Mathematical representation of the drought decision model

Trisha Shrum

January 25, 2017

1 Overview

2 Indices

- Time is indexed by t, units in years. Variables with this index are allowed to vary by year.
- Insurance decision is indexed by i, where $i = 1$ corresponds to the purchase of insurance and $i = 0$
- Revenues and costs are indexed by d, the presence of drought and the drought adaptation measure where:
  - base (base costs),
  - nd (no drought, equivalent to base),
  - noadpt (drought, no adaptation),
  - feed (drought, buy feed),
  - rent (drought, rent pasture),
  - sell (drought, sell and replace)

3 Data

- Grid base data (what is this?): `grid_base.RData`, called in `load.R` script
- Insurance base data (what is this?): `insurance_base.RData`, called in `load.R` script
- NOAA Index (be more specific): `noaaIndex.RData`, called in `load.R` script
- Coops (what is this?): `coops.RData`, called in `load.R` script
- MLRA zone data (be more specific): `mlra=readOGR("data","mlra_v42")` called in `load.R` script. *I don't know what this function means*

4 Precipitation and Forage

The model is currently set for a default to the Central Plans Experimental Range (CPER), but alternative locations for COOP sites in Colorado may also be used. We will expand the model to include more states.

- Function: `getStationGauge`
  - Description: Returns precipitation record and locational attributes for the target location. Default is Central Plains Experimental Range (CPER) but alternative locations at COOP sites across Colorado may be specified.
  - Inputs: `target.loc` (target location. default set to `CPER`, but COOP sites in Colorado can also be specified.
  - Outputs: `zonewt` (Zone Weights, the weight given to precipitation in each month in terms of how much it contributes to annual forage), `stzone` (Weights Zone, a single numeric value that designates the MLRA zone), `stgg` (station or gauge precipitation data from 1948 to 2015), `tgrid` (Target Grid Cell for reading in PRF index values at a given point in time).
  - Upon completion of the function `getStationGauge`, a new sub-environment `station.gauge` is generated, which contains `zonewt`, `stzone`, `stgg`, and `tgrd` based on the target location.

If CPER (default):

1. Assign `stzone` a value of 3, corresponding to the zone where the CPER resides.
2. Zone Weights are read from the Excel model which is based on [xx] drought calculator state forage potential weights that we cannot reproduce. We are missing spatial reference information necessary to assign each target location to a state zone.
3. Station Gauge, historical precipitation totals dating back to 1948, which are also read in from the Excel model. Precip totals are collected at CPER itself and do not rely on precip data from COOP sites.
4. The target grid cell 25002 is assigned to the PRF grid cell (assuming this is the correct grid cell)

If custom location is specified:

1. Loads `data/coops.RData` which was generated by the `coop_scraper.R` script.
2. Zone Weights are based upon the Major Land Resource Area in which the COOP site resides. The MLRA forage potential is an average of plant growth curves calculated

for a series of ecological site surveys (ESS) performed for that MLRA (using functions `COOPinMRLA` and `getMLRAWeights`). Our decision to use an average plant growth curve is a placeholder that could be replaced, for example, by a regression framework like the one used to calculate the state weights. Alternatively, we could contact the authors of the [xx] state weights and use these instead of MLRAs (since these are likely more accurate than the weights used here).)

3. Weights Zone which in this case corresponds to the MLRA site within which the target COOP site resides.

4. Station Gauge, which is read from the historical precipitation data `precip` attached to the target COOP site list `target.coop`.) `tgrid` (Target Grid Cell which belongs to the list of COOP site attributes `target.coop`. This variable is computed by converting the coordinates of the COOP site to a SpatialPoint object and finding the underlying PRF grid cell.

