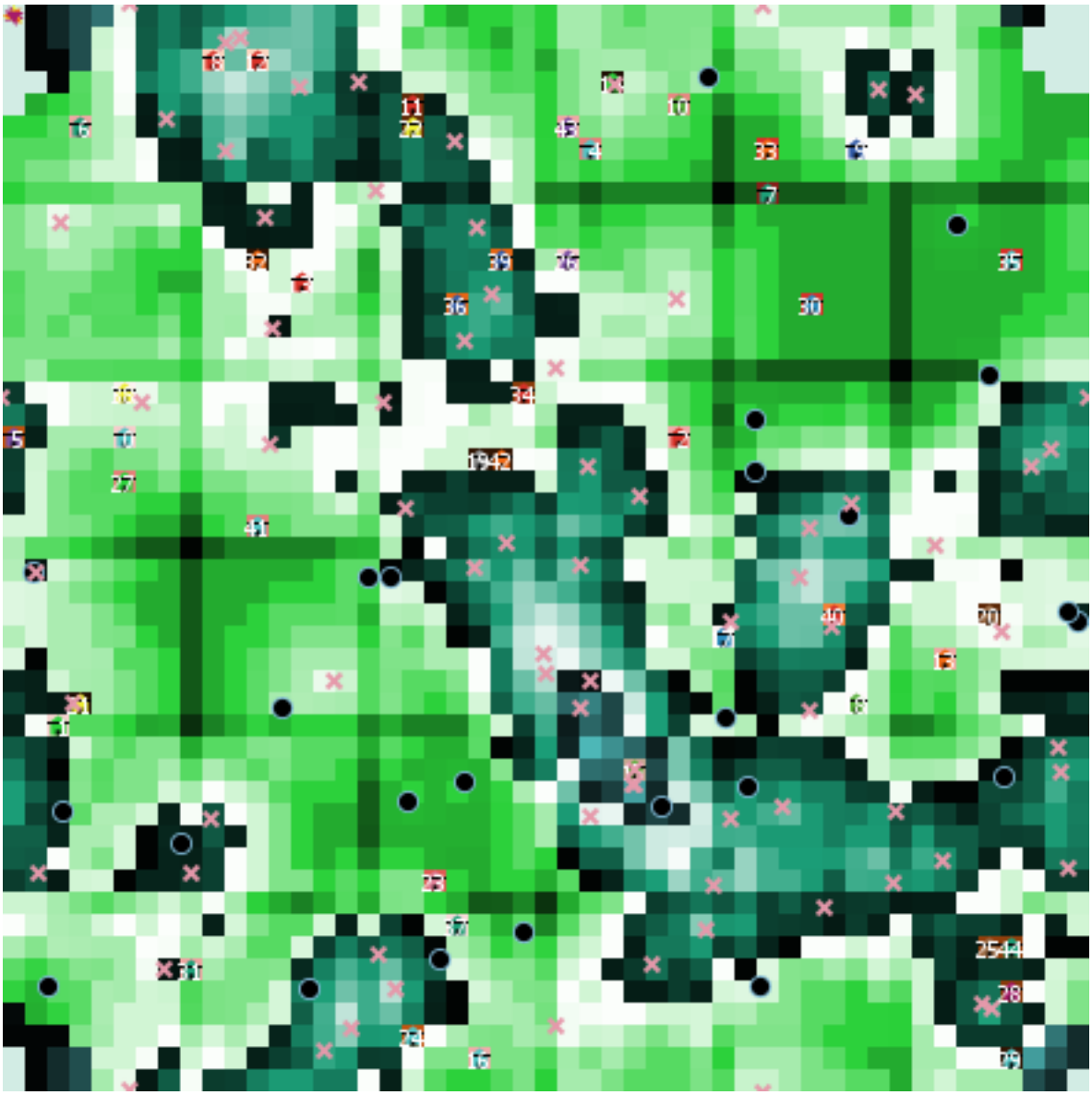


Figure 2: Example Feral Swine Presence - Entire Map

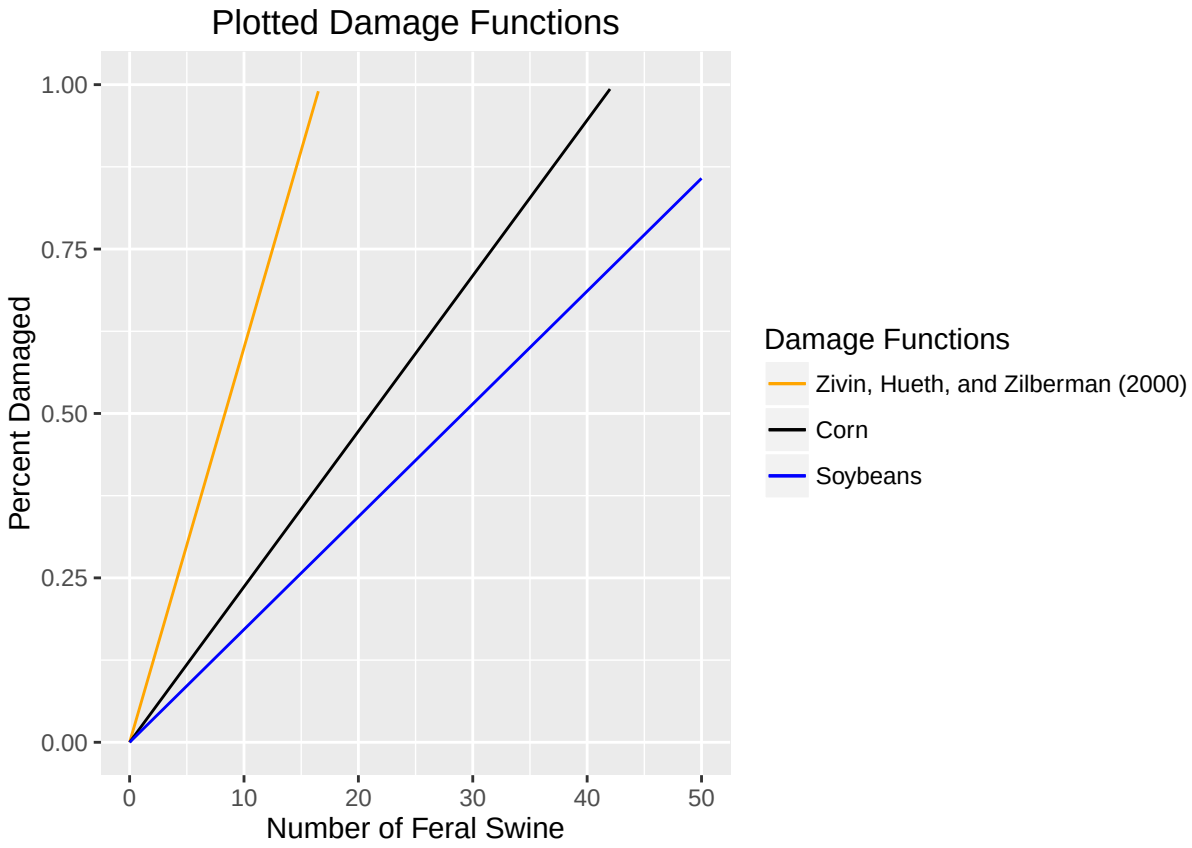
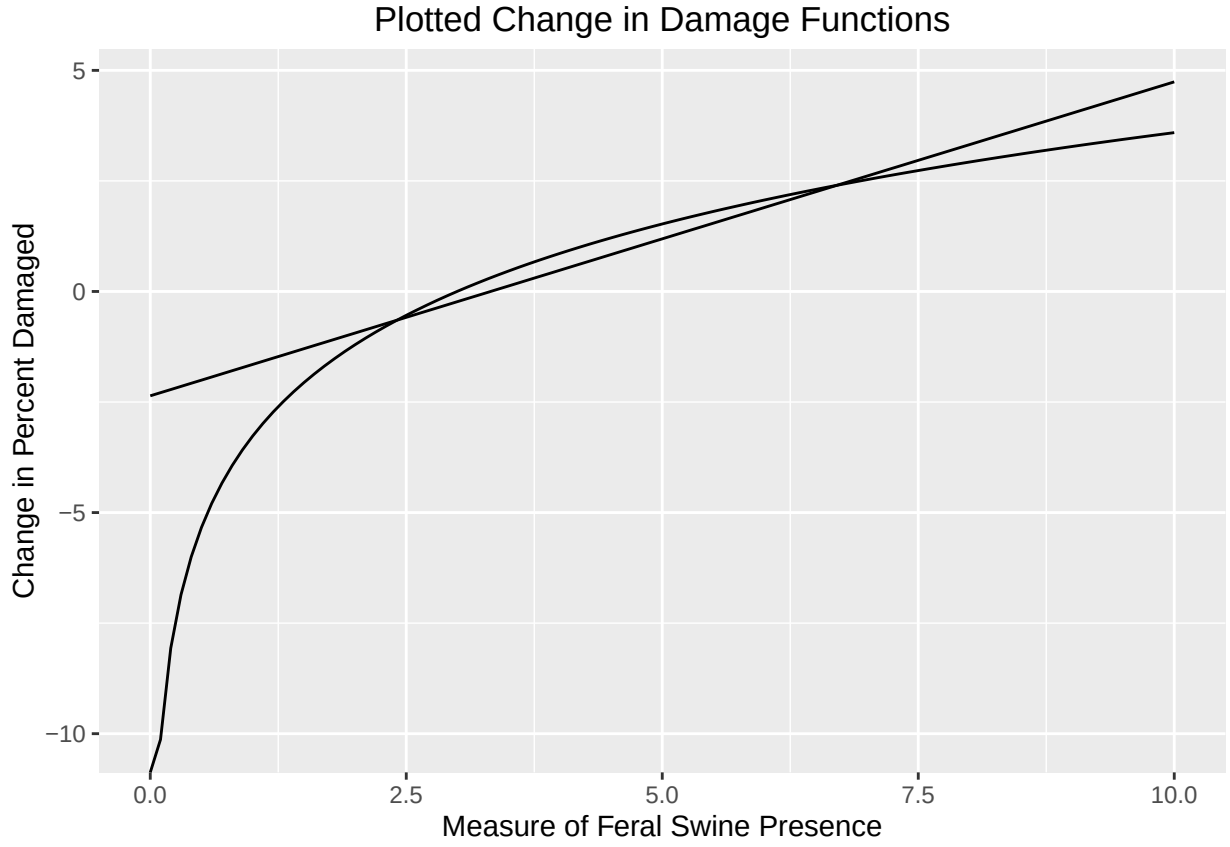


Figure 3: Crop Damage Functions Plotted.

the United States, but certainly demonstrates the shape of what the damage function should be in a more complex representation. Hone (1995) found that there is a curvilinear relationship between the extent of rooting and freqency of rooting and feral swine density. The plots below demonstrate the curvature found when a measure of pig density (x) with one more measure for each line is regressed on the change in frequency of rooting.



2.6.1 Implementation of Crop Damage

In the planning stages, crop damage is calculated by the `dynamic-prep-step1` function. The relevant excerpt below shows that a prediction of population after each kind of action is calculated and this value is used to calculate an expected yield contingent upon the action chosen.

```
...
    set no_damage_corn_GM P_corn * adjYieldCorn - varCost_corn
    set no_damage_soy_GM  P_soy  * adjYieldSoy - varCost_soy
    set pop_if_none     pigTerritoryStrength * (1 - none)
    set pop_if_light    pigTerritoryStrength * (1 - 0.25)
    set pop_if_heavy    pigTerritoryStrength * (1 - 0.90)
    set dam_corn_none   dam_Corn * pop_if_none
    set dam_corn_light  dam_Corn * pop_if_light
    set dam_corn_heavy  dam_Corn * pop_if_heavy
    set dam_soy_none    dam_Soy  * pop_if_none
    set dam_soy_light   dam_Soy  * pop_if_light
    set dam_soy_heavy   dam_Soy  * pop_if_heavy
    set yield_corn_none     adjYieldCorn * (1 - dam_corn_none)
    set yield_corn_light    adjYieldCorn * (1 - dam_corn_light)
    set yield_corn_heavy    adjYieldCorn * (1 - dam_corn_heavy)
```

```
    set yield_soy_none      adjYieldSoy  * (1 - dam_soy_none)
    set yield_soy_light     adjYieldSoy  * (1 - dam_soy_light)
    set yield_soy_heavy     adjYieldSoy  * (1 - dam_soy_heavy)
    ...
```

Similarly, at the income calculation stage (`myopic-income`), damage is calculated based on the actual presence at harvest.

```
to myopic-income
...
    set tempCornDam  dam_Corn * pigTerritoryStrength
    set tempSoyDam   dam_Soy  * pigTerritoryStrength
...
    if a_corn_none > 0  [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_none
      ]
    if a_corn_light > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_light
      ]
    if a_corn_heavy > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_heavy
      ]

    if a_soy_none > 0  [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_none
      ]
    if a_soy_light > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_light
      ]
    if a_soy_heavy > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_heavy
      ]

    if a_crp_none > 0  [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_none
      ]
    if a_crp_light > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_light
      ]
    if a_crp_heavy > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_heavy
      ]
```

2.7 Feral Swine Grouping

Grouping of feral swine is a known phenomena. The exact mechanism is unknown, but it is believed to be important in modeling their behavior. As such we have a simple first pass attempt. An agent is initialized for the purpose of being the focal point of a sounder.

```
sounders-own [
  sounder_id
  age
  scount
]
```

The actual creation is carried out during the `setup` process. The agent has an age, identification, and can store a count of sows who are members.

```
to setup-sounders
  create-sounders init_sounder
  [
  set age 0
  set sounder_id who
  set scount 0
  ]
end
```

Movement functions (documented in "Feral Swine Movement") and a function governing the splitting of large sounders are the only current use of this idea. `splitTheSounder` splits sounders that get larger than a user defined maximum size of sounder.

```
to splitTheSounder
  if any? sounders [
  let zz (maxSounderN + ALLOWABLEOVERCAP)
  ask sounders [
    set scount count sows with [sounder_id = [sounder_id] of myself]
    if scount > zz [
      let oldme who
      hatch-sounders 1 [
        set age 0
        set sounder_id who
        let sid sounder_id
        let op scount - zz
        ask n-of op sows with [sounder_id = oldme]
        [
          set sounder_id sid
          ]
      ]
    ]
  ]
  ]
end
```

2.8 Feral Swine Movement

Movement is a complicated process and is the strength of the `NetLogo` platform. The first step is to establish the attractiveness of each patch based on quality, last year's removal pressure, and relative population.

$$\text{Attractiveness}_i = f(\frac{1}{\text{Relative Pop. Density}_i}, Quality_i, \frac{1}{\text{Previous Year Removal Pressure}_i})$$

```
to prepareToMove
  ; land desirability is
        ;inversely related to population
        ;inversely related to pressure
        ;inversely related to other sounder's territory strength
        ;positively related to quality
     count_and_claim
     let maxperpatch (max_pigs / total_patches)
  ask patches[
```

```
    ; find patch populations, set as proportion to full capacity

    let mapPopulation boarCount + sowCount
    let x (mapPopulation / maxPigsPerPatch)
    ; recall quality
    let z quality
    ; establish attractiveness pre-land claim (additive)
    let tempattr ((1 / (x + 2.22e-16)) + (z))
    ; recall psi t-1
    let y psi_lastPeriod
    ifelse y < 0.01 ; inverse epsilon small numbers are very big and that is
    ;nonsense anyway
    [set PatchAttractivenessPriortoLandClaim tempattr]
    [set PatchAttractivenessPriortoLandClaim (tempattr + (1 / (y + 2.22e-16)))]
  ]
end
```

