Robotic Milking Systems

1. Concept: This application is developed as a decision-support tool for dairy producers in assessing their investment decisions for robotic milking systems (AMS). The primary motivation for this app is the increasing labor shortage in the dairy industry and the advancement of AMS.

2. Goals:

* Provide an assessment for investing in AMS and other milking system upgrades (e.g., retrofit parlors and new parlors), given the user-provided data on his/her farm and assumptions
* Communicate with the user on key concepts in this decision-making such as Annualized Recovery Cost of Capital, weighted average cost of capital (WACC), Annualized After-tax Net Impact, and tax implication of the investment
* Increase the user’s understanding of the economics of AMS for strategic investment decision on his/her farm
* Engage the user through interactive design and multi-faceted presentation conducive to experimentation and learning

3. Design:

A. User-Interface:

Components:

The UI consists of the following tabs.

* Introduction: introduces the application to the user for its purpose, usage, and design
* Data Entry: asks for the user-provided data entry and presents a dashboard of results
* Partial Budget: presents a partial budget analysis
* Cash Flow: presents a cash flow analysis
* Summary: presents a summary of the results from three investment profiles (Robots, Retrofit Parlors, and New Parlors)
* Sensitivity: presents a sensitivity analysis for 10 key variables
* About: shows credits and contact information etc.

Interactions:

The user is first asked to select a set of default user-input values from several “Cases” that describe the underlying farm type for production practices and facilities in ways that help predict how AMS and other milking systems would change the operation for the farm. Then, the user skims through the default values while making changes as needed. The emphasis is at first on familiarizing himself/herself on the input variables and getting some sense of the analysis the application provides.

The main analysis is presented in three forms: Dashboard, Partial Budget, and Cash Flow. The Dashboard pane displays an “executive summary” showing the final impact calculation and its components and is designed to facilitate an interactive analysis for how various data inputs influence the results.

The Partial Budget pane displays item-by-item accounts of the changes in operation, followed by adjustment items that bridge the gap between a typical “snapshot” perspective of the partial budget analysis and a typical “down-the-road-forecast” perspective of the cash flow analysis. This pane is designed to help conceptualize how the investment changes his/her operational budget and how that compares to the finance cost of the long-term investment with certain tax implication.

The Cash Flow pane displays the year-to-year projection of the operating budget and after-tax cash flow along with key measurements such as WACC, Net Present Value (NPV), and return on investment (ROI). The final calculation result dubbed “Net (After-tax) Annual Impact”, which shows up in all three panes, is the annualized NPV at the discount rate of WACC (i.e., constant payments over the planning horizon). This pane is designed to help conceptualize the sequence of ups and downs in the operating budget and after-tax cash flow that are influenced primarily driven by inflation parameters, financing terms, and the tax deductions for interest payments and depreciation.

B. Server Design:

Components:

The major distinction in server-side files is: files used for the main calculation (“session\_calculation\_main.R”, “session\_calculation\_steps.R”, “session\_dashboard.R”, “session\_partial\_budget.R”, “session\_cash\_flow.R”), a file used for rendering outputs to UI (“session\_render\_base.R”), files used for additional features (“session\_summary.R”, “session\_sensitivity