- Function: foragePWt
  - Description: Returns a weight representing annual forage potential for a given gridcell or station gauge's annual precipitation record. By default, it computes the sum of weighted product of forage potential and long-term precipitation deviation from average for a given year relative to a grid cell or station gauge's period of record. To assemble the weights: It uses the product of deviation in precipitation from long-term (1948-2015) average and zone weights for months occurring before and during the decision month. For months occurring after the decision month, use the product of group average deviation from the long-term average and zone weight. This approach roughly approximates a 'best guess' scenario based on rain gauge observations - what should my precip for the remainder of the year look like given what I know by the decision month?
  - Inputs: `stgg` (station gauge or grid cell precip record), `stzone` (state zone), `zonewt` (weights for state zone), `styear` (year of interest), `decision` (use 'decision under uncertainty' mode (default FALSE))
  - Options: A 'decision making under uncertainty' mode is also available (when 'decision' is set to TRUE). To build monthly weights, it generates a typology of years 1948-2015 (k-medoids) based on monthly precipitation values observed at the station.
  - Limitations: Need to build in inputs for state.

1. Pull zone weights from `zonewt` for station/grid of interest, `stzone`

2. Pull monthly precipitation totals for starting year from `stgg`

3. Create a monthly precipitation index by dividing monthly precipitation for the year and station/grid of interest by monthly average precipitation
4. Under default mode:
    - Create monthly forage potential weights by multiplying zone weights for each month by the monthly precipitation index, then sum for an annual forage potential.
5. Under decision-making under uncertainty mode:
    - Return to this later or talk to Joe about this functionality.

- Function: `getMRLAWeights`
  - Description: Computes forage potential weights using the mean of plant growth curves by MRLA for a specified state.
  - Data Source: https://esis.sc.egov.usda.gov/WelcomeFSG/pgFSGSelectFormat.aspx
  - Inputs: `state.code` (two-letter state code for referencing plant growth potential curves)
  - Outputs: `forage_mlra` (Monthly precipitation forage weights)
- Function: `COOP_in_MRLA`
  - Description: Returns the MLRA in which a specified coop site is located.
  - Inputs: `coop`
  - Output: MLRA (this needs further clarification)
- Function: `forageWeights2Intervals`
  - Description: Helper function for binning monthly forage weights into 2-month intervals matching the RMA insurance.
  - Inputs: `fpwt`
  - Outputs: `fpwt_iv`

5 Profit Model

$$\pi_t = R_{total,d,i,t} - C_{total,d,i,t} \tag{1}$$

5.1 Revenues

Total Revenue Function:

$$R_{total,d,i,t} = R_{d,t} + indem_{i,t} \tag{2}$$

Base sales:

- Function: `CalculateExpSales`
- Inputs: $h =$ `herd` (herd size (number of cows, does not include calves)), $\phi =$ `calf.sell` (average percentage of calves sold), $\bar{w} =$ `expected.wn.wt` (average calf weight at weaning under normal conditions (pounds)), $p_t =$ `p.wn`(price of calves at weaning in year t ($/pound))

$$R_{base,t} = \phi h * \bar{w} * p_t \tag{3}$$

5.2 Costs

Total Cost Function:

$$C_{total,t} = C_{d,t} + iprem_{i,t} \tag{4}$$

5.2.1 Base costs

- Function: `CalculateBaseOpCosts`
  - Inputs: h `herd` (number of cows, does not include calves), $\gamma =$ `cow.cost` (base operating costs ($/cow))

$$C_{base,t} = \gamma h \tag{5}$$

6 Drought Adaptation

Drought Adaptation Options:

1. Do Nothing
2. Buy Feed
3. Rent Pasture
4. Sell Pairs and Replace

How much adaptation is needed? Depends on the length of adaptation action (λ = `days.act`) and the intensity of the drought (α = `forage.potential`). Different drought adaptation actions are scaled slightly differently to account for fixed costs (e.g., trucking) and variables costs (e.g., days of pasture rental).