Movement of sounders is the most complicated movement. Each time through the sounders claim territory and then move according to their movement function. Each sounder is intended to move away from other sounders. Later generations will build on the loop structure commented out as timesteps are narrowed.

```
to move-sounders
  if any? sounders [
    let nb num_bases_per_year
    ;foreach n-values 5 [ ? + 1 ][
  ;set over-all attractiveness of patches
    prepareToMove
  ;set current territory
  ;evaluate attractiveness for each sounder ;sounders move
    ask sounders [
      let tempPig sounder_id
      ask patches in-radius srange [
        let nm (int sounderTerritory - int tempPig) ^ 2
        ifelse nm > 0 [set notmine 1][set notmine 0]
        set localsounderTerritoryStrength sounderTerritoryStrength * notmine
        set localAttractiveness (PatchAttractivenessPriortoLandClaim - localsounderTerritoryStrength)
        if localAttractiveness < 0.001 [set localAttractiveness 0]
        ]
      let ideal-site max-one-of patches in-radius srange [localAttractiveness]
      move-to ideal-site
      if any? sows with [sounder_id = [sounder_id] of myself]
      [
        ask sows with [sounder_id = [sounder_id] of myself][
          move-to sounder tempPig
          ]
        ]
      ]
    ;rinse and repeat
    ;]
  ]
end
```

Boars move for similar reasons. By not having to move in concert, the boar movement function is much simpler.

```
to moveBoars
```

```
  if any? boars [
    ask boars [
      let b brange
      ask patches in-radius brange [
        set localsounderTerritoryStrength sounderTerritoryStrength * notmine
        set localAttractiveness (PatchAttractivenessPriortoLandClaim - localsounderTerritoryStrength)
        if localAttractiveness < 0.001 [set localAttractiveness 0]
      ]
      let ideal-site max-one-of patches in-radius b [localAttractiveness]
      move-to ideal-site
    ]
  ]
end
```

2.9 Feral Swine Removal Pricing

Depending on the simulation, it may make sense to price removal in different ways.
The removal of feral swine is increasingly difficult as the population approaches zero. The most extreme statement of this is:

$$\lim_{n \to 0} w_{control}(n) \ \to L.$$

There have been successful eradication campaigns, so the price of removal to zero is not infinite. Saunders and Bryant (1988) proposed a removal cost function based on flight time used in an exercise in Australia,

> p = 1 - 1.177exp(0.001375t). With one helicopter we can assume an average of 400 min flying time per day. To remove 95% of the population would have therefore taken the exercise to 7 days (2297 min). Similarly, a 99% reduction would take 9 days (3468 min). (Saunders and Bryant 1988, 78)

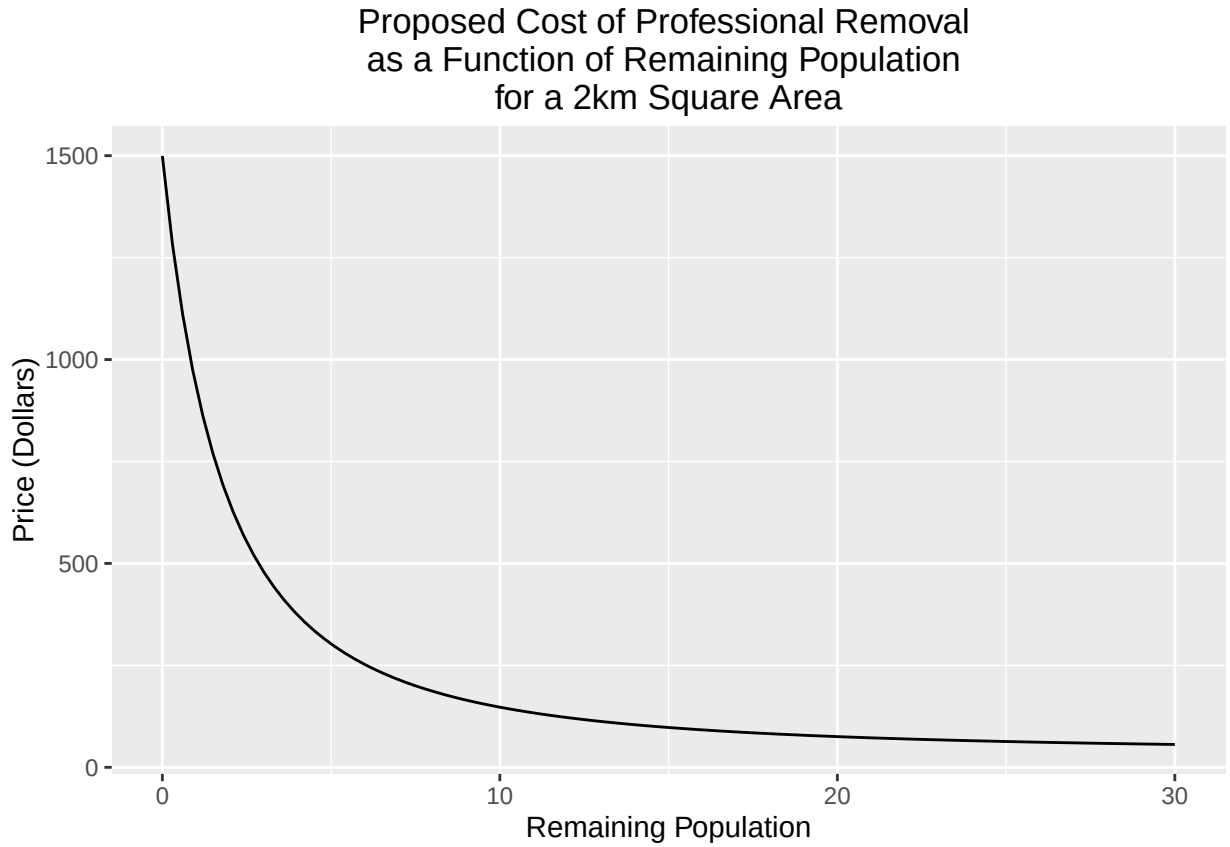
Wildlife managers understand the increasing cost of removal, but they also understand the need to communicate average costs. That is, we know that we are currently removing something close to 20-30% of the feral swine in Texas (Carson 2013) and mixed removal actions in Texas cost approximately $65/head (Bodenchuk 2014). If we solve the equation proposed by Saunders and Bryant (1988) for t, we find:

$$t = 8000/11 \times ln(-100/177 \times (p-1)) \ for \ p < 1.$$

However, what we really need is a pricing mechanism for professional removal that reflects the increasing difficulty of removing as the population reaches zero. A function like the one below reflects the price we currently face with abundant populations and approaches the theoretical price curve proposed by Saunders and Bryant (1988).

$$w_{control} = 35.404 + (1464)/(1 + (x)/3.299004)^{1.843566})$$

Plotted, this function retains the shape of Saunders and Bryant (1988) but instead works off of the expected remaining population and calibrates to approximate twenty percent removal at approximately $65/head and removal of the last animal at approximately $1500.



Alternatively, it could be modeled that each type of removal has a different cost or revenue associated with it. The sections below detail how that could be carried out.

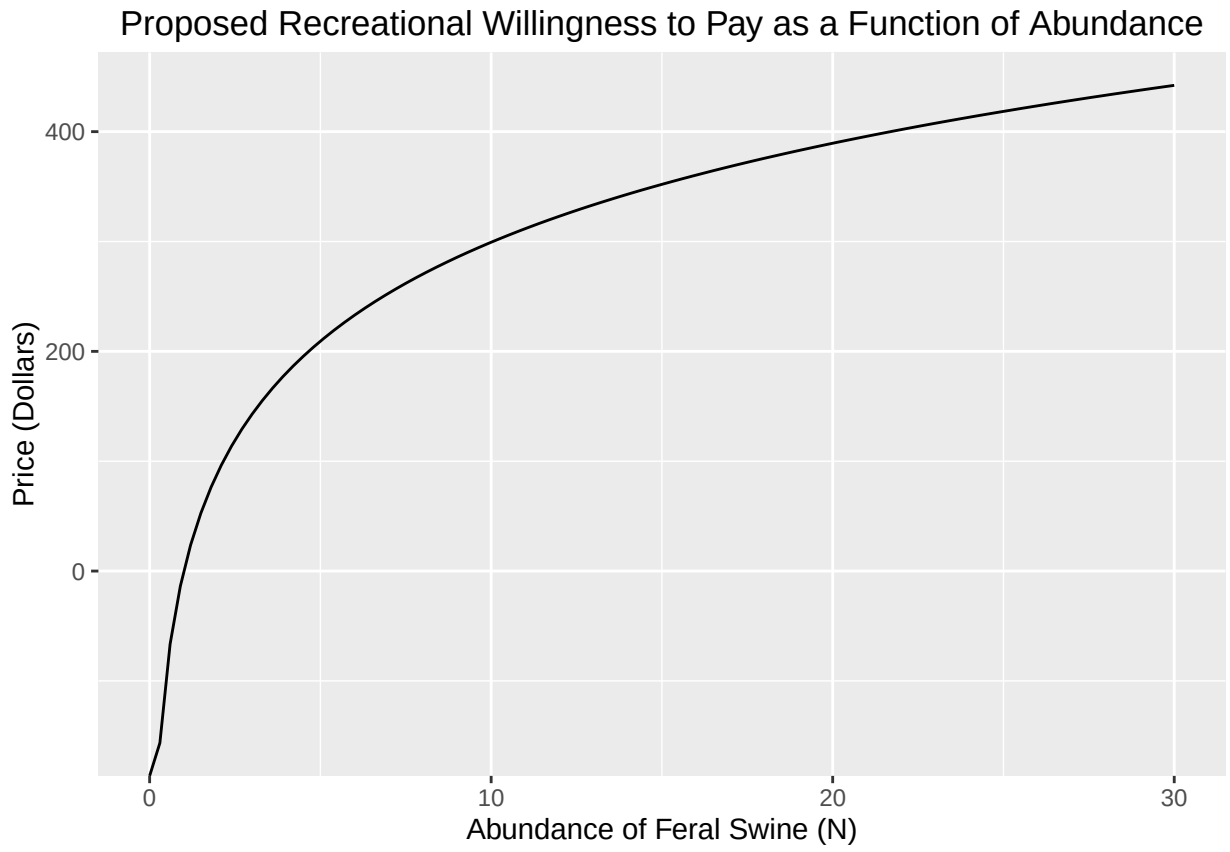
2.9.1 Price consumers are willing to pay to hunt.

We do not know much about pricing of commercial feral swine hunts. We can gather from Hainds (2011) that about $300 is the going rate for a certain, ground based, kill. It seems reasonable to assume that at some threshold of density a kill is practically guaranteed on average. Lets call that threshold `minPigsFee`. For the moment, lets say that there is nothing more valuable than the guaranteed kill, so this $300 is a minimum of a day of hunting and a minimum of a single kill. Obviously, there are a number of factors such as operating as a bed and breakfast that would add value and possibly add to willingness to pay. Those factors are beyond the immediate scope of the study.

A price function such as:

$$p = 260 * ln(n^{1/2})$$

where p is the price of a hunt and n is the population on a parcel (patch) of property. This function reaches the price of $300 per hunt when the property has at least ten head (per ten acres), equal to `minPigsFee`.



2.9.2 Cost of removal effort by paid professionals.

2.9.2.1 Professional Removal as a Function of Remaining Population

We can use $65 per head (Bodenchuk 2014) as a starting place for the cost of removing feral swine. However, we know that it is progressively more expensive to remove swine as fewer remain on a parcel. Saunders and Bryant (1988) document the cost of an eradication effort in Australia. No such study exists for the US to the knowledge of the authors. This is likely not a correctly calibrated parameterization for the United States as the study area was a very large open area. The environment in some places in the US might be quite similar, however, a smaller wooded area would have different results. In any case the curvature describes what is anecdotely known about the cost of removing a large proporion of the animals in a given area; Saunders and Bryant (1988) found:

> p = 1 - 1.177exp(0.001375t). With one helicopter we can assume an average of 400 min flying time per day. To remove 95% of the population would have therefore taken the exercise to 7 days (2297 min). Similarly, a 99% reduction would take 9 days (3468 min). (Saunders and Bryant 1988, 78)

We do not necessarily know how many animals are desired to be removed *a priori*, However, ecologists are able to estimate what proportion of full population is present. If we work from that standpoint, and say that we know full capacity is `max_pigs/total_patches` and the landowner is able to perceive that there is a presence of `sounderTerritoryStrength+boarTerritoryStrength` on a given patch. If we solve Saunders and Bryant's equation for t we get:

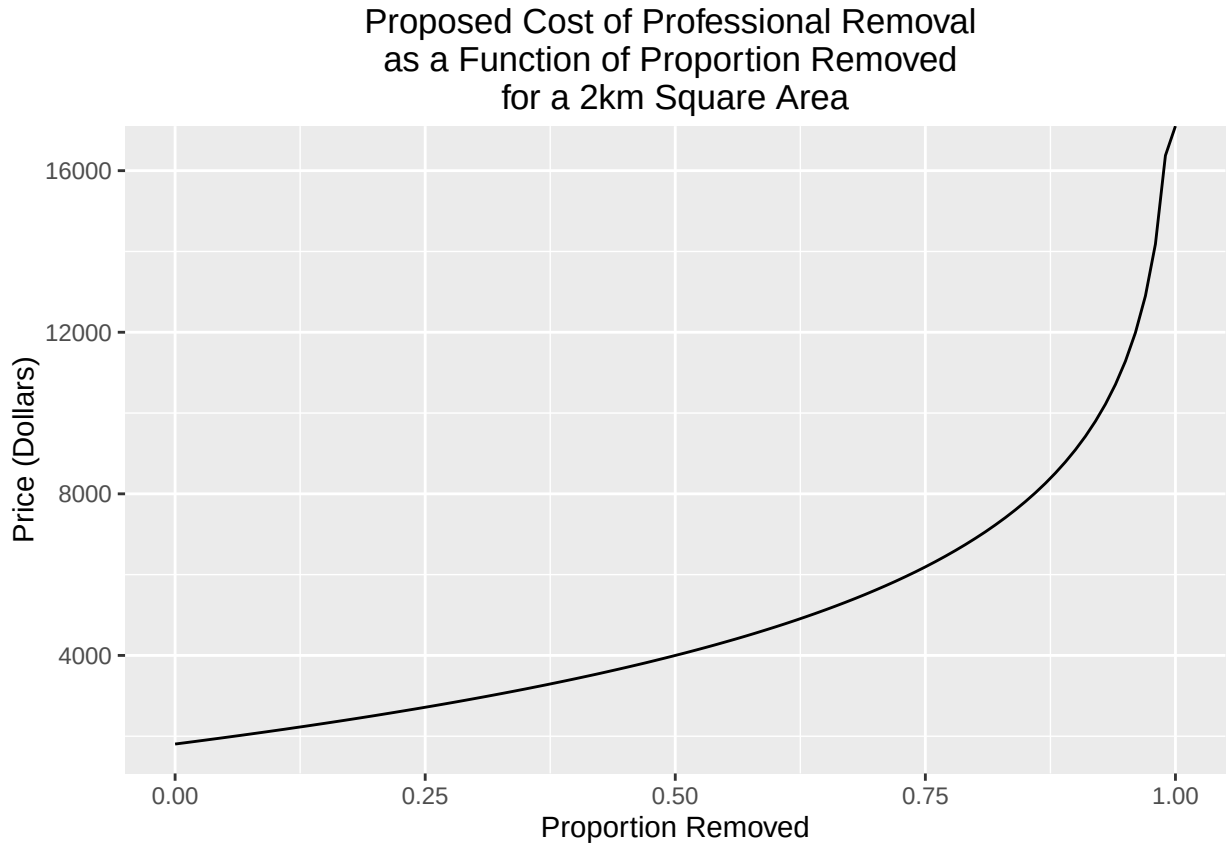
$$t = 8000/11 \times ln(-100/177 \times (p-1)) \; for \; p < 1.$$

If we set $(-1 * `sounderTerritoryStrength` + `boarTerritoryStrength`) = p$ we could then multiply the time required by the price of helicopter rental until a more exact number can be found. This would bring mean

the cost function would be approximately:

$$varCost_{Paid} = (8000/11 \times ln(-100/177 \times (p-1))) * pMinuteFlight) \ for \ p < 1.$$

