Mathematical representation of the drought decision model - Shiny Version

Trisha Shrum

June 29, 2017

1 Scripts

1.1 global.R

1. Sources other scripts
2. Javascript coding
3. Populate a new environment with rainfall gauge info: `getStationGauge()`
4. Populate a new environment with constant (user) variables: `getConstantVars()`
5. Setting additional variables: acres, start years, simulation lengths
6. Create state variables for practice and full runs: `getSimVars()`
7. Create lists of variables for practice and full runs: `practiceRuns`, `simRuns`
8. Establish additional settings

1.2 load.R

Loads necessary packages

1.3 shinySupportFunctions.R

1. `getJulyInfo` function: Calculates available and predicted forage in July, creates a UI to display info and allows user to select adaptation level.
    - Called in `simUI.R`
2. `getCowSell` function: Creates a UI for the user to select how many cows and calves to sell. Called in `simUI.R`.

3. `shinyInsurance` function: Calculates premium and indemnification for a specific year and grid cell. Currently returns are summed but this could be done on a index interval basis instead.

1.4 forageFunctions.R

- `getForagePotential` function: Returns an index representing annual forage production for a given gridcell or station gauge's annual precipitation record. Called in `calfCowFunctions.R`.
- `whatIfForage` function: calculates expected forage for a given scenario. Called in `shinySupportFunctions.R` and `simUI.R`.
- `getMLRAWeights` function: Computes forage potential weights using the mean of plant growth curves by MRLA for a specified state. Called in `initialFunctions.R`.
- `COOP_in_MLRA` function: Returns the MLRA in which a specified coop site is located. Called in `initialFunctions.R`.

1.5 adaptationFunctions.R

- `calculateAdaptationIntensity` function: 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). Called in `shinySupportFunctions.R` and `simUI.R`.

1.6 costRevenueFunctions.R

- `calculateExpSales` function: Calculates expected calf revenues for non-drought year.
- `calculateFeedCost` function: Calculates the costs of purchasing additional feed. Called in `getAdaptCost` in `costRevenueFunctions.R`.
- `CalculateRentPastCost` function: Calculates the costs of renting pasture and trucking pairs. Called in `getAdaptCost` in `costRevenueFunctions.R`.
- `getAdaptCost` function: Calculates the cost of adaptation based on strategy, intensity needed, days, and herd size. Called in `shinySupportFunctions.R` and `simUI.R`.

1.7 initialFunctions.R

- `getConstantVars` function: Reads in constant variables into a `constvars` environment using the file `data/constant_vars.csv`. Called in `global.R`.

- `getSimVars` function: Creates list of simulation variables. Called in `global.R`.
- `getStationGauge` function: 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. Called in `global.R`.
- `createResultsFrame` function: This function creates a theoretical previous result from the year before the simulation begins right now this assumes that there was no drought the year before the simulation and revenues were 0. These assumptions are likely unrealistic and can be adjusted to accomodate different scenarios. Called in `shinySupportFunctions.R` and `server.R`.

1.8 calfCowFunctions.R

- `AdjWeanSuccess` function: Adusts weaning success downward for the year of the drought and the following year. Called in `simUI.R`.
- `calfDroughtWeight` function: If forage potential is less than 1, then the calf weight is less than the optimal weight. Called in `shinySupportFunctions.R` and `simUI.R`.
- `calfWeanWeight` function: Computes calf weights based on station/grid cell forage potential for a n-year period. Called in `initialFunctions.R`.
- `shinyHerd` function: calculates the size of herd for the shiny app. Called in `simUI.R`.

1.9 assetFunctions.R

- `CalcCowAssets` function: Calculates the cow assets for each year. Called in `initialFunctions`.

2 Function Details

2.1 AdjWeanSuccess

2.2 Current State

- Function: `AdjWeanSuccess`
  - Description: Adjusts weaning success downward for the year of the drought and the following year based on a modified logistic equation.
  - Inputs: `stgg`, `zonewt`, `stzone`, `styear`, `noadpt`, `normal.wn.succ`, `t`
  - Output: `wn.succ` (tx1 vector of weaning success in percentage of cows that will have calves that survive to be fully weaned)

– Assumptions: This equation is based on what I consider to be "reasonable" estimates of weaning success based on forage potential. These fall roughly in line with body condition scores from the *Nutrient Requirements of Beef Cattle*, but are only ballpark estimates.

If drought adaptation is not undertaken and $\alpha < 1$:

$$wn_1 = \bar{wn} * \frac{1}{1 + e^{2(-1+\alpha_1)}} \tag{1}$$

$$wn_2 = \bar{wn} * \frac{1}{1 + e^{(-1+\alpha_1)}} \tag{2}$$

$$wn_3 = \bar{wn} \tag{3}$$

$$wn_4 = \bar{wn} \tag{4}$$

$$wn_5 = \bar{wn} \tag{5}$$

$$\tag{6}$$

Otherwise:

$$wn_t = \bar{wn} \tag{7}$$

2.2.1 Desired Future State

The weaning percentage default and maximum is 88%.

If forage production falls below 1 in year $t = 1$, then weaning percentage falls slightly in year $t = 1$ and more drastically in year $t = 2$. If forage production falls below 1 in a year $t = 1$ where weaning percentage was already decremented because of previous forage production deficits or insufficient culling, then weaning percentage falls further in years $t = 1, 2$ than it would have if the starting point was at the maximum weaning percentage.

Where `wn.succ` is the weaning success going into a given year (say, t = 5) and `forage.production` < 1: for year t=5, the final weaning success for the full year is given by:

```
wn.succ <- wn.succ * (1 / (1 + exp(-(1 + forage.production_t5)*2)))
```

For the year t=5, if the `forage.production_t6` ≥ 1:

```
wn.succ <- wn.succ * (1 / (1 + exp(-(1 + forage.production_t5))))
```

If forage production is below 1 for more than year in a row, then the wn.succ is decremented twice. Once with the first year decrement, and once with the second year decrement.

year 1 and 2 have low forage, but recovers to 1 or more in year 3: `wn.succ_2 <-  wn.succ * (1 / (1 + ex`
`wn.succ_3 <-  wn.succ * (1 / (1 + exp(-(1 + forage.production_2))))`