- Function: `CalculateDaysAction`
  - Description: Calculate the number of days rancher pays for a drought adaptation action.
  - Inputs: `act.st.yr` (year the action starts), `act.st.m` (month the action starts), `act.end.yr` (year the action ends), `act.end.m` (month the action ends)
  - Outputs: Number of days drought adaptation action takes place (days) (`days.act`)
  - Assumptions: Assumes that the actions take place only in one year.
  - Limitation: Not equipped to handle drought adaptation in multiple years. Currently only works for the first year (bug identified).

$$\lambda = 30 * (\texttt{act.end.m} - \texttt{act.end.m}) \tag{6}$$

- Function: `CalculateAdaptationIntensity`
  - Description: Takes forage potential and an adaptation intensity factor to provide a scalar of drought action. If forage potential is above 1 (no drought), then this variable goes to 0 (no adaptation).
  - Inputs: ψ = `adpt.intensity.factor` (parameter that scales adaptation actions to reflect actual adaptation behavior. Currently defaults to 1 which assumes a one-to-one ratio of drops in forage percentage to need for forage replacement.), α = `forage.potential` (the percentage of average forage produced in a year based on rainfall. See forage potential functions.)
  - Output: `drght.act.adj` (scales action to account for forage potential's deviation from the norm.)
  - Assumptions: The variable has a maximum of 1, which assumes that drought actions are parameterized at full forage replacement for the full herd.

$$\beta = \texttt{drght.act.adj} = \begin{cases} \min\{1, (1-\alpha) * \psi)\} & \text{if } \alpha \leq 0 \\ 0 & \text{else.} \end{cases} \tag{7}$$

6.1 Drought Adaptation Option: Do Nothing

If ranchers do nothing when forage potential drops below 1, then calves do not gain as much weight and cows produce fewer calves.

6.1.1 Costs

Costs are unchanged from base operating costs (eq 5):

$$C_{noadpt,t} = C_{base,t} \tag{8}$$

6.1.2 Revenues

When drought occurs and no adaption is undertaken, revenues are affected by calf sales through both reduced weaning weights and lower weaning success.

- Function: `calfWeanWeight`
  - Description: Compute calf weights based on station/grid cell forage potential for a five-year period. Wean weights are computed for each of the five years as a summed product of the target location's forage potential weights and precipitation index by interval.
  - Utilizes `foragePWt` and `calfDroughtWeight` functions.
  - Inputs: y_0 = `styr` (starting year of the five-year period)
  - Outputs: These are returned as a matrix of calf weights by year, `calf_weights_ann`.

1. Get forage weights for the years and zone with `foragePWt` function.
2. Calculate calf drought weight according to eq 9 with `calfDroughtWeight` function.

- Function: `calfDroughtWeight`
  - Description:
  - Inputs: $\bar{w}$ = `expected.wn.wt` (average calf weight at weaning under normal conditions (pounds)), $w_{calf,0}$ = `calf.wt` ('current' weight of calves at decision point), α = `forage.potential` (annual forage potential weight for zone)
  - Output: $w_{noadpt,t}$ = `wn.wt` (calf weight at weaning under no drought adaptation in year t)

$$w_{noadpt,t} = w_{calf,0} + \alpha(\bar{w} - w_{calf,0}) \tag{9}$$

6.2 Drought Adaptation Option: Buy Feed

6.2.1 Costs

In addition to base costs, $C_{base,t}$, with the buy feed adaptation option, we add the cost of buying feed according to the following function:

- Function: `CalculateFeedCost`

- Description: Calculating the costs of purchasing additional feed
- Inputs: $ration_{ray}$ = `hay.ration` (hay ration assuming no grazing (pounds/head/day) **Source needed**), p_{hay} = `p.hay` (price of hay ($/ton). user input), $ration_{oth}$ = `oth.ration` (ration of non-hay feed (pounds/head/day) **Source needed**, p_{oth} = `p.oth` (price of other feed ($/ton). User input. Does not come into play since the model assumes only feeding hay), β = `intens.adj` (drought intensity adjustment, eq. 7), λ = `days.act` (days adaptation action (days), eq. 6), h = `herd` (size of herd (head of cows, does not include calves))
- Outputs: `cost.feed` (additional costs to feed the herd over the remainder of the season ($/year))

$$C_{feed,t} = \beta\lambda h \left(\frac{ration_{hay}}{2000} * p_{hay} + \frac{ration_{oth}}{2000} * p_{oth} \right) \tag{10}$$