We can implement this in `R` and `NetLogo` using the cost of rental for training hours of $261/hr. This works out to $4.35/minute.(http://www.flyhaa.com/helicopter/costs/details/)

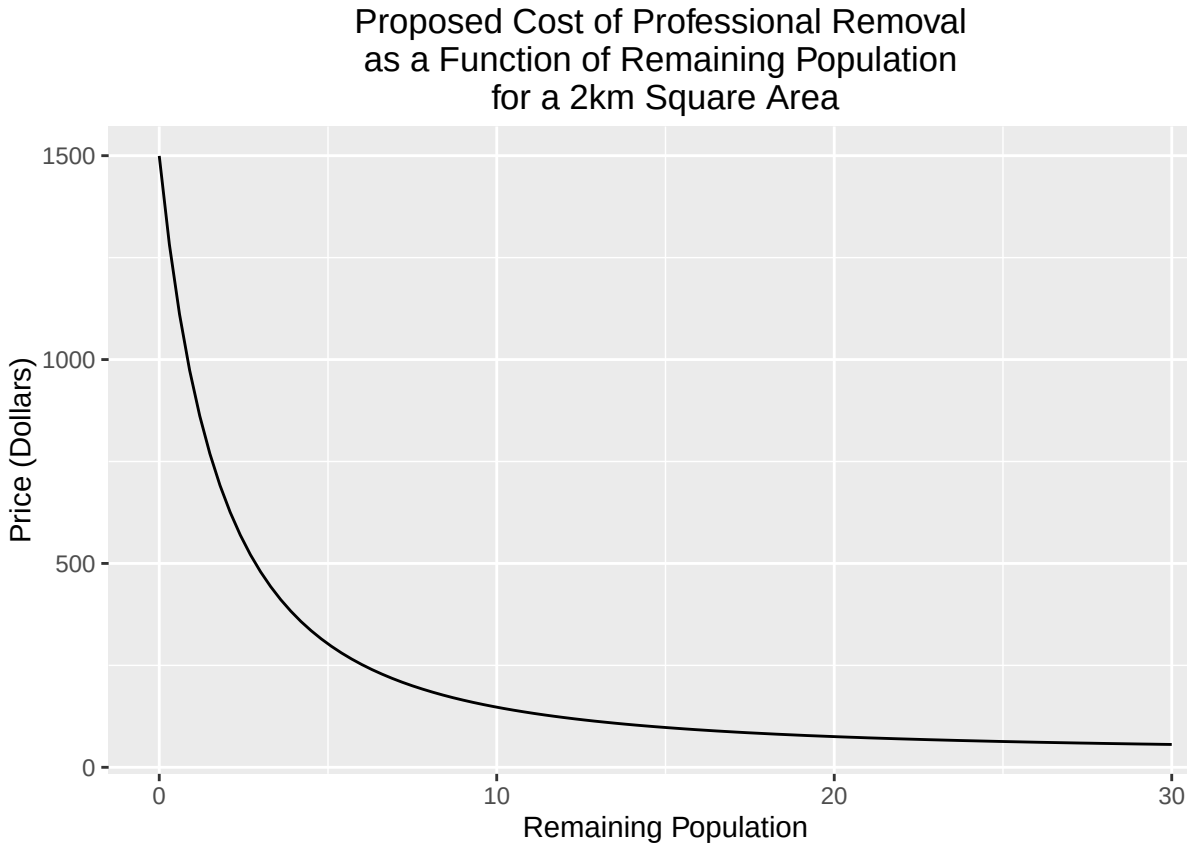


2.9.2.2 Professional Removal as a Function of Remaining Population

However, what we really need is a pricing mechanism for professional removal that reflects the increasing difficulty of removing as the population reaches zero. A function like the one below reflects the price we currently face with abundant populations and approaches the theoretical price curve proposed by Saunders and Bryant (1988).

$$p = 35.404 + (1464)/(1 + (x)/3.299004)^{1.843566})$$

Plotted, this function retains the shape of Saunders and Bryant (1988) but instead works off of the expected re-



maining population.

Parameterizing any of these functions based on some form of data would be ideal. This will have to suffice until that can be done.

2.10 Feral Swine Removal

Survival of an individual is a function of harvest effort among other factors (Hanson et al. 2009). Sex, season, weight, and age were other factors used to predict the survival of individuals (Hanson et al. 2009). Parameters for Hanson et.al. (2009)'s models were not published so the precise relationship between those variables and survival is unknown. We also know (at least anecdotely) that swine respond to removal effort. We will model the probability of an individual feral swine removal as

$$pr(Removal_i) = f(\text{current effort}, \text{previous effort experienced by individual}, \text{chance}).$$

2.10.1 Landowner Removal

Landowners choose to remove feral swine when

$$benefit \geq cost$$

purely in pecuniary terms. Non-pecuniary benefits could be easily added through a price mechanism. Removal effort is represented through the variable `Psi` or Ψ. First, the wild pigs are counted on home territory. Landowners are aware of the cumulative presence of wild pigs on each patch they control through `pigTerritoryStrength` and choose their actions based on this knowledge. However, removal is based upon actual presence on a patch. The implication is that the landowner must choose to remove on the wild pig's home patch in order to actually harvest an animal. Scheduling is carried out in the `masterSchedule()` function. Within this function, count is updated, `Psi` is calculated, and wild pigs are removed.

```
### Update FS CageynessFS Breed and Move ------------------------------------------------------------
    RNetLogo::NLCommand('go-pigops')
    if (gui == TRUE){
      readline(prompt="Press [enter] to continue")
    }
    RNetLogo::NLCommand('go-myopic-removal')
    if (gui == TRUE){
      readline(prompt="Press [enter] to continue")
    }
```

Counting is documented in the `processing-functions` documentation. `Psi` is calculated in the `NetLogo` function `go-pigops`.

```
to go-pigops
  scarePigs
  breedR
  countFSTerritoryStrength
  move-sounders
  moveBoars
  splitTheSounder
end
```

The `scarePigs` function operationalizes the idea that exposure to removal pressure will make animals more responsive to future control efforts and thus less likely to be removed. This cageyness has a decay function, here each year only if cdecay equals 4, 25% of last year's cageyness will carry into this year and 50% of last year's experiences will be incorporated into this year's cageyness.

$$\_cagey = \frac{1}{cdecay} \times \_cagey + \frac{1}{2} \times psi\_lastPeriod_i$$

```
to scarePigs
   if any? sows [
     ask sows [
       set scagey 1 / cdecay * scagey + [1 / 2 * psi_lastPeriod] of patch-here
     ]
   ]
    if any? boars [
     ask boars [
       set bcagey 1 / cdecay * bcagey + [1 / 2 * psi_lastPeriod] of patch-here
     ]
    ]
end
```

Breeding (`breedR`) is covered in the Feral Swine Reproduction chapter and the remaining functions are covered in the Movement chapter. The rest of the removal functions are carried out in the `go-myopic-removal` function. The 'myopic' in the name of the function has no bearing on its actual function. The function was created for a specific operation, and has proven useful in many later cases.

```
to go-myopic-removal
  myopic-calcPsi
  goRemoveBoars
  goRemoveSounders
  countDead
end
```

The count is taken in the `goPigOps` function and that count is used for the removal functions as well. `Psi` is then calculated for each patch based on landowner decisions. Effort is measured by `Psi` and is simply chosen by the decision-maker for each patch in each time period. The implication is that effort is constrained to

fall roughly between zero and one. The goal of each of these operations is to build a probability of removal for each animal.

$$\Psi = [0, ..., 0.999]$$

The second to last line is a defensive programming technique to prevent an inadvertant setting of `Psi` above its defined range.

```
to myopic-calcPsi
  ask patches [
    if a_corn_none > 0  [
      set psi a_corn_none * 0
      ]
    if a_corn_light > 0 [
      set psi a_corn_light * 0.25
      ]
    if a_corn_heavy > 0 [
      set psi a_corn_heavy * 0.90
      ]

    if a_soy_none > 0  [
      set psi a_soy_none * 0
      ]
    if a_soy_light > 0 [
      set psi a_soy_light * 0.25
      ]
    if a_soy_heavy > 0 [
      set psi a_soy_heavy * 0.90
      ]

    if a_crp_none > 0  [
      set psi a_crp_none * 0
      ]
    if a_crp_light > 0 [
      set psi a_crp_light * 0.25
      ]
    if a_crp_heavy > 0 [
      set psi a_crp_heavy * 0.90
      ]
   if psi > .999 [set psi .999]
   print psi
  ]
end
```

Probability removed is then set for each animal on the map. The appropriate cagey variable modifies the pressure experienced by each individual in the current period. As stated above,

$$Pr(Removal)_{pig} = f(\text{current effort}, \text{previous effort experienced by individual}, \text{chance}).$$

The element of previous effort experienced by individual a pig, captured by `bcagey` and `scagey`, is added to an individual probability for each animal `PrRemoved` in the function below.

```
to setPrRemoved
  if any? boars [
    ask boars [
      set mypsi [psi] of patch-here
      set prRemoved (mypsi - mypsi * bcagey)
```

```
    ]
  ]
  if any? sows [
    ask sows [
      set mypsi [psi] of patch-here
      set prRemoved (mypsi - mypsi * scagey)
    ]
  ]
end
```

Once each animal has its own probability of removal, each boar and sow is presented with an opportunity to be removed. Once removed, a hidden agent is created for tracking purposes. Probability of removal is not sufficient for removal. If the threshold is met, the animal is removed. In this function, each animal is presented individually with a probability of removal. The *chance* component of the probability of removal is implemented at this point. A random draw from a uniform distribution $U(0, 1)$ is compared to `prRemoved` and if it is less than the `prRemoved` the animal is removed. For example if $prRemoved = 0.75$ there would be a nearly 75% chance that a random draw, $x \in [0, 1] < 0.75$.

```
to goRemoveBoars
  if any? boars[
    ask boars [
      let rn random-float 1.0
      show (sentence "prRemoved" prRemoved "bcagey" bcagey "rn" rn)
      if rn < prRemoved [
      hatch-killed 1 [ht]
      hatch-tk 1 [ht]
      die
      ]
      ]
    ]
end

to goRemoveSounders
  if any? sows[
    ask sows[
      let rn random-float 1.0
      show (sentence "prRemoved" prRemoved "scagey" scagey "rn" rn)
      if rn < prRemoved [
        hatch-killed 1 [ht]
        hatch-tk 1 [ht]
        die
        ]
      ]
    ]
end
```

2.11 Feral Swine Reproduction

Modeling on an annual timestep we do not necessarily need the the actual birthrate and natural deathrate. We should be able to combine those into a single variable: `effectiveBirthRate`. If it is known that we have to kill between 60% and 80% per year to keep the stable what does this mean for an effective reproductive rate? Population in period t is the previous period's population plus the previous period's population times an effective birthrate minus the previous year's population times a removal rate.

$$pop_t = pop_{t-1} + pop_{t-1} \times effectiveBirthRate - pop_{t-1} \times removalRate$$

if:

$$pop_t = pop_{t-1}$$

then

$$pop_{t-1} = pop_{t-1} + pop_{t-1} \times effectiveBirthRate - pop_{t-1} \times removalRate$$

solve for $effectiveBirthRate$

$$1 = 1 + effectiveBirthRate - removalRate$$

which means that if

$$pop_t = pop_{t-1} \tag{1}$$

$$\text{then} \tag{2}$$

$$removalRate = effectiveBirthRate \tag{3}$$

So by this logic we can expect each pig (male or female) to produce 0.6 - 0.8 offspring per year. Since only females have offspring, lets solve again to account for that.

$$0 = pop_{t-1} \times ProportionFemale \times effectiveBirthRate - pop_{t-1} \times removalRate \tag{4}$$

$$\frac{removalRate}{ProportionFemale} = effectiveBirthRate \tag{5}$$

So if 75% of feral swine are female and it requires 60% removal to maintain a stable population then we can expect

$$60/75 = 0.8$$

offspring per year. This rate appears to be in line with previous literature (Bieber and Ruf 2005; Geisser and Reyer 2005).

We can represent this with a default litter.

```
defaultLitter = msm::rtnorm(1,mean=0.8, sd=1*sdMult,lower=0,upper=1)
```

Further, swine on better patches will have higher fecundity than those on worse patches, however the magnitude of the effect depends on the landscape in question [Bieber and Ruf (2005);geisser2005]. Removal pressure will also impact the ability of sows to produce viable adults, although the magnatude of this effect is less well known.

```
#' @rdname WildPigABM
#' @export
breedR <- function(verboseR,psi_lastPeriod,...){
  NL_pushToMap("psi_lastPeriod",psi_lastPeriod[,,1],nl.obj = NULL)
  RNetLogo::NLCommand('breedR')
  if (verboseR == TRUE) {print("pigs reproduced")}
}
```

The `breedR` function in R pushes last year's pressure to `NetLogo` and calls the `NetLogo breedR` function. Sows pass on membership to their sounder to their female offspring. The number of expected offspring will go to zero as the population nears capacity.

$$al = (1 - pigCountSum/(max\_pigs)) * defaultLitter \tag{6}$$

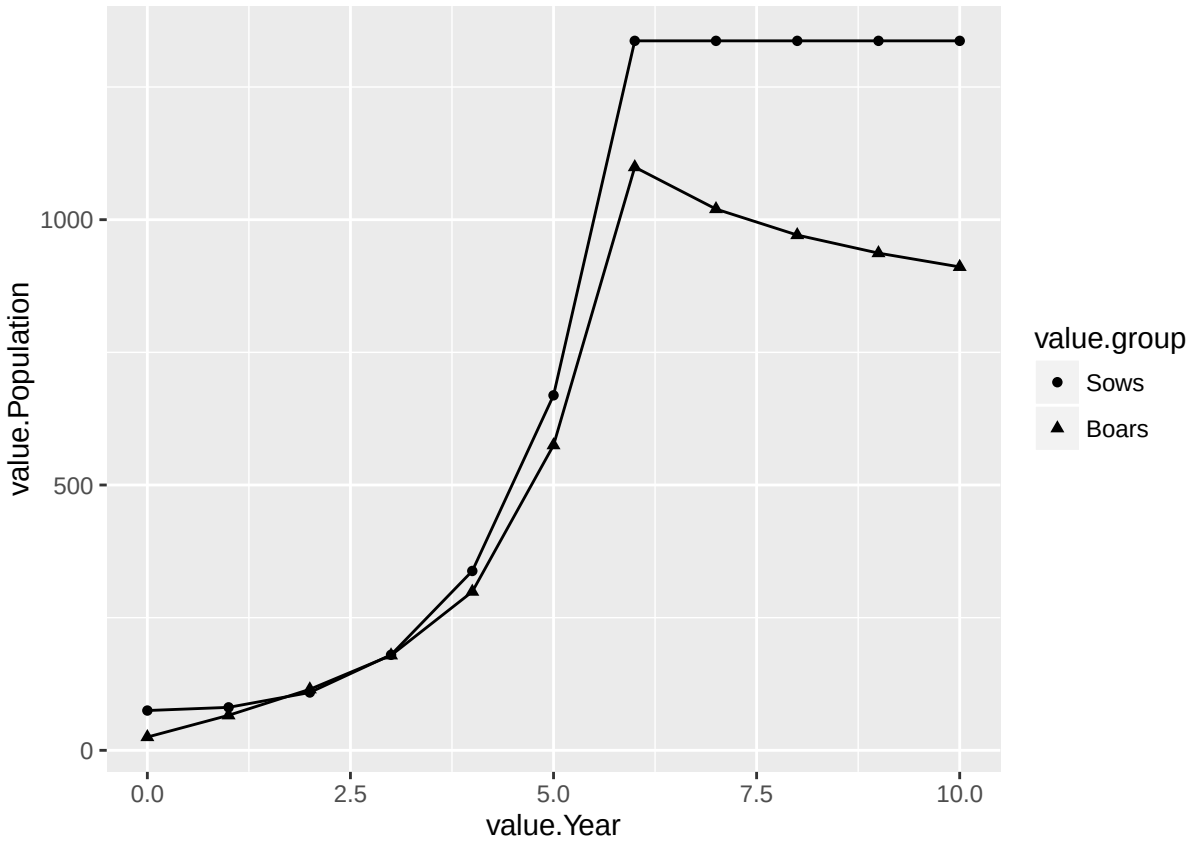
The base litter expectation al is then adjusted for land quality and last year's pressure on the patch where the sow is currently living.

$$al2 = (al * (1 + (1 - qu) - psil)) \tag{7}$$

The individualized litter is then split according to the expected sex ratio and rounded up to the nearest integer.

```
to breedR
  countPigs
      ask sows
      [
        let qu [quality] of patch-here
        let psil [psi_lastPeriod] of patch-here
        let al (1 - pigCountSum / (max_pigs)) * defaultLitter
        let al2 (al * (1 + qu - psil))
        let boarLitter round (al2 * sexRatio  + random-float 0.75)
        let sowLitter round (al2 * (1 - sexRatio))
        hatch-boars boarLitter [set age 0]
        hatch-sows sowLitter
        [
          set age 0
          ]
]
end
```

The resulting population trends when feral swine are not controlled appear to follow an expected population trend.



2.12 Counting Feral Swine and Presence

Most of the functions of this model rely on a perception of the number of feral swine on a patch and on the map. There are two primary kinds of counts in this model, count of head and count of presence. Count of head is counting the actual number of animals. We see that in the first part of the counting function sows and boars are counted on each patch, then added together for a total count.

```
to count_and_claim
  ;count sows
      ask patches [
      ifelse any? sows-here
      [set sowCount count sows-here]
      [set sowCount 0]
      ]
    set sowCountSum count sows
  ;count boars
    ask patches [
      ifelse any? boars-here
      [set boarCount count boars-here]
      [set boarCount 0]
      ]
    set boarCountSum count boars
  ;add them together
    set pigCountSum (boarCountSum + sowCountSum)
```

The next part establishes which sounder claims a given patch. The sounders simply claim patches that are in

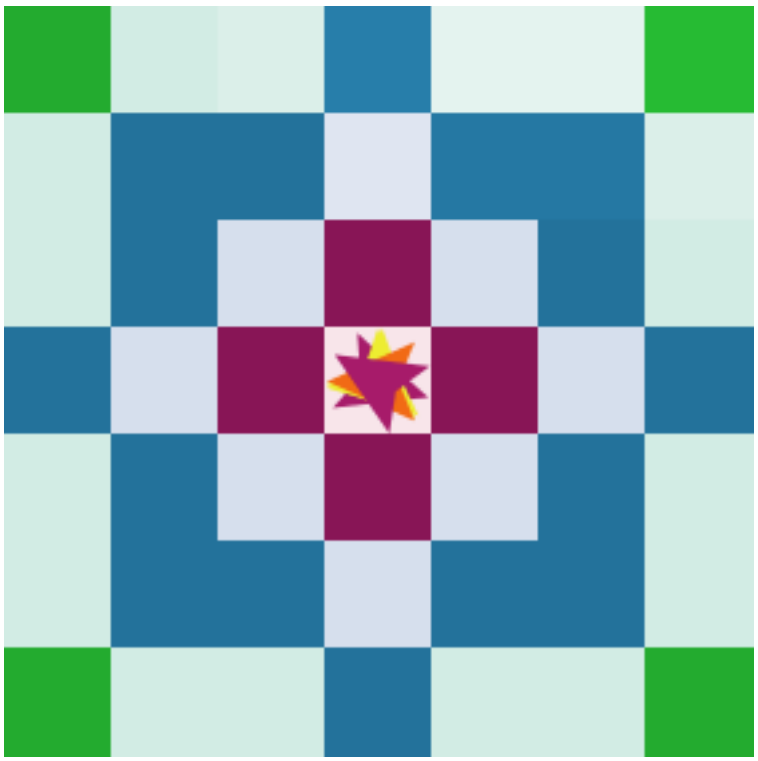


Figure 4: Example Feral Swine Presence

a radius of four patches. Sounders are presented each period in random order and the last sounder to claim a patch will have ownership of it. Movement of sounders is based on the outcome of `sounderTerritory`.

```
;establish sow territory
if any? sounders[
   ask sounders [
     if any? sows with [sounder_id = [sounder_id] of myself]
     [
       let sid sounder_id
       ask patches in-radius 4 [; range of 5
         set sounderTerritory sid
       ]
     ]
   ]
]
```

Sounders claim a location, but individual pigs have an impact on the landscape. As such presence is a essentially a count of full and partial days of pig presence. Illustrated in the figure below, we see a center patch with several feral swine located on it. That patch should have a presence of $n \times 1$, the four red patches around that will have a presence of $n \times 1/4$, the eight light blue patches outside of that have a presence of $n \times 1/8$, the 16 patches outside of that have a presence of $n \times 1/16$, the lighter blue patches outside of those have a presence of $n \times 1/32$, and the green patches in the corners are untouched. If the range is increased, the impact continues to halve each distance outward. The sixty-one patches with varying levels of impact make an approximately 2.47 km^2 home range. This is in the reported range of home ranges in the southeastern United States (Schlichting et al. 2016).

Applying this idea (with different colors) to the entire map in the figure below we see how presence compounds between animals across space.

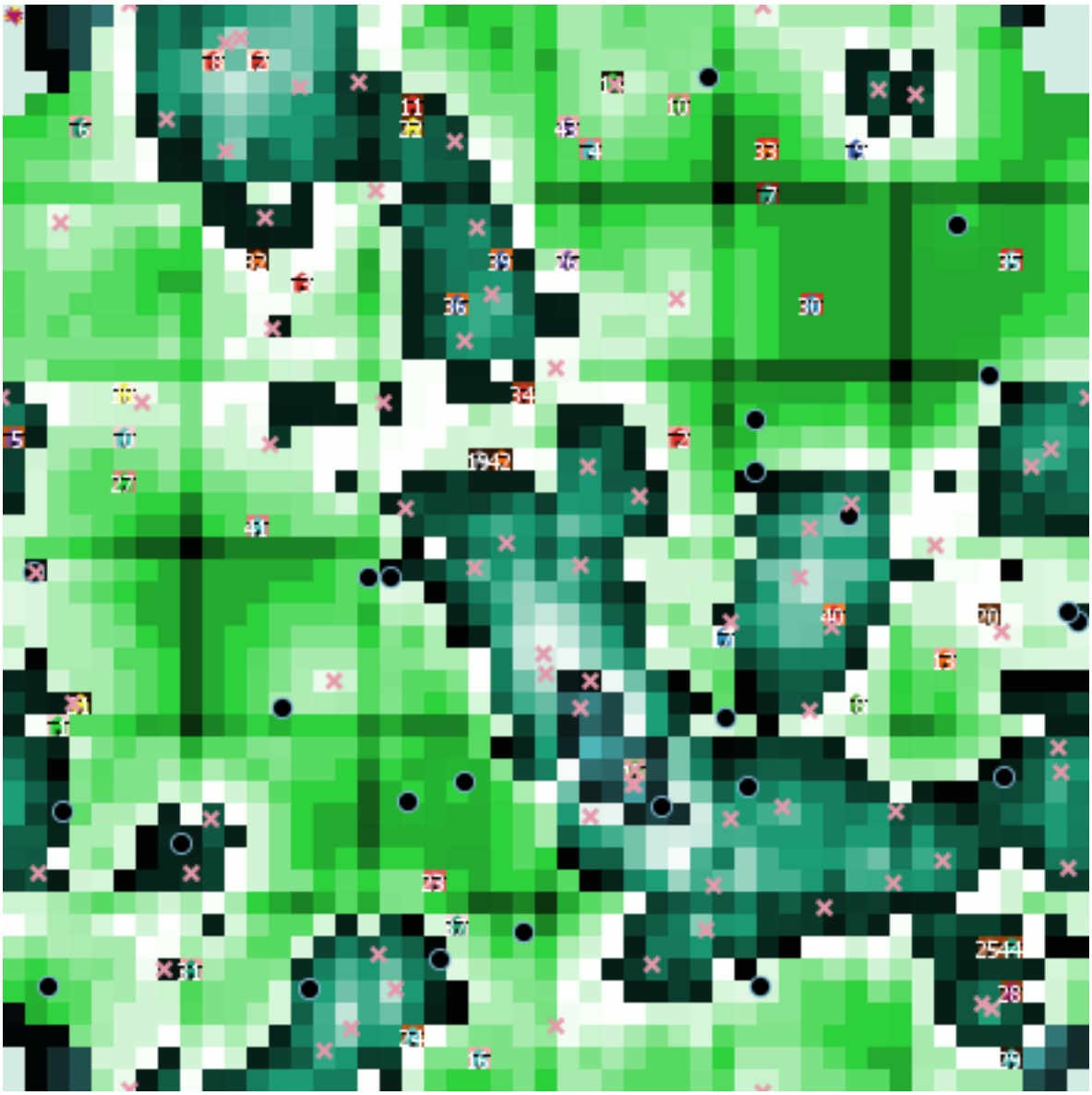


Figure 5: Example Feral Swine Presence - Entire Map

```
  ;establish the strength of sow territory
  ask patches [
    if any? sows [
      let sh count sows-here
      let s1 count sows in-radius 1
      let s2 count sows in-radius 2
      let s3 count sows in-radius 3
      let s4 count sows in-radius 4
      let s5 count sows in-radius 5
      set sts (sh +
        s1 - sh * (1 / 4) +
        s2 - s1 * (1 / 8) +
        s3 - s2 * (1 / 16) +
        s4 - s3 * (1 / 32) +
        s5 - s4 * (1 / 64))
      ]


    if any? boars [
      let bh count boars-here
      let b1 count boars in-radius 1
      let b2 count boars in-radius 2
      let b3 count boars in-radius 3
      let b4 count boars in-radius 4
      let b5 count boars in-radius 5
      set bts (bh +
        (b1 - bh) * (1 / 4) +
        (b2 - b1) * (1 / 8) +
        (b3 - b2) * (1 / 16) +
        (b4 - b3) * (1 / 32) +
        (b5 - b4) * (1 / 64))
    ]
      set pigTerritoryStrength (bts + sts + 2.22e-16)
      set pcolor pcolor + pigTerritoryStrength
      set boarTerritoryStrength bts
      set sounderTerritoryStrength sts
      set bts 0
      set sts 0
    ]
end
```

`pigTerritoryStrength`, `boarTerritoryStrength`, `sounderTerritoryStrength`, `boarCountSum`, and `sowCountSum` will be used for the major functions of the model.

2.13 Miscellaneous and Processing Functions

The following functions are functions that help process data or pass data. ### clearNL() Clear NL

```
#' @rdname WildPigABM
#' @export
clearNL <- function(verboseR,...){
  RNetLogo::NLCommand("clear-all")
  if (verboseR == TRUE){print("NL Cleared!")}
}
```

2.13.1 countDead() - Count killed feral swine

```
# Count the Dead --------------
#' @rdname WildPigABM
#' @export
countDead <- function(verboseR,...){
  RNetLogo::NLCommand('countDead')
  c <- RNetLogo::NLReport("killed-count")
  if (verboseR == TRUE) {print("dead counted!")}
  return(c)
}
```

```
to countDead
  if any? killed[
  set killed-count ((count killed) + killed-count)
  ]
end
```

2.13.2 countHouseholds() - Count households in NL and report back to R

```
#' @rdname WildPigABM
#' @examples \dontrun{countHouseholds()}
#' @export
countHouseholds <- function(){
  households <- RNetLogo::NLGetAgentSet("who", "households")
  return(households)
}
```

2.13.3 countOtherWildlife - Count the otherwildlife if turned on

```
#' @rdname WildPigABM
#' @export
countOtherWildlife <- function(map_height, map_width,...) {
  if (competingWildlife == TRUE){
    RNetLogo::NLCommand(
      "ask patches [
    set othercrittercount count othercritters-here
    ]"
    )# count patches with pigs etc
    array(data = RNetLogo::NLGetPatches("othercrittercount",
                                        as.data.frame = FALSE, as.matrix = TRUE),
          c(map_height, map_width))
  }
}
```

2.13.4 generateAndPassPV() - Generate and pass patch variables to NL

```
#' @rdname WildPigABM
#' @export
generateAndPassPV <- function(map_height,map_width,total_patches,
                              allow,
                              varCost_corn, varCost_soy, verCost_CRP,
```

```
                            P_corn, P_soy, P_CRP,
                            t_corn, t_soy, t_CRP,
                            Yield_soy, Yield_corn,
                            ...){
initialPatchVarArray <- array(data=0,c(map_height, map_width,8))
dimnames(initialPatchVarArray) <- list(paste("height",1:map_height,sep = ""),
                          paste("width",1:map_width,sep = ""),
                          c("quality","effect_comm","effect_free","effect_govt","effect_fee","memory",'

# generate quality variable for each patch
initialPatchVarArray[,,'quality']       <- array(data = stats::runif(total_patches, min = 0.5, max =
# generate effect_comm variable for each patch
initialPatchVarArray[,,'effect_comm']   <- array(data = stats::runif(total_patches, min = 0.5, max =
# generate effect_free variable for each patch
initialPatchVarArray[,,'effect_free']   <- array(data = stats::runif(total_patches, min = 0.5, max =
# generate effect_govt variable for each patch
initialPatchVarArray[,,'effect_govt']   <- array(data = stats::runif(total_patches, min = 0.5, max =
# generate effect_fee variable for each patch
initialPatchVarArray[,,'effect_fee']    <- array(data = stats::runif(total_patches, min = 0.9, max =
# generate memory variable for each patch
initialPatchVarArray[,,'memory']        <- array(data = stats::runif(total_patches, min = -0.25, max =
# generate patch_id variable for each patch
initialPatchVarArray[,,'patch_id']       <- array(data = 1:(total_patches), c(map_height, map_width))
# generate initial psi_t-1 variable for each patch
initialPatchVarArray[,,'psi_lastPeriod'] <- array(data = 2.220e-16, c(map_height, map_width))


NL_pushToMap("quality",initialPatchVarArray[,,'quality'],nl.obj = NULL)
NL_pushToMap("effect_comm",initialPatchVarArray[,,'effect_comm'],nl.obj = NULL)
NL_pushToMap("effect_free",initialPatchVarArray[,,'effect_free'],nl.obj = NULL)
NL_pushToMap("effect_govt",initialPatchVarArray[,,'effect_govt'],nl.obj = NULL)
NL_pushToMap("effect_fee",initialPatchVarArray[,,'effect_fee'],nl.obj = NULL)
NL_pushToMap("memory",initialPatchVarArray[,,'memory'] ,nl.obj = NULL)
NL_pushToMap("patch_id", initialPatchVarArray[,,'patch_id'],nl.obj = NULL)
NL_pushToMap("psi_lastPeriod",initialPatchVarArray[,,'psi_lastPeriod'] ,nl.obj = NULL)

#allow
RNetLogo::NLSetPatches('allowFee',allow[,,'fee'])
RNetLogo::NLSetPatches('allowFree',allow[,,"free"])
RNetLogo::NLSetPatches('allowPro',allow[,,'pro'])
#varcost
vca <- array(data=0,c(map_height, map_width,6))
dimnames(vca) <- list(paste("height",1:map_height,sep = ""),
                      paste("width",1:map_width,sep = ""),
                      c("Corn","Soy","CRP","SellPaid","Free","PayPaid"))
vca[,,'Corn']     <- array(data = varCost_corn, c(map_height, map_width))
vca[,,'Soy']      <- array(data = varCost_soy, c(map_height, map_width))
vca[,,'CRP']      <- array(data = verCost_CRP, c(map_height, map_width))
RNetLogo::NLSetPatches('varCost_corn',vca[,,'Corn'])
RNetLogo::NLSetPatches('varCost_soy',vca[,,'Soy'] )
RNetLogo::NLSetPatches('verCost_CRP',vca[,,'CRP'])
# update Prices-------------------------------------------------------------------------
prices <- array(data=0,c(map_height, map_width,6))
dimnames(prices) <- list(paste("height",1:map_height,sep = ""),
```

```
                          paste("width",1:map_width,sep = ""),
                          c("pCorn","pSoy","pCRP","pSellPaid","pFree","pPayPaid"))
  prices[,,'pCorn']     <- array(data = P_corn, c(map_height, map_width))
  prices[,,'pSoy']      <- array(data = P_soy, c(map_height, map_width))
  prices[,,'pCRP']      <- array(data = P_CRP, c(map_height, map_width))
  RNetLogo::NLSetPatches('P_corn',prices[,,'pCorn'])
  RNetLogo::NLSetPatches('P_soy',prices[,,'pSoy'] )
  RNetLogo::NLSetPatches('P_CRP',prices[,,'pCRP'])
  # create yield matrices-------------------------------------------------------------
  yield<-array(data=0,c(map_height, map_width,3))
  dimnames(yield) <- list(paste("height",1:map_height,sep = ""),
                          paste("width",1:map_width,sep = ""),
                          c("Yield_corn","Yield_soy","Yield_CRP"))
  yield[,,'Yield_corn']    <- matrix(data=Yield_corn,nrow=map_height,ncol=map_width)
  yield[,,'Yield_soy']    <- matrix(data=Yield_soy,nrow=map_height,ncol=map_width)
  yield[,,'Yield_CRP']    <- matrix(data=patchSize,nrow=map_height,ncol=map_width)
  RNetLogo::NLSetPatches('Yield_corn',yield[,,'Yield_corn'])
  RNetLogo::NLSetPatches('Yield_soy',yield[,,'Yield_soy'])
  RNetLogo::NLSetPatches('Yield_CRP',yield[,,'Yield_CRP'])
  # send time requirement to NL
  time <- array(0,c(map_height,map_width,6))
  dimnames(time) <- list(paste('height',1:map_height,sep = ""),
                         paste('width',1:map_width,sep = ""),
                         c('corn','soy','CRP','FEEPAYING',
                           'UNPAIDPRES','PAIDREMOVAL'))
  time[,,'corn'] <- array(data = t_corn, c(map_height,map_width))
  time[,,'soy'] <- array(data = t_soy, c(map_height,map_width))
  time[,,'CRP'] <- array(data = t_CRP, c(map_height,map_width))
  RNetLogo::NLSetPatches('t_soy',time[,,'soy'])
  RNetLogo::NLSetPatches('t_corn',time[,,'corn'])
  RNetLogo::NLSetPatches('t_CRP',time[,,'CRP'])



  return(initialPatchVarArray)
}
```