6.2.2 Revenues

Revenues are unchanged from the base level. (eq. 3)

$$R_{feed,t} = R_{base,t} \tag{11}$$

6.3 Drought Adaptation: Rent Pasture

6.3.1 Costs

In addition to base costs, we add the cost of renting pasture according to the following function:

- Function: `CalculatePastureRentCost`
  - Description: Calculates the costs of renting pasture and trucking pairs
  - Inputs: m = `n.miles` (distance to rented pasture (miles)), p_{truck} = `truck.cost` (trucking cost per loaded mile ($/mile/truck)), p_{rent} = `past.rent` (price of renting pasture per animal unit month, where an animal unit is a cow/calf pair ($/pair/month)), β = `intens.adj` (portion of herd moving to rented pasture), λ = `days.act` (days on rented pasture(days), C_{fixed} = `oth.cost` (all other non-rental, non-trucking costs ($)), w_{max} = `max.wt` (maximum weight per truck (pounds)), w_cow = `cow.wt` (average cow weight (pounds)), w_{calf} = `calf.wt` (average 'current' weight of calves before trucking to rented pasture (pounds)), h =`herd` (size of herd (head of cows, does not include calves))
  - Output: `cost.rentpast` (total costs of using renting pasture including transport costs on top of normal operating costs ($/year))

– Assumptions: Only cows are trucked back home. Fixed costs cover transaction costs (time, etc.) of arranging pasture rental.

Number of trucks needed to transport portion of herd (pairs) to rented pasture:

$$n_{to} = \lceil \beta h * \lceil w_{max}/(w_{cow} + w_{calf}) \rceil \rceil \tag{12}$$

Number of trucks needed to transport portion of herd (cows only) back to home pasture:

$$n_{from} = \lceil \beta h * \lceil w_{max}/w_{cow} \rceil \rceil \tag{13}$$

Cost of hiring trucks:

$$C_{trucks} = m * p_{truck}(n_{to} + n_{from}) \tag{14}$$

Cost of renting pasture:

$$C_{past} = \beta \lambda h \frac{p_{rent}}{30} \tag{15}$$

Total cost of ranching operation with drought adaptation through rental pasture:

$$C_{rent,t} = C_{trucks} + C_{past} + C_{fixed} + C_{base} \tag{16}$$

6.3.2 Revenues

Due to losses during the stress of trucking cows and calves, the revenues are lower than normal.

- Function: `CalculateRentPastRevenue`
  - Description: Calculates calf sale revenues after trucking pairs to rented pastures
  - Inputs: `calf.loss` (additional calf deaths due to transport stress (head of calves)), `calf.wt.adj` (adjustment for calf weaning weights (%)), `calf.sell` (average percentage of calves sold (%)), w_t = `wn.wt` (average weight at weaning (pounds)), p_t = `p.wn` (expected sale price of calves ($/pound)), `herd` (size of herd (head of cows, does not include calves)), β = `intens.adj` (portion of herd moving to rented pasture)
  - Outputs: `rev.rentpast` (change in revenue due to mortality and weight loss from trucking to rented pasture)

Number of calves sold after accounting for calf mortality in transport:

$$calves_{rent} = h * \beta * \texttt{calf.sell} - \texttt{calf.loss} \tag{17}$$

$$calves_{home} = h * (1 - \beta) * \texttt{calf.sell} \tag{18}$$

Selling weight after accounting for weight loss due to transport stress

$$wt_{rent} < -wt_{normal} * (1 + \texttt{calf.wt.adj}) \tag{19}$$

Expected calf sale revenues

$$R_{rent} = p_{wn}(calves_{rent} * wt_{rent} + calves_{home} * wt_{normal}) \tag{20}$$

6.4 Drought Adaptation: Sell and Replace

6.4.1 Costs

When a rancher sells the herd, the operating costs of the ranch drops in the year the herd is sold, includes only basic fixed operating costs when there is no herd, and then goes back to normal base operating costs.