2.13.5 indivArraySpace() Create HH Result Space Arrays

```
#' @rdname WildPigABM
#' @export
indivArraySpace <- function(data, max_cycles,...){
  array(data, c(1, max_cycles))
}
```

2.13.6 is.installed() detect packages

```
#' @keywords wild pigs
#' @examples \dontrun{is.installed("cplexAPI")}
#' @rdname WildPigABM
#' @export
is.installed <- function(mypkg) is.element(mypkg, installed.packages()[,1])
```

2.13.7 mapArraySpace() Create Map Result Space Arrays

```
#' @rdname WildPigABM
#' @export
mapArraySpace <- function(data,map_height,map_width,max_cycles,...){
  array(data, c(map_height, map_width, max_cycles))
}
```

2.13.8 NL_pushToMap() Push characteristic to map in NL

```
#' @rdname WildPigABM
#' @export
NL_pushToMap <- function(NLcharacteristic,data,nl.obj = NULL, ...){
  RNetLogo::NLSetPatches(NLcharacteristic, data, nl.obj = NULL)
  if (verboseR == TRUE) {print("map data pushed to NL!")}
}
```

2.13.9 normalize() Normalize data to fall between 0 and 1

```
#' @rdname WildPigABM
#' @export
normalize <- function(x){(x-min(x) + 2.220e-15)/(max(x)-min(x) + 2.220e-15)}
```

2.13.10 setIterName() - set a name for each iteration

```
#' @rdname WildPigABM
#' @examples \dontrun{setIterName(t,now,verboseR=FALSE)}
#' @export
setIterName <- function(t, now, verboseR = FALSE, ...) {
  iterName <- paste("iteration_", t, "_", now, sep = "")
  if (verboseR == TRUE){print(iterName)}
  return(iterName)
}
```

2.14 GAMS Code For OCPM

```
cat(readLines('Untitled_6.gms'), sep = '\n')
```

```
Warning in readLines("Untitled_6.gms"): incomplete final line found on
'Untitled_6.gms'
```

```


Sets
 n1 /p1/
 t period    /0*75/
 k crop     /corn,soy,crp/  ;

Parameters

  Yield(k)
  /corn  66150
```

```
  soy    22250
  crp 490/
  P(k)
  /corn  3.71
  soy    9.15
  crp 75/
  VC(k)
  /corn  183260
  soy    115640
  crp 2.22e-16/
  c(k)
  /corn  0.0237
  soy    0.171
  crp 2.22e-16/
Quality(n1)
 / p1        =        1
/
  ;
Scalars

 r       discount rate    1 2 5 real rate
  /.05/
  gamma  reproductive rate  (60% baseline) [0 1]
  /0.70/
 Premoval
/4/
 maxD      "max density of FS - based on zivin 15pigs/km2 this is approx 2km2"
  /30/
initPigs initial number of FS
/21/
   ;
Variables
  use(k,n1,t)  Land use
  rP(n1,t)   remaining feral swine population
  P_removal(n1,t)
  q(n1,t)    removal effort
  dam(n1,t)   percent of crop damaged
  z  total management benefits
  ;

  use.lo(k,n1,t) = 0;
  use.up(k,n1,t) = 1;
  rP.LO(n1,t) = 0     ;
  q.LO(n1,t) = 0  ;


  alias (n1,n4);
  free Variable z ;
  positive variables dam(n1,t)    ;

Equations
  TDB TOTAL DISCOUNTED Benefits
  initPop(n1,t) Initial population
  damage(n1,t) pct damage as a function of population
```

```
  LUR(n1,t) patches can only be used for one crop
  CONTROL(n1,t)
  restric(n1,t)
*  pRem(n1,t)
  restric0(n1,t)
  pRem0(n1,t)
  priceRemaining(n1,t)
;

*Initial conditions t=0
  initPop(n1,t)$(ord(t) eq 1)  .. initPigs=e=rP(n1,t);
  restric0(n1,t)$(ord(t) eq 1) .. q(n1,t)=l=initPigs;
  pRem0(n1,t)$(ord(t)lt 2)     .. P_removal(n1,t)=e=10000  ;
*  t=1 to t=10
  CONTROL(n1,t)$(ord(t) ne 1)  .. rP(n1,t-1)+[rP(n1,t-1)*gamma*{1-[rP(n1,t-1)*maxD**(-1)]}]-q(n1,t)=e=r
  restric(n1,t)$(ord(t) ne 1)  .. q(n1,t)=l=rP(n1,t-1)   ;
  damage(n1,t)$(ord(t))        ..  0.9 + (-0.899)/(1 + (rP(n1,t)/14.21)**5)=e=dam(n1,t)  ;
*  pRem(n1,t)$(ord(t)ne 1)      .. (1165351988 + (72.00141 - 1165352000)/(1 + (q(n1,t)/11999.6)**2.2745
  priceRemaining(n1,t)$(ord(t)ge 2) .. 35.404 + (1464)/(1 + ((rP(n1,t))/3.299004)**1.843566)   =e= P_re

  LUR(n1,t)$(ord(t))       .. sum((k), use(k,n1,t))=l=1  ;


  TDB ..z=e=sum((k,n1,t),(
                         [[P(k)*Yield(k)*Quality(n1)-VC(k)]*use(k,n1,t) -
                         [P(k)*Yield(k)*Quality(n1)*dam(n1,t)]*use(k,n1,t)]-
                         (P_removal(n1,t)*q(n1,t))
                         )
                         *(1+r)**(ord(t)*(-1))) ;

Models
       Total       "Combined Land and Pig Mgmt Model" /all/   ;

*Total.iterlim = 4000;
*Option NLP = COINIPOPT
*Solve Total using NLP maximizing z;

parameter report(*,*,*)  "process level report" ;

initPigs  =  50  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result50.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
execute 'gdxxrw.exe Result50.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result50.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result50.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result50.gdx var=z.L rng=z.L!A1' ;

initPigs  =  40  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result40.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
```

```
execute 'gdxxrw.exe Result40.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result40.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result40.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result40.gdx var=z.L rng=z.L!A1' ;

initPigs  =  30  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result30.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
execute 'gdxxrw.exe Result30.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result30.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result30.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result30.gdx var=z.L rng=z.L!A1' ;

initPigs  =  20  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result20.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
execute 'gdxxrw.exe Result20.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result20.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result20.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result20.gdx var=z.L rng=z.L!A1' ;

initPigs  =  10  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result10.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
execute 'gdxxrw.exe Result10.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result10.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result10.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result10.gdx var=z.L rng=z.L!A1' ;

initPigs  =  6  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result6.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
execute 'gdxxrw.exe Result6.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result6.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result6.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result6.gdx var=z.L rng=z.L!A1';

initPigs  =  2  ;
Total.iterlim = 4000;
Option NLP = COINIPOPT
Solve Total using NLP maximizing z;
execute_unload "Result2.gdx" CONTROL.M use.L use.M   rP.L rP.M  q.L q.M  z.L initPigs;
execute 'gdxxrw.exe Result2.gdx equ=CONTROL.M rng=CONTROL.M!A1'
execute 'gdxxrw.exe Result2.gdx var=rP.L rng=rP.L!A1'
execute 'gdxxrw.exe Result2.gdx var=q.L rng=q.L!A1'
execute 'gdxxrw.exe Result2.gdx var=z.L rng=z.L!A1';
```

```
*"  DmgPop(n1,t)$(ord(t) ne 1)   .. [rP(n1,t)+sum((n4),rP(n4,t)*0.5)] =e= dpop(n1,t)  ;"

*" varCost_{Paid} =     "
*"  Bal(n1,t)$(ord(t) ne 1)      .. =e= mov(n1,t) ;           "
*" [rP(n1,t-1)+[rP(n1,t-1)*gamma*{1-[rP(n1,t-1)*maxD**(-1)]}]] -rP(n4,t)*q(n4,t)) -rP(n1,t)*q(n1,t)  "
```