- Function: CalculateSellPrsCost
  - Description: Calculates the operating costs to sell pairs in year 1 and replacing cows in year 3
  - Inputs: `op.cost.adj` (change in operating costs in year 1 per cow after selling herd ($/cow/year)), `sell.cost` (selling cost per cow ($/cow)), $h =$ `herd` (size of herd (head of cows, does not include calves)), `base.cost` (baseline annual cost of operating ranch with full herd ($/year)), `fixed.op.cost` (fixed operating costs for a year without a herd ($/year))
  - Outputs: `cost.sellprs` (5x1 vector of changes in operating costs for years 1 through 5 from selling pairs in year 1 and replacing them at the end of year 3)
  - Assumptions:
    * It is assumed that cows are replaced on last day of the second year after they are sold. For example, cows sold in 2011 are replaced on 12/31/2013.
    * The adjustment in operating costs does not depend on when the herd is sold.
    * Selling costs are additional to normal selling costs.
    * No additional purchasing costs are added when the herd is restocked.
    * The herd size is the same before and after selling the herd.
    * Entire herd is sold.

$$
\begin{aligned}
C_{sell,1} &= C_{base} + h(op.cost.adj + sell.cost) \\
C_{sell,2} &= C_{herdless} \\
C_{sell,3} &= C_{herdless} \\
C_{sell,4} &= C_{base} \\
C_{sell,5} &= C_{base}
\end{aligned}
$$

6.4.2 Revenues

- Function: CalculateSellPrsRev

- Description: Calculates calf sales revenues due to selling pairs and replacing cows for years 1 through 3
- Inputs: `base.sales` (calf sales in a normal year ($/year)), $p_{calf,0}$ = `p.wn.t0` (current sale price calves ($/pound), h = `herd` (size of herd (head of cows, does not include calves)), `wn.succ` (average percentage of cows that successfully wean calves (%)), $w_{calf,0}$ = `calf.wt` (average 'current' weight of calves (pounds))
- Outputs: `rev.sellprs` (5x1 vector of calf revenues for years 1 through 5)
- Assumptions: It is assumed that cows are replaced on last day of the second year after they are sold. For example, cows sold in 2011 are replaced on 12/31/2013. The herd size is the same before and after selling the herd. Entire herd is sold.

$$
\begin{aligned}
R_{sell,1} &= h * wn.succ * w_{calf,0} * p_{calf,0} \\
R_{sell,2} &= 0 \\
R_{sell,3} &= 0 \\
R_{sell,4} &= R_{base,4} \\
R_{sell,5} &= R_{base,5}
\end{aligned}
$$

7 Assets and Net Worth

Calculating assets and net worth.

- Function: `CalcCowAssets`
  - Description: Calculates the cow assets for each year.
  - Inputs: `herd`, p_{cow} = `p.cow`, `sell.year` (year the herd is sold. single numeric value. default is year 1.), `replace.year` (year the herd is replaced. single numeric value. default is year 3).
  - Output: A_{cow} = `cow.assets` (6x1 vector of cow assets for each year, including t=0)

  If herd is never sold:

$$A_{cow,t} = h * p_{cow} \tag{21}$$

  If herd is sold and replaced:

$$A_{cow,t} = \begin{cases} h * p_{cow}, & \text{if } t < t_{sell}, t > t_{replace}, \text{or } t_{sell} = \varnothing \\ 0, & \text{if } t_{sell} \leq t < t_{replace} \end{cases} \tag{22}$$

- Function: `CalcCapSalesPurch`

- Description: Calculates vectors of capital sales and capital purchases from changes in cow assets and normal culling rates.
- Inputs: `assets.cow` (tx1 vector of the value of cow assets each year)
- Outputs: S_t = `cap.sales` (tx1 vector of capital sales for each year), P_t = `cap.purch` (tx1 vector of capital purchases for each year)
- Assumptions: Assumes sale/purchase of cows is only capital sales/purchase. Assumes normal culling is counted as a capital sales (*Is this correct?*.