2.15 NetLogo Complete Script

```
cat(readLines('~/GitHub/Version 2/WildPigABM/data/FromR_1.0.nlogo'),n = -1, sep = '\n')



breed [intersections intersection]
breed [killed one-killed]
breed [tk otk]
breed [households household]
breed [sounders sounder]
breed [boars boar]
breed [sows sow]
breed [pigs pig]
breed [otherCritters otherCritter]
breed [govt one-govt]

sounders-own [sounder_id
              age
              scount
]


households-own [
                HH_income  ;;
                hh_num  ;;
                wealthi  ;;
                ]

govt-own [policy]

boars-own [
               age  ;;
               bcagey  ;;
               mypsi ;;
               pig_num  ;;
               prRemoved
               ]
sows-own [
               age  ;;
               litter  ;;
               mypsi

               pig_num  ;;
               scagey  ;;
               sounder_id
```

```
                prRemoved
                ]

killed-own [
                pig_num  ;;
                age  ;;
                ]


patches-own [
  a_corn_heavy
  a_corn_light
  a_corn_none
  a_CRP_heavy
  a_CRP_light
  a_CRP_none
  a_heavy
  a_light
  a_none
  a_soy_heavy
  a_soy_light
  a_soy_none
  adjYieldCorn
  adjYieldCRP
  adjYieldSoy
  boarCount  ;;
  dam_corn_heavy
  dam_corn_light
  dam_corn_none
  dam_soy_heavy
  dam_soy_light
  dam_soy_none
  damage  ;;
  gm_corn_heavy
  gm_corn_light
  gm_corn_none
  gm_CRP_heavy
  gm_CRP_light
  gm_CRP_none
  gm_soy_heavy
  gm_soy_light
  gm_soy_none
  localAttractiveness  ;;
  localsounderTerritoryStrength  ;;
  maxPigsPerPatch
  my-column  ;;  the column of the intersection counting from the upper left corner of the world. -1 fo
  my-row  ;;  the row of the intersection counting from the upper left corner of the world. -1 for non-
  neighborCountBoar
  neighborCountSow
  no_damage_corn_GM
  no_damage_soy_GM
  notmine  ;;
  othercrittercount  ;;
  owner  ;;
```

```
  P_feePaying  ;;
  P_PaidRem  ;;
  parcel_NI  ;;
  patch_id  ;;
  PatchAttractivenessPriortoLandClaim
  pFree
  pigcount  ;;
  pigTerritoryStrength
  pop_if_heavy
  pop_if_light
  pop_if_none
  pPayPaid
  pSellPaid
  psi  ;;
  psi_lastPeriod  ;;
  quality  ;;
  removal_heavy
  removal_light
  removal_none
  rev_corn_heavy
  rev_corn_light
  rev_corn_none
  rev_soy_heavy
  rev_soy_light
  rev_soy_none
  road?  ;;  A patch that is roaded (but can still mostly be cropped)
  sounderCount  ;;
  sounderNcount  ;;
  sounderTerritory  ;;
  sounderTerritoryStrength  ;;
  tempCornDam
  tempSoyDam
  varCost_Fee  ;;
  varCost_free
  varCost_Paid
  verCost_CRP  ;;
  yield_corn_heavy
  yield_corn_light
  yield_corn_none
  yield_soy_heavy
  yield_soy_light
  yield_soy_none
  sts
  bts
  tkcount

              ]
globals [
  areThereAnySows;;
  boarCountSum  ;;
  boarTerritoryStrength ;;
  discount
  grid-x-inc  ;;  the amount of patches in between two roads in the x direction
  grid-y-inc  ;;  the amount of patches in between two roads in the y direction
```

```
  initBoarN  ;;
  initSounderN  ;;
  initSounderSize  ;;
  initSowN  ;;
  killed-count  ;;
  killed-mean  ;;
  max_cycles  ;;
  num_bases_per_year
  pigCountSum ;;
  sowCount ;;
  sowCountSum ;;
  t  ;;
  xval  ;;
  Yield_CRP  ;;
  yval  ;;

              ]
;;========================================== setup functions ======================================
to setup
  random-seed 28128868
  reset-ticks




  set-default-shape households "house"
  set-default-shape othercritters "dot"
  set-default-shape sows "dart"
  set-default-shape boars "circle 2"

  setup-patches
  setup-seededHH
  setup-hh
  setup-landgrab
  setup-roads

  setup-sounders

  setup-boars
  setup-sows
  ;setup-others
  ;setup-govt
end

to setup-sounders
  create-sounders init_sounder
  [
  set age 0
  set sounder_id who
  set scount 0
  setxy random-xcor random-ycor
  set color 60
  pd
  ]
```

```
end

to setup-others
  if competingWildlife = TRUE
  [
    create-othercritters 1000 [setxy random-xcor random-ycor]
  ]
end

to setup-patches
  ask patches [
    set maxPigsPerPatch ((max_pigs  / (2401 + 2.220e-15)) + allowableOverCap)
    set owner -1     ;;set land as unclaimed
    set pigcount 0   ;;sets the pig count on a parcel at 0
    set psi 0        ;;sets the initial total effective hunting effort at 0
    set psi_lastPeriod 0    ;;sets initial previous period hunting effort at 0
    set parcel_NI 0
    set pcolor 65
    ; # elementwise multiplication to find no damage yield
    set adjYieldCorn (Yield_corn  * quality); # set baseline corn yield per patch
    set adjYieldSoy (Yield_soy * quality);# set baseline soybean yield per patch
    ]
end

to setup-roads ;; PURELY VISUAL - NO OTHER CURRENT USE
; Put in north-south roads
    let xer min-pxcor + X-road-density
    while [ xer < max-pxcor ]
    [
    ask patches with [ pxcor = xer ] [
    set road? TRUE
    set pcolor  pcolor - 2
    ]
    set xer  xer + X-road-density
    ]
    ; Put in east-west roads
    let yer min-pycor + Y-road-density
    while [ yer < max-pycor ]
    [
    ask patches with [ pycor = yer ] [
    set road? TRUE
    set pcolor  pcolor - 2
    ]
    set yer  yer + Y-road-density
    ]
  print("roads created!")
end

to setup-seededHH
ask n-of init_hh patches [
    sprout-households 1
    ]
end
```

```
to setup-hh
  ask households [
    set hh_num who
    set wealthi 0
    set label  hh_num
    let inDreams hh_num
    ask patch-here [set owner inDreams set pcolor red] ; build a farmstead but you can farm here
    ]
end

to setup-landgrab
  while [any? patches with [owner = -1]][
    ask n-of init_hh patches with [owner != -1] [
    let tempowner owner
    if any? neighbors with [owner = -1][
    ask one-of neighbors with [owner = -1] [
    set owner tempowner
    set pcolor pcolor - 1
    ]
    ]
    ]
    ]
end

to setup-govt
  ask n-of 1 patches [
    sprout-govt 1]
end

to setup-boars
  set initBoarN ceiling (sexRatio * init_pigs)
  create-boars initBoarN
    [
    set size 1
    set color 97
    set age 0
    setxy random-xcor random-ycor
    pd
    ]
end


to setup-sows
  let n ceiling ((1 - sexRatio) * init_pigs) / init_sounder
  ask sounders [
  hatch-sows n
    [
      set pig_num self
      set size 1
      set color 136
      set age 0
      setxy random-xcor random-ycor
      pd
      ]
```

```
    ]

end
;;================================================ GO FUNCTIONS ========================================
to demo-run
  if ticks >= 10 [stop]
at-random
go-pigops
go-myopic-removal
go-wrap-up-myopic
killTheDead
tick
end

to go-pigops
    count_and_claim
  scarePigs
  breedR
  count_and_claim
  move-sounders
  count_and_claim
  moveBoars
  splitTheSounder
  smallSounderOps
end

to go-myopic-removal
  myopic-calcPsi
  setPrRemoved
  goRemoveBoars
  goRemoveSounders
  countDead
end

to go-wrap-up-myopic
  ;myopic-income
  real-income
  calculateHHIncome
end

to count_and_claim
  ;count sows
      ask patches [
      ifelse any? sows-here
      [set sowCount count sows-here]
      [set sowCount 0]
      ]
    set sowCountSum count sows
  ;count boars
    ask patches [
      ifelse any? boars-here
      [set boarCount count boars-here]
      [set boarCount 0]
      ]
```

```
  set boarCountSum count boars
;add them together
  set pigCountSum (boarCountSum + sowCountSum)
;establish sow territory
if any? sounders[
   ask sounders [
     if any? sows with [sounder_id = [sounder_id] of myself]
     [
       let sid sounder_id
       ask patches in-radius 4 [; range of 5
         set sounderTerritory sid
       ]
     ]
   ]
]

;establish the strength of sow territory
ask patches [
  if any? sows [
    let sh count sows-here
    let s1 count sows in-radius 1
    let s2 count sows in-radius 2
    let s3 count sows in-radius 3
    let s4 count sows in-radius 4
    let s5 count sows in-radius 5
    set sts (sh +
      s1 - sh * (1 / 4) +
      s2 - s1 * (1 / 8) +
      s3 - s2 * (1 / 16) +
      s4 - s3 * (1 / 32) +
      s5 - s4 * (1 / 64))
    ]
  if any? boars [
    let bh count boars-here
    let b1 count boars in-radius 1
    let b2 count boars in-radius 2
    let b3 count boars in-radius 3
    let b4 count boars in-radius 4
    let b5 count boars in-radius 5
    set bts (bh +
      (b1 - bh) * (1 / 4) +
      (b2 - b1) * (1 / 8) +
      (b3 - b2) * (1 / 16) +
      (b4 - b3) * (1 / 32) +
      (b5 - b4) * (1 / 64))
  ]
    set pigTerritoryStrength (bts + sts + 2.22e-16)
    if pigTerritoryStrength >= 10 [set pcolor 15]
    if floor pigTerritoryStrength = 9 [set pcolor 16]
    if floor pigTerritoryStrength = 8 [set pcolor 17]
    if floor pigTerritoryStrength = 7 [set pcolor 18]
    if floor pigTerritoryStrength = 6 [set pcolor 28]
    if floor pigTerritoryStrength = 5 [set pcolor 27]
    if floor pigTerritoryStrength = 4 [set pcolor 26]
```

```
      if floor pigTerritoryStrength = 3 [set pcolor 25]
      if floor pigTerritoryStrength = 2 [set pcolor 45]
      if floor pigTerritoryStrength = 1 [set pcolor 47]
      if floor pigTerritoryStrength = 0 [set pcolor 65]

      set boarTerritoryStrength bts
      set sounderTerritoryStrength sts
      set bts 0
      set sts 0
    ]
end

to breedR
      ask sows
      [
        let qu [quality] of patch-here
        let psil [psi_lastPeriod] of patch-here
        let al (1 - pigCountSum / (max_pigs)) * defaultLitter
        let al2 (al * (qu - psil))
        let boarLitter round (al2 * sexRatio  + random-float 0.5)
        show boarLitter
        let sowLitter round (al2 * (1 - sexRatio))
        show sowLitter
        hatch-boars boarLitter [set age 0]
        hatch-sows sowLitter   [set age 0]
]
end

to scarePigs
   if any? sows [
     ask sows [
       set scagey 1 / cdecay * scagey + [psi_lastPeriod] of patch-here
       show scagey
     ]
   ]
    if any? boars [
     ask boars [
       set bcagey 1 / cdecay * bcagey + [1 / 2 * psi_lastPeriod] of patch-here
       show bcagey
     ]
    ]
end

to prepareToMove
  ; land desirability is
        ;inversely related to population
        ;inversely related to pressure
        ;inversely related to other sounder's territory strength
        ;positively related to quality
     count_and_claim
     let maxperpatch (max_pigs / total_patches)
  ask patches[
    ; find patch populations, set as proportion to full capacity
```

```
    let mapPopulation boarCount + sowCount
    let x (mapPopulation / maxPigsPerPatch)
    ; recall quality
    let z quality
    ; establish attractiveness pre-land claim (additive)
    let tempattr ((1 / (x + 2.22e-16)) + (z))
    ; recall psi t-1
    let y psi_lastPeriod
    ifelse y < 0.01 ; inverse epsilon small numbers are very big and that is
    ;nonsense anyway
    [set PatchAttractivenessPriortoLandClaim tempattr]
    [set PatchAttractivenessPriortoLandClaim (tempattr + (1 / (y + 2.22e-16)))]
  ]
end

to move-sounders
  ask patches [set sounderTerritory -99]
  if any? sounders [
   prepareToMove
  ;set current territory

  ;evaluate attractiveness for each sounder ;sounders move
    ask sounders [
      let try 0
      while [sounderTerritory > 0 AND try < 40]
      [
        ask patches with [sounderTerritory < 0] [
          set localAttractiveness PatchAttractivenessPriortoLandClaim
          if localAttractiveness < 0.001 [set localAttractiveness 0]
          ]
        let ideal-site max-one-of patches in-radius srange [localAttractiveness]
        move-to ideal-site
        ;setxy random-xcor random-ycor
        set try try + 1
      ]
      let tempPig sounder_id
      ask patches in-radius srange
      [
        set sounderTerritory tempPig
      ]
    ]
    if any? sows with [sounder_id = sounder_id]
      [
        ask sows with [sounder_id = sounder_id][
          move-to sounder sounder_id
          ]
      ]
  ]

;
;        let nm (int sounderTerritory - int tempPig) ^ 2
;        ifelse nm > 0 [set notmine 1][set notmine 0]
;        set localsounderTerritoryStrength sounderTerritoryStrength * notmine
;        set localAttractiveness (PatchAttractivenessPriortoLandClaim - localsounderTerritoryStrength)
```

```
;        if localAttractiveness < 0.001 [set localAttractiveness 0]
;        ]
;
;
;
;
;
;        ]
;
;      ]
;    ;rinse and repeat
;    ;]
;  ]
end

to moveBoars
  if any? boars [
    ask boars [
      let b brange
      ask patches in-radius brange [
        set localsounderTerritoryStrength sounderTerritoryStrength * notmine
        set localAttractiveness (PatchAttractivenessPriortoLandClaim - localsounderTerritoryStrength)
        if localAttractiveness < 0.001 [set localAttractiveness 0]
      ]
      let ideal-site max-one-of patches in-radius b [localAttractiveness]
      move-to ideal-site
    ]
  ]
end

to splitTheSounder
  if any? sounders [
  let zz (maxSounderN + ALLOWABLEOVERCAP)
  ask sounders [
    set scount count sows with [sounder_id = [sounder_id] of myself]
    if scount > zz [
      let oldme who
      hatch-sounders 1 [
        set age 0
        set sounder_id who
        let sid sounder_id
        let op scount - zz
        ask n-of op sows with [sounder_id = oldme]
        [
          set sounder_id sid
          ]
      ]
    ]
  ]
  ]
end

to smallSounderOps
  if any? sounders [
```

```
    ask sounders
    [
      set scount count sows with [sounder_id = [sounder_id] of self]
      ]
    if any? sounders with [scount < minSounderN]
    [
      ask sows with [sounder_id = [sounder_id] of self]
      [
        set sounder_id [sounder_id] of min-one-of (other turtles) [ distance myself]
        ]
      ]
    ask sounders
    [
      set scount count sows with [sounder_id = [sounder_id] of myself]
      if scount < 1 [die]
      ]
    ]
end

;; goRemoveBoarsandSows ---------------------------------------------


to myopic-calcPsi
  ask patches [
    if a_corn_none > 0  [
      set psi a_corn_none * 0
      ]
    if a_corn_light > 0 [
      set psi a_corn_light * 0.25
      ]
    if a_corn_heavy > 0 [
      set psi a_corn_heavy * 0.90
      ]

    if a_soy_none > 0  [
      set psi a_soy_none * 0
      ]
    if a_soy_light > 0 [
      set psi a_soy_light * 0.25
      ]
    if a_soy_heavy > 0 [
      set psi a_soy_heavy * 0.90
      ]

    if a_crp_none > 0  [
      set psi a_crp_none * 0
      ]
    if a_crp_light > 0 [
      set psi a_crp_light * 0.25
      ]
    if a_crp_heavy > 0 [
      set psi a_crp_heavy * 0.90
      ]
   if psi > .999 [set psi .999]
```

```
   print psi
  ]
end

to setPrRemoved
  if any? boars [
    ask boars [
      set mypsi [psi] of patch-here
      set prRemoved (mypsi - mypsi * bcagey)
    ]
  ]
  if any? sows [
    ask sows [
      set mypsi [psi] of patch-here
      set prRemoved (mypsi - mypsi * scagey)
    ]
  ]
end

to goRemoveBoars
  if any? boars[
    ask boars [
      let rn random-float 1.0
      show (sentence "prRemoved" prRemoved "bcagey" bcagey "rn" rn)
      if rn < prRemoved [
      hatch-killed 1 [ht]
      hatch-tk 1 [ht]
      die
      ]
      ]
    ]
end

to goRemoveSounders
  if any? sows[
    ask sows[
      let rn random-float 1.0
      show (sentence "prRemoved" prRemoved "scagey" scagey "rn" rn)
      if rn < prRemoved [
        hatch-killed 1 [ht]
        hatch-tk 1 [ht]
        die
        ]
      ]
    ]
end

to countDead
  if any? killed[
  set killed-count count killed
  ]
end

;;-----------------------------------------------------------------------
```

```
to at-random
    ask patches[
      let choices int random 9
      if choices = 0 [
        set a_corn_none  1
        set a_corn_light 0
        set a_corn_heavy 0
        set a_soy_none  0
        set a_soy_light 0
        set a_soy_heavy 0
        set a_crp_none  0
        set a_crp_light 0
        set a_crp_heavy 0
        ]
      if choices = 1 [
        set a_corn_none  0
        set a_corn_light 1
        set a_corn_heavy 0
        set a_soy_none  0
        set a_soy_light 0
        set a_soy_heavy 0
        set a_crp_none  0
        set a_crp_light 0
        set a_crp_heavy 0
        ]
      if choices = 2 [
        set a_corn_none  0
        set a_corn_light 0
        set a_corn_heavy 1
        set a_soy_none  0
        set a_soy_light 0
        set a_soy_heavy 0
        set a_crp_none  0
        set a_crp_light 0
        set a_crp_heavy 0
        ]
      if choices = 3 [
        set a_corn_none  0
        set a_corn_light 0
        set a_corn_heavy 0
        set a_soy_none  1
        set a_soy_light 0
        set a_soy_heavy 0
        set a_crp_none  0
        set a_crp_light 0
        set a_crp_heavy 0
        ]
      if choices = 4 [
        set a_corn_none  0
        set a_corn_light 0
        set a_corn_heavy 0
        set a_soy_none  0
        set a_soy_light 1
```

```
      set a_soy_heavy 0
      set a_crp_none  0
      set a_crp_light 0
      set a_crp_heavy 0
      ]
    if choices = 5 [
      set a_corn_none  0
      set a_corn_light 0
      set a_corn_heavy 0
      set a_soy_none  0
      set a_soy_light 0
      set a_soy_heavy 1
      set a_crp_none  0
      set a_crp_light 0
      set a_crp_heavy 0
      ]
    if choices = 6 [
      set a_corn_none  0
      set a_corn_light 0
      set a_corn_heavy 0
      set a_soy_none  0
      set a_soy_light 0
      set a_soy_heavy 0
      set a_crp_none  1
      set a_crp_light 0
      set a_crp_heavy 0
      ]
    if choices = 7 [
      set a_corn_none  0
      set a_corn_light 0
      set a_corn_heavy 0
      set a_soy_none  0
      set a_soy_light 0
      set a_soy_heavy 0
      set a_crp_none  0
      set a_crp_light 1
      set a_crp_heavy 0
      ]
    if choices = 8 [
      set a_corn_none  0
      set a_corn_light 0
      set a_corn_heavy 0
      set a_soy_none  0
      set a_soy_light 0
      set a_soy_heavy 0
      set a_crp_none  0
      set a_crp_light 0
      set a_crp_heavy 1
      ]
    ]
end
```

```
to dynamic-prep-step1
  count_and_claim
  ask patches [
    let none  0
    let light  (pigTerritoryStrength * 0.25)   ;((8000 / 11 * (ln (-100 / 177 * -0.25 * 100))))
    let heavy  (pigTerritoryStrength * 0.90)  ;((8000 / 11 * (ln (-100 / 177 * -0.90 * 100))))
    set neighborCountBoar count (boars-on neighbors)
    set neighborCountSow count (sows-on neighbors)
    set no_damage_corn_GM P_corn * adjYieldCorn - varCost_corn
    set no_damage_soy_GM  P_soy  * adjYieldSoy - varCost_soy
    set pop_if_none     pigTerritoryStrength * (1 - none)
    set pop_if_light    pigTerritoryStrength * (1 - 0.25)
    set pop_if_heavy    pigTerritoryStrength * (1 - 0.90)
    set dam_corn_none   dam_Corn * pop_if_none
    if dam_corn_none  > 0.999 [set dam_corn_none 0.999]
    set dam_corn_light  dam_Corn * pop_if_light
    if dam_corn_light > 0.999 [set dam_corn_light 0.999]
    set dam_corn_heavy  dam_Corn * pop_if_heavy
    if dam_corn_heavy > 0.999 [set dam_corn_heavy 0.999]
    set dam_soy_none    dam_Soy  * pop_if_none
    if dam_soy_none   > 0.999 [set dam_soy_none 0.999]
    set dam_soy_light   dam_Soy  * pop_if_light
    if dam_soy_light  > 0.999 [set dam_soy_light 0.999]
    set dam_soy_heavy   dam_Soy  * pop_if_heavy
    if dam_soy_heavy  > 0.999 [set dam_soy_heavy 0.999]

    set yield_corn_none    adjYieldCorn * (1 - dam_corn_none)
    if yield_corn_none < 0  [set yield_corn_none  0]
    set yield_corn_light   adjYieldCorn * (1 - dam_corn_light)
    if yield_corn_light < 0 [set yield_corn_light 0]
    set yield_corn_heavy   adjYieldCorn * (1 - dam_corn_heavy)
    if yield_corn_heavy < 0 [set yield_corn_heavy 0]
    set yield_soy_none     adjYieldSoy  * (1 - dam_soy_none)
    if yield_soy_none < 0   [set yield_soy_none 0]
    set yield_soy_light    adjYieldSoy  * (1 - dam_soy_light)
    if yield_soy_light < 0  [set yield_soy_light 0]
    set yield_soy_heavy    adjYieldSoy  * (1 - dam_soy_heavy)
    if yield_soy_heavy < 0  [set yield_soy_heavy 0]

    set rev_corn_none   P_corn * yield_corn_none
    set rev_corn_light  P_corn * yield_corn_light
    set rev_corn_heavy  P_corn * yield_corn_heavy
    set rev_soy_none    P_soy  * yield_soy_none
    set rev_soy_light   P_soy  * yield_soy_light
    set rev_soy_heavy   P_soy  * yield_soy_heavy
    set removal_none   generic_removal_price * none
    set removal_light  generic_removal_price * light
    set removal_heavy  generic_removal_price * heavy
    set gm_corn_none   rev_corn_none  - varCost_corn
    set gm_corn_light  rev_corn_light - varCost_corn
    set gm_corn_heavy  rev_corn_heavy - varCost_corn
    set gm_soy_none    rev_soy_none   - varCost_soy
    set gm_soy_light   rev_soy_light  - varCost_soy
```

```
    set gm_soy_heavy   rev_soy_heavy  - varCost_soy
    set gm_CRP_none    P_CRP * 10 - verCost_CRP
    set gm_CRP_light   P_CRP * 10 - verCost_CRP
    set gm_CRP_heavy   P_CRP * 10 - verCost_CRP
     ]
end


to myopic-income
  count_and_claim
  ask patches [
    let none  0
    let light (pigTerritoryStrength * 0.25)  ;((8000 / 11 * (ln (-100 / 177 * -0.25 * 100))))
    let heavy (pigTerritoryStrength * 0.90)  ;((8000 / 11 * (ln (-100 / 177 * -0.90 * 100))))
    set tempCornDam  dam_Corn * pigTerritoryStrength
    if tempCornDam > 0.999 [set tempCornDam 0.999]
    set tempSoyDam   dam_Soy  * pigTerritoryStrength
    if tempSoyDam > 0.999 [set tempSoyDam 0.999]
    set removal_none   generic_removal_price * none
    set removal_light  generic_removal_price * light
    set removal_heavy  generic_removal_price * heavy

    if a_corn_none > 0  [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_none
      ]
    if a_corn_light > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_light
      ]
    if a_corn_heavy > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_heavy
      ]

    if a_soy_none > 0  [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_none
      ]
    if a_soy_light > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_light
      ]
    if a_soy_heavy > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_heavy
      ]

    if a_crp_none > 0  [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_none
      ]
    if a_crp_light > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_light
      ]
    if a_crp_heavy > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_heavy
      ]
    output-show (sentence "parcel_NI:" parcel_NI
                          "pigTerritoryStrength" pigTerritoryStrength
                          "removal_none" removal_none
```

```
                          "removal_light" removal_light
                          "removal_heavy" removal_heavy
                          "tempCornDam" tempCornDam
                          "tempSoyDam" tempSoyDam
                          "a_corn_none" a_corn_none
                          "a_corn_light" a_corn_light
                          "a_corn_heavy" a_corn_heavy
                          "a_soy_none" a_soy_none
                          "a_soy_light" a_soy_light
                          "a_soy_heavy" a_soy_heavy
                          "a_crp_none" a_crp_none
                          "a_crp_light" a_crp_light
                          "a_crp_heavy" a_crp_heavy)

  ]
end

to real-income
    count_and_claim
  ask patches [
    let none  0
    let light (pigTerritoryStrength * 0.25)
    let heavy (pigTerritoryStrength * 0.90)
    set tempCornDam    0.9 + (-0.899) / (1 + (pigTerritoryStrength / 14.21) ^ 5)
    if tempCornDam > 0.999 [set tempCornDam 0.999]
    set tempSoyDam     0.9 + (-0.899) / (1 + (pigTerritoryStrength / 14.21) ^ 5)
    if tempSoyDam > 0.999 [set tempSoyDam 0.999]
    set removal_none   0
    set removal_light  (35.404 + (1464) / (1 + (pigTerritoryStrength / 3.299004) ^ 1.843566))  * light
    set removal_heavy  (35.404 + (1464) / (1 + (pigTerritoryStrength / 3.299004) ^ 1.843566))  * heavy

    set tkcount count tk

    if a_corn_none > 0  [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_none + tkcount *
      ]
    if a_corn_light > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_light + tkcount
      ]
    if a_corn_heavy > 0 [
      set parcel_NI P_corn * adjYieldCorn * (1 - tempCornDam) - varCost_corn + removal_heavy + tkcount
      ]

    if a_soy_none > 0  [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_none + tkcount * bou
      ]
    if a_soy_light > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_light + tkcount * bo
      ]
    if a_soy_heavy > 0 [
      set parcel_NI P_soy  * adjYieldSoy * (1 - tempSoyDam) - varCost_soy + removal_heavy + tkcount * bo
      ]

    if a_crp_none > 0  [
```

```
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_none + tkcount * bounty
      ]
    if a_crp_light > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_light + tkcount * bounty
      ]
    if a_crp_heavy > 0 [
      set parcel_NI P_CRP  * 10 - verCost_CRP + removal_heavy + tkcount * bounty
      ]
    output-show (sentence "parcel_NI:" parcel_NI
                          "pigTerritoryStrength" pigTerritoryStrength
                          "removal_none" removal_none
                          "removal_light" removal_light
                          "removal_heavy" removal_heavy
                          "tempCornDam" tempCornDam
                          "tempSoyDam" tempSoyDam
                          "a_corn_none" a_corn_none
                          "a_corn_light" a_corn_light
                          "a_corn_heavy" a_corn_heavy
                          "a_soy_none" a_soy_none
                          "a_soy_light" a_soy_light
                          "a_soy_heavy" a_soy_heavy
                          "a_crp_none" a_crp_none
                          "a_crp_light" a_crp_light
                          "a_crp_heavy" a_crp_heavy)

  ]
end


;;-----------------------------------------------------------------------
to calculateHHIncome
  ask households [
    let aa sum [parcel_NI] of patches with [owner = [hh_num] of myself]
    set HH_income aa
    set wealthi wealthi + HH_income
    show (sentence "HH_income" HH_income "Weath" wealthi)
    ]

end


;;-----------------------------------------------------------------------

to killTheDead
  if any? boars  [ask boars [set age age + 1]]
  if any? sows [ ask sows [set age age + 1]]
  if any? sounders [ask sounders [set age age + 1]]
  if any? tk [ask tk [die]]

  ask patches [
    set psi_lastPeriod psi
    set psi 0
    ]
end
```

```
@#$#@#$#@
GRAPHICS-WINDOW
95
10
546
482
-1
-1
9.0
1
10
1
1
1
0
1
1
1
0
48
-48
0
0
0
1
ticks
30.0

PLOT
95
485
545
645
Count of Boars and Sounders
Time
Count
0.0
10.0
0.0
10.0
true
true
"" ""
PENS
"Sows" 1.0 0 -16777216 true "" "plot count sows"
"Boars" 1.0 0 -1604481 true "" "plot count boars"
"Killed " 1.0 0 -2674135 true "" "plot count killed"
"Turtles" 1.0 0 -955883 true "" "plot count turtles"
"Sounders" 1.0 0 -7500403 true "" "plot count sounders"

SLIDER
720
135
897
```

```
168
X-road-density
X-road-density
1
24
8
1
1
patches
HORIZONTAL

SLIDER
720
165
897
198
Y-road-density
Y-road-density
1
24
8
1
1
patches
HORIZONTAL

MONITOR
0
555
95
600
NIL
count killed
1
1
11

MONITOR
0
465
95
510
count sows
count sows
0
1
11

MONITOR
0
510
95
555
count boars
```

```
count boars
0
1
11

OUTPUT
5
645
545
805
11

SLIDER
550
475
725
508
minSounderN
minSounderN
0
10
15
1
1
NIL
HORIZONTAL

SLIDER
550
15
725
48
allowableOverCap
allowableOverCap
0
100
10
1
1
NIL
HORIZONTAL

SLIDER
550
45
725
78
boarTerritorydecay
boarTerritorydecay
0
10
1
1
1
```

```
NIL
HORIZONTAL

SLIDER
550
75
725
108
brange
brange
0
100
5
1
1
NIL
HORIZONTAL

CHOOSER
550
105
725
150
competingWildlife
competingWildlife
"TRUE" "FALSE"
1

SLIDER
550
150
725
183
dam_Corn
dam_Corn
0
1
0.0237
.0001
1
NIL
HORIZONTAL

SLIDER
550
180
725
213
dam_soy
dam_soy
0
1
0.0171
.0001
```

```
1
NIL
HORIZONTAL

SLIDER
550
210
725
243
defaultLitter
defaultLitter
0
2
0.8
0.1
1
NIL
HORIZONTAL

SLIDER
550
240
725
273
generic_removal_price
generic_removal_price
-100
100
35
1
1
NIL
HORIZONTAL

CHOOSER
550
270
725
315
Govt_ON
Govt_ON
"TRUE" "FALSE"
1

SLIDER
550
315
727
348
init_hh
init_hh
45
45
45
```

```
1
1
DO NOT CHANGE
HORIZONTAL

SLIDER
550
345
725
378
init_pigs
init_pigs
0
1000
100
1
1
NIL
HORIZONTAL

SLIDER
550
375
725
408
max_pigs
max_pigs
0
2000
1440
1
1
NIL
HORIZONTAL

SLIDER
550
405
725
438
maxSounderN
maxSounderN
0
100
50
1
1
NIL
HORIZONTAL

SLIDER
550
440
725
```

```
473
mergeAccelerator
mergeAccelerator
0
1
0.25
.01
1
NIL
HORIZONTAL

SLIDER
550
510
725
543
P_corn
P_corn
0
5
3.71
.01
1
NIL
HORIZONTAL

SLIDER
550
540
725
573
P_CRP
P_CRP
0
200
0
1
1
NIL
HORIZONTAL

SLIDER
550
575
725
608
P_soy
P_soy
0
10
9.15
0.01
1
NIL
```

```
HORIZONTAL

SLIDER
550
610
725
643
pig_range
pig_range
0
100
10
1
1
NIL
HORIZONTAL

SLIDER
550
640
725
673
sexRatio
sexRatio
0
1
0.25
.01
1
NIL
HORIZONTAL

SLIDER
550
670
725
703
SounderterritoryDecay
SounderterritoryDecay
0
10
1
1
1
NIL
HORIZONTAL

SLIDER
550
700
725
733
splitAccelerator
splitAccelerator
```

```
0
1
0.25
.01
1
NIL
HORIZONTAL

SLIDER
550
730
725
763
srange
srange
0
100
5
1
1
NIL
HORIZONTAL

SLIDER
550
765
725
798
varCost_corn
varCost_corn
0
5000
0
1
1
NIL
HORIZONTAL

SLIDER
725
15
897
48
varCost_soy
varCost_soy
0
5000
0
1
1
NIL
HORIZONTAL

SLIDER
```

```
725
45
897
78
wealth
wealth
0
1000000
0
1
1
NIL
HORIZONTAL

SLIDER
725
75
897
108
Yield_corn
Yield_corn
0
2000
1350
1
1
NIL
HORIZONTAL

SLIDER
725
105
897
138
Yield_soy
Yield_soy
0
1000
450
1
1
NIL
HORIZONTAL

BUTTON
3
10
93
43
setup
ca\nsetup
NIL
1
T
```

```
OBSERVER
NIL
NIL
NIL
NIL
1

SLIDER
725
200
897
233
init_sounder
init_sounder
0
10
5
1
1
NIL
HORIZONTAL

BUTTON
5
45
92
78
Demo-Run
demo-run
T
1
T
OBSERVER
NIL
NIL
NIL
NIL
1

SLIDER
725
235
900
268
total_patches
total_patches
2401
2401
2401
1
1
DO NOT CHANGE
HORIZONTAL
```

```
SLIDER
725
270
897
303
cdecay
cdecay
0
100
4
1
1
NIL
HORIZONTAL

SLIDER
725
300
897
333
bounty
bounty
0
100
50
10
1
NIL
HORIZONTAL

@#$#@#$#@
## WHAT IS IT?
This paper will demonstrate the importance of the interaction between individuals across time and space


## CREDITS AND REFERENCES
http://modelingcommons.org/browse/one_model/2328#model_tabs_browse_applet
Uri Wilensky (Author)
Patch Clusters Example


@#$#@#$#@
default
true
0
Polygon -7500403 true true 150 5 40 250 150 205 260 250

airplane
true
0
Polygon -7500403 true true 150 0 135 15 120 60 120 105 15 165 15 195 120 180 135 240 105 270 120 285 150

arrow
true
```

```
0
Polygon -7500403 true true 150 0 0 150 105 150 105 293 195 293 195 150 300 150

box
false
0
Polygon -7500403 true true 150 285 285 225 285 75 150 135
Polygon -7500403 true true 150 135 15 75 150 15 285 75
Polygon -7500403 true true 15 75 15 225 150 285 150 135
Line -16777216 false 150 285 150 135
Line -16777216 false 150 135 15 75
Line -16777216 false 150 135 285 75

bug
true
0
Circle -7500403 true true 96 182 108
Circle -7500403 true true 110 127 80
Circle -7500403 true true 110 75 80
Line -7500403 true 150 100 80 30
Line -7500403 true 150 100 220 30

butterfly
true
0
Polygon -7500403 true true 150 165 209 199 225 225 225 255 195 270 165 255 150 240
Polygon -7500403 true true 150 165 89 198 75 225 75 255 105 270 135 255 150 240
Polygon -7500403 true true 139 148 100 105 55 90 25 90 10 105 10 135 25 180 40 195 85 194 139 163
Polygon -7500403 true true 162 150 200 105 245 90 275 90 290 105 290 135 275 180 260 195 215 195 162 165
Polygon -16777216 true false 150 255 135 225 120 150 135 120 150 105 165 120 180 150 165 225
Circle -16777216 true false 135 90 30
Line -16777216 false 150 105 195 60
Line -16777216 false 150 105 105 60

car
false
0
Polygon -7500403 true true 300 180 279 164 261 144 240 135 226 132 213 106 203 84 185 63 159 50 135 50 
Circle -16777216 true false 180 180 90
Circle -16777216 true false 30 180 90
Polygon -16777216 true false 162 80 132 78 134 135 209 135 194 105 189 96 180 89
Circle -7500403 true true 47 195 58
Circle -7500403 true true 195 195 58

circle
false
0
Circle -7500403 true true 0 0 300

circle 2
false
0
Circle -7500403 true true 0 0 300
Circle -16777216 true false 30 30 240
```

```
cow
false
0
Polygon -7500403 true true 200 193 197 249 179 249 177 196 166 187 140 189 93 191 78 179 72 211 49 209 4
Polygon -7500403 true true 73 210 86 251 62 249 48 208
Polygon -7500403 true true 25 114 16 195 9 204 23 213 25 200 39 123

cylinder
false
0
Circle -7500403 true true 0 0 300

dart
true
0
Polygon -7500403 true true 135 90 150 285 165 90
Polygon -7500403 true true 135 285 105 255 105 240 120 210 135 180 150 165 165 180 180 210 195 240 195 2
Rectangle -1184463 true false 135 45 165 90
Line -16777216 false 150 285 150 180
Polygon -16777216 true false 150 45 135 45 146 35 150 0 155 35 165 45
Line -16777216 false 135 75 165 75
Line -16777216 false 135 60 165 60

dot
false
0
Circle -7500403 true true 90 90 120

face happy
false
0
Circle -7500403 true true 8 8 285
Circle -16777216 true false 60 75 60
Circle -16777216 true false 180 75 60
Polygon -16777216 true false 150 255 90 239 62 213 47 191 67 179 90 203 109 218 150 225 192 218 210 203

face neutral
false
0
Circle -7500403 true true 8 7 285
Circle -16777216 true false 60 75 60
Circle -16777216 true false 180 75 60
Rectangle -16777216 true false 60 195 240 225

face sad
false
0
Circle -7500403 true true 8 8 285
Circle -16777216 true false 60 75 60
Circle -16777216 true false 180 75 60
Polygon -16777216 true false 150 168 90 184 62 210 47 232 67 244 90 220 109 205 150 198 192 205 210 220

fish
```

```
false
0
Polygon -1 true false 44 131 21 87 15 86 0 120 15 150 0 180 13 214 20 212 45 166
Polygon -1 true false 135 195 119 235 95 218 76 210 46 204 60 165
Polygon -1 true false 75 45 83 77 71 103 86 114 166 78 135 60
Polygon -7500403 true true 30 136 151 77 226 81 280 119 292 146 292 160 287 170 270 195 195 210 151 212
Circle -16777216 true false 215 106 30

flag
false
0
Rectangle -7500403 true true 60 15 75 300
Polygon -7500403 true true 90 150 270 90 90 30
Line -7500403 true 75 135 90 135
Line -7500403 true 75 45 90 45

flower
false
0
Polygon -10899396 true false 135 120 165 165 180 210 180 240 150 300 165 300 195 240 195 195 165 135
Circle -7500403 true true 85 132 38
Circle -7500403 true true 130 147 38
Circle -7500403 true true 192 85 38
Circle -7500403 true true 85 40 38
Circle -7500403 true true 177 40 38
Circle -7500403 true true 177 132 38
Circle -7500403 true true 70 85 38
Circle -7500403 true true 130 25 38
Circle -7500403 true true 96 51 108
Circle -16777216 true false 113 68 74
Polygon -10899396 true false 189 233 219 188 249 173 279 188 234 218
Polygon -10899396 true false 180 255 150 210 105 210 75 240 135 240

house
false
0
Rectangle -7500403 true true 45 120 255 285
Rectangle -16777216 true false 120 210 180 285
Polygon -7500403 true true 15 120 150 15 285 120
Line -16777216 false 30 120 270 120

leaf
false
0
Polygon -7500403 true true 150 210 135 195 120 210 60 210 30 195 60 180 60 165 15 135 30 120 15 105 40
Polygon -7500403 true true 135 195 135 240 120 255 105 255 105 285 135 285 165 240 165 195

line
true
0
Line -7500403 true 150 0 150 300

line half
true
```

```
0
Line -7500403 true 150 0 150 150

pentagon
false
0
Polygon -7500403 true true 150 15 15 120 60 285 240 285 285 120

person
false
0
Circle -7500403 true true 110 5 80
Polygon -7500403 true true 105 90 120 195 90 285 105 300 135 300 150 225 165 300 195 300 210 285 180 195
Rectangle -7500403 true true 127 79 172 94
Polygon -7500403 true true 195 90 240 150 225 180 165 105
Polygon -7500403 true true 105 90 60 150 75 180 135 105

plant
false
0
Rectangle -7500403 true true 135 90 165 300
Polygon -7500403 true true 135 255 90 210 45 195 75 255 135 285
Polygon -7500403 true true 165 255 210 210 255 195 225 255 165 285
Polygon -7500403 true true 135 180 90 135 45 120 75 180 135 210
Polygon -7500403 true true 165 180 165 210 225 180 255 120 210 135
Polygon -7500403 true true 135 105 90 60 45 45 75 105 135 135
Polygon -7500403 true true 165 105 165 135 225 105 255 45 210 60
Polygon -7500403 true true 135 90 120 45 150 15 180 45 165 90

sheep
false
15
Circle -1 true true 203 65 88
Circle -1 true true 70 65 162
Circle -1 true true 150 105 120
Polygon -7500403 true false 218 120 240 165 255 165 278 120
Circle -7500403 true false 214 72 67
Rectangle -1 true true 164 223 179 298
Polygon -1 true true 45 285 30 285 30 240 15 195 45 210
Circle -1 true true 3 83 150
Rectangle -1 true true 65 221 80 296
Polygon -1 true true 195 285 210 285 210 240 240 210 195 210
Polygon -7500403 true false 276 85 285 105 302 99 294 83
Polygon -7500403 true false 219 85 210 105 193 99 201 83

square
false
0
Rectangle -7500403 true true 30 30 270 270

square 2
false
0
Rectangle -7500403 true true 30 30 270 270
```

```
Rectangle -16777216 true false 60 60 240 240

star
false
0
Polygon -7500403 true true 151 1 185 108 298 108 207 175 242 282 151 216 59 282 94 175 3 108 116 108

target
false
0
Circle -7500403 true true 0 0 300
Circle -16777216 true false 30 30 240
Circle -7500403 true true 60 60 180
Circle -16777216 true false 90 90 120
Circle -7500403 true true 120 120 60

tree
false
0
Circle -7500403 true true 118 3 94
Rectangle -6459832 true false 120 195 180 300
Circle -7500403 true true 65 21 108
Circle -7500403 true true 116 41 127
Circle -7500403 true true 45 90 120
Circle -7500403 true true 104 74 152

triangle
false
0
Polygon -7500403 true true 150 30 15 255 285 255

triangle 2
false
0
Polygon -7500403 true true 150 30 15 255 285 255
Polygon -16777216 true false 151 99 225 223 75 224

truck
false
0
Rectangle -7500403 true true 4 45 195 187
Polygon -7500403 true true 296 193 296 150 259 134 244 104 208 104 207 194
Rectangle -1 true false 195 60 195 105
Polygon -16777216 true false 238 112 252 141 219 141 218 112
Circle -16777216 true false 234 174 42
Rectangle -7500403 true true 181 185 214 194
Circle -16777216 true false 144 174 42
Circle -16777216 true false 24 174 42
Circle -7500403 false true 24 174 42
Circle -7500403 false true 144 174 42
Circle -7500403 false true 234 174 42

turtle
true
```

```
0
Polygon -10899396 true false 215 204 240 233 246 254 228 266 215 252 193 210
Polygon -10899396 true false 195 90 225 75 245 75 260 89 269 108 261 124 240 105 225 105 210 105
Polygon -10899396 true false 105 90 75 75 55 75 40 89 31 108 39 124 60 105 75 105 90 105
Polygon -10899396 true false 132 85 134 64 107 51 108 17 150 2 192 18 192 52 169 65 172 87
Polygon -10899396 true false 85 204 60 233 54 254 72 266 85 252 107 210
Polygon -7500403 true true 119 75 179 75 209 101 224 135 220 225 175 261 128 261 81 224 74 135 88 99

wheel
false
0
Circle -7500403 true true 3 3 294
Circle -16777216 true false 30 30 240
Line -7500403 true 150 285 150 15
Line -7500403 true 15 150 285 150
Circle -7500403 true true 120 120 60
Line -7500403 true 216 40 79 269
Line -7500403 true 40 84 269 221
Line -7500403 true 40 216 269 79
Line -7500403 true 84 40 221 269

wolf
false
0
Polygon -16777216 true false 253 133 245 131 245 133
Polygon -7500403 true true 2 194 13 197 30 191 38 193 38 205 20 226 20 257 27 265 38 266 40 260 31 253 3
Polygon -7500403 true true -1 195 14 180 36 166 40 153 53 140 82 131 134 133 159 126 188 115 227 108 230

x
false
0
Polygon -7500403 true true 270 75 225 30 30 225 75 270
Polygon -7500403 true true 30 75 75 30 270 225 225 270

@#$#@#$#@
NetLogo 5.3.1
@#$#@#$#@
@#$#@#$#@
@#$#@#$#@
<experiments>
  <experiment name="experiment" repetitions="10" runMetricsEveryStep="true">
    <setup>setup</setup>
    <go>Demo-Run</go>
    <metric>if any? sows [count sows]</metric>
    <metric>if any? boars [count boars]</metric>
    <metric>if any? killed [count killed]</metric>
    <metric>min [wealthi] of households</metric>
    <metric>max [wealthi] of households</metric>
    <metric>mean [wealthi] of households</metric>
    <metric>median [wealthi] of households</metric>
    <steppedValueSet variable="P_CRP" first="0" step="25" last="50"/>
    <steppedValueSet variable="sexRatio" first="0.1" step="0.2" last="0.9"/>
    <steppedValueSet variable="varCost_soy" first="0" step="500" last="2000"/>
    <enumeratedValueSet variable="srange">
```

```
  <value value="5"/>
</enumeratedValueSet>
<enumeratedValueSet variable="init_pigs">
  <value value="2"/>
  <value value="100"/>
  <value value="503"/>
</enumeratedValueSet>
<enumeratedValueSet variable="total_patches">
  <value value="2401"/>
</enumeratedValueSet>
<enumeratedValueSet variable="allowableOverCap">
  <value value="10"/>
</enumeratedValueSet>
<enumeratedValueSet variable="SounderterritoryDecay">
  <value value="1"/>
</enumeratedValueSet>
<enumeratedValueSet variable="boarTerritorydecay">
  <value value="1"/>
</enumeratedValueSet>
<enumeratedValueSet variable="wealth">
  <value value="0"/>
</enumeratedValueSet>
<enumeratedValueSet variable="splitAccelerator">
  <value value="0.25"/>
</enumeratedValueSet>
<enumeratedValueSet variable="brange">
  <value value="5"/>
</enumeratedValueSet>
<enumeratedValueSet variable="competingWildlife">
  <value value="&amp;quot;FALSE&amp;quot;"/>
</enumeratedValueSet>
<enumeratedValueSet variable="Yield_corn">
  <value value="1350"/>
</enumeratedValueSet>
<enumeratedValueSet variable="generic_removal_price">
  <value value="35"/>
</enumeratedValueSet>
<enumeratedValueSet variable="dam_soy">
  <value value="0.0171"/>
</enumeratedValueSet>
<enumeratedValueSet variable="P_corn">
  <value value="3.71"/>
</enumeratedValueSet>
<enumeratedValueSet variable="max_pigs">
  <value value="1440"/>
</enumeratedValueSet>
<enumeratedValueSet variable="maxSounderN">
  <value value="50"/>
</enumeratedValueSet>
<enumeratedValueSet variable="minSounderN">
  <value value="15"/>
</enumeratedValueSet>
<enumeratedValueSet variable="P_soy">
  <value value="9.15"/>
```

```
    </enumeratedValueSet>
    <enumeratedValueSet variable="dam_Corn">
      <value value="0.0237"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="Yield_soy">
      <value value="450"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="Govt_ON">
      <value value="&quot;FALSE&quot;"/>
    </enumeratedValueSet>
    <steppedValueSet variable="varCost_corn" first="0" step="500" last="2000"/>
    <enumeratedValueSet variable="init_sounder">
      <value value="5"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="Y-road-density">
      <value value="8"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="defaultLitter">
      <value value="0.8"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="X-road-density">
      <value value="8"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="pig_range">
      <value value="10"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="init_hh">
      <value value="45"/>
    </enumeratedValueSet>
    <enumeratedValueSet variable="mergeAccelerator">
      <value value="0.25"/>
    </enumeratedValueSet>
  </experiment>
</experiments>
@#$#@#$#@
@#$#@#$#@
default
0.0
-0.2 0 0.0 1.0
0.0 1 1.0 0.0
0.2 0 0.0 1.0
link direction
true
0
Line -7500403 true 150 150 90 180
Line -7500403 true 150 150 210 180

@#$#@#$#@
1
@#$#@#$#@
-1
```