$$S_t = \begin{cases} A_{t-1} - A_t, & \text{if } A_t < A_{t-1} \\ n_{cull} * p_{cow}, & \text{if } A_t \geq A_{t-1} \text{ and } A_{t-1} \neq 0 \\ 0, & \text{if } A_t \geq A_{t-1} \text{ and } A_{t-1} = 0 \end{cases} \tag{23}$$

$$P_t = \begin{cases} A_t - A_{t-1}, & \text{if } A_t > A_{t-1} \\ 0, & \text{if } A_t \leq A_{t-1} \end{cases} \tag{24}$$

- Function: `CalcCapTaxes`
  - Description: Calculates capital taxes on herd sales. Tax treatment is different depending on whether herd is sold and replaced by the end of the third year or if the herd is sold and not replaced during a drought emergency.
  - Inputs: `cap.sales`, `cap.purch`, r_{cap} = `cap.tax.rate`, `drought.emrg` (binary variable to indicate whether drought emergency was in place when the herd was sold currently set to a default of 1. This only matters if the herd is sold and not replaced.)
  - Outputs: τ = `cap.taxes` (5x1 vector of capital taxes)
  - Assumptions: Assumes that the entire herd is sold and replaced at the same rate. Not sure how the tax code treats changes in prices. This abstracts away from that. The price dynamics could matter here, but for now we are leaving them out.

Without a federally declared drought disaster:

$$\tau_t = \begin{cases} 0, & \text{if cows are replaced within two years} \\ S_t * r_{tax}, & \text{if cows are not replaced within two years} \end{cases} \tag{25}$$

With a federally declared drought disaster, capital taxes are delayed one year:

$$\tau_t = 0 \tag{26}$$

$$\tau_{t+1} = S_t * r_{tax} \tag{27}$$

8 Insurance Model

- Function: `dcInfo`
  - Description: Extracts drought calculator information from a grid cell
  - Inputs: `dc` (drought calculator output), `tgrd` (target grid cell id)
  - Outputs: `dcinf` (be more specific)
- Function: `droughtCalculator`
  - Description: Emulates RMA's precipitation-based insurance index in raster. NOTE that premium/indemnity estimates will be slightly off those of RMA because our index values 'intervalNOAA' slightly disagree.
  - Inputs: `yy` (year of interest), `clv` (RMA coverage level. Accepted values are 0.7, 0.75, 0.8, 0.85, 0.9), `acres` (insured acres), `pfactor` (productivity factor of grazing land), `insPurchase` (a m? x n? matrix of intervals from 1-11 for which insurance is purchased. For example, purchases for the April-May and May-June intervals at 50% protection each would be entered as 'rbind(c(3,0.5),c(5,0.5))' Consecutive intervals are not allowed.)
  - Outputs (list): `prem_noSbdy` (total premium with subsidy), `prem_wSbdy` (total premium without subsidy), `prodPrem` (premium paid by producer), `indemrate` (indemnity rate (stack, by month)), `indemnity` (indemnity (stack, by month)), `indemtot` (total indemnity)
  - Requirements: Insurance allocation for consecutive intervals is not permitted. Insurance must be allocated for at least two intervals. Insurance allocation intervals must range from 1-11. Insurance allocation may not exceed 60% per interval. Insurance allocation must sum to 100%.

1. Get appropriate subsidy rate based on coverage level from `covsub` matrix: Coverage Subsidies (`covsub`):

Coverage Level	Subsidy Rate
70%	59%
75%	59%
80%	55%
85%	55%
90%	51%

$$sbdy = \begin{cases} 0.59, & \text{if } \texttt{clv} \leq 0.75 \\ 0.55, & \text{if } 0.75 < \texttt{clv} < 0.90 \\ 0.51, & \text{if } \texttt{clv} \geq 0.90 \end{cases} \tag{28}$$

2. Put insurance purchase values for each interval (`insPurchase`) into an insurance purchase vector (1x11).

3. Calculate the policy rate:

$$\texttt{plrt} = \texttt{clv} * \texttt{acres} * \texttt{pfactor} * \texttt{basePrice} \tag{29}$$

   - #### Where does `basePrice` and `covsub` come from in the code? What is the policy rate?