References

Anderson, A., C. Slootmaker, E. Harper, J. Holderieath, and S.A. Shwiff. 2016. "Economic estimates of feral swine damage and control in 11 US states." *Crop Protection* 89. doi:10.1016/j.cropro.2016.06.023.

Berck, P., and G. Helfand. 2011. *The Economics of the Environment*. Pearson Addison-Wesley.

Bieber, Claudia, and Thomas Ruf. 2005. "Population Dynamics in Wild Boar Sus Scrofa: Ecology, Elasticity of Growth Rate and Implications for the Management of Pulsed Resource Consumers." *Journal of Applied Ecology* 42 (6). Wiley Online Library: 1203–13.

Bodenchuk, Michael J. 2014. "Method specific costs of feral swine removal in a large metapopulation." p.c.

Carson, Rhys. 2013. "'Boared' to death: Rooting out the feral hog issue in Texas." *Texas Tech Administrative Law Journal* Summer 2013 (c): 1–32.

Field, B.C. 2001. *Natural Resource Economics: An Introduction*. Waveland Press.

Geisser, Hannes, and Heinz-Ulrich Reyer. 2005. "The Influence of Food and Temperature on Population Density of Wild Boar Sus Scrofa in the Thurgau (Switzerland)." *Journal of Zoology* 267 (1). Wiley Online Library: 89–96.

Hainds, Mark J. 2011. *Year of the Pig*. Tuscaloosa, AL: The University of Alabama Press.

Hanson, Laura B, Michael S Mitchell, James B Grand, D Buck Jolley, Bill D Sparklin, and Stephen S Ditchkoff. 2009. "Effect of Experimental Manipulation on Survival and Recruitment of Feral Pigs." *Wildlife Research* 36 (3). CSIRO: 185–91.

Hone, Jim. 1995. "Spatial and temporal aspects of vertebrate pest damage with emphasis on feral pigs." *Journal of Applied Ecology* 32 (2): 311–19. doi:10.2307/2405098.

Moss, C.B. 2013. *Agricultural Finance*. Routledge Textbooks in Environmental and Agricultural Economics. Taylor & Francis.

Railsback, Steven F, and Volker Grimm. 2011. *Agent-based and individual-based modeling: a practical introduction*. Princeton university press.

Saunders, G, and H Bryant. 1988. "The evaluation of a feral pig eradication program during a simulated exotic disease outbreak." *Australian Wildlife Research* 15 (1): 73–81. doi:10.1071/WR9880073.

Schlichting, Peter E., Sarah R. Fritts, John J. Mayer, Philip S. Gipson, and C. Brad Dabbert. 2016. "Determinants of variation in home range of wild pigs." *Wildlife Society Bulletin* 40 (3): 487–93. doi:10.1002/wsb.662.

Shi, Wei, and Scott H Irwin. 2005. "Optimal Hedging with a Subjective View: An Empirical Bayesian Approach." *American Journal of Agricultural Economics* 87 (4). Oxford University Press USA: 918–30.

Thijssen, Geert. 1996. "Farmers' investment behavior: An empirical assessment of two specifications of expectations." *American Journal of Agricultural Economics* 78 (1). Oxford University Press: 166–74.

Zivin, Joshua, Brent M Hueth, and David Zilberman. 2000. "Managing a Multiple-Use Resource: The Case of Feral Pig Management in California Rangeland." *Journal of Environmental Economics and Management* 39 (2): 189–204. doi:http://dx.doi.org/10.1006/jeem.1999.1101.