4. Generate inputs to compute premiums

5. Compute premiums

6. Round premiums to match RMA (as closely as possible)

7. Compute indemnities

8. Prepare outputs into `outList`

- Function: `insMat`
  - Description: Generates a matrix representing insurance premium payments and indemnities for a specified grid cell over a five-year interval.
  - Inputs: `tgrd` (target grid cell), `yyr` (starting year), `clv` (coverage level), `acres` (insured acres), `pfactor` (land productivity factor), `insPurchase` (a matrix representing insurance allocation to two-month intervals, with rows written in the format [mm,amt])
  - Outputs: a 5 x n? matrix with insurance premium payments (column ?) and indemnities (column ?) for a specified grid cell over a five-year interval.

Insurance Purchase (`insPurchase`, `insp`):

Default:Excel model defaults $\texttt{ins} = \begin{bmatrix} 3 & 0.5 \\ 5 & 0.5 \end{bmatrix}$

Option "autoSelect.insurance":

- Function: `insAlloc`
  - Description: Automates range insurance allocation to two-month RMA intervals using a grid cell/COOP site's forage potential weights. Returns a matrix formatted as the 'insPurchase' input for function 'insMat'. Allocation for chosen two-month intervals is roughly proportional to the relative value of each interval's forage potential weight. Adjustments to allocation percentages are automatically made if a selection is invalid for one or more intervals, either too high (>60%) or too low (10%). User-specified min/max allocation percentages falling within this range may also be substituted by setting the 'max.alloc' and 'min.alloc' arguments.

- Inputs: `fpwt` (A vector of monthly forage potential weights for the target site. Monthly intervals are averaged to two-month intervals to match RMA insurance selections.), `niv` (number of two-month intervals to insure), `by.rank` (if TRUE (default), ranks forage potential weights by interval in descending order and selects the 'niv' most highly ranked non-consecutive intervals to insure. If FALSE, selects the combination of 'niv' non-consecutive two-month intervals with the highest average forage potential weights.), `max.alloc` (maximum interval allocation, 0.6), `min.alloc` (minimum interval allocation, 0.1).
- Outputs:

9 Additional Functions

- Function: `gridToRaster`
  - Description: Simple wrapper to turn a matrix into a raster using a template
  - Inputs: `grid` (data in matrix format), `rasterTemplate` (template to use in rasterization of the matrix)
  - Outputs: Raster of grid data
- Function: `dataToRast`
  - Description: Convert a data.frame/data.table field to raster.
  - Inputs: `inData` (input data frame/table), `target.var` (character representing fields to map to raster grid, default set to NULL)
  - Outputs: `dataRast`
- Function: `:=`
  - Description: Magical function that allows you to return more than one variable from other functions. Code from http://stackoverflow.com/questions/1826519/function-returning-more-than-one-value

10 Simulation Run

Function: `sim_run`

1. Load `insurance_base.RData`
2. Calculate base sales and operating costs with average weaning weights (Functions: `CalculateExpSales`, `CalculateBaseOpCosts`)
3. Compute insurance premiums and indemnities (Functions: `insMat`)

4. Calculate base cow assets and capital taxes (Functions: `CalcCowAssets, CalcCapSalePurch, CalcCapT`
5. Calculate outputs without a drought (Function: `OptionOutput`)
6. Calculate forage potential vector (Function: `foragePWt`)
7. Calculate drought action adjustment (Function: `CalculateAdaptationIntensity`)
8. Calculate days of action (Function: `CalculateDaysActions`)
9. Calculate expected sales revenues without adaptation, thus weaning weight is affected by forage potential (Function: `CalculateExpSales`)

11 Limitations

Currently not equipped to handle multi-year droughts. This can be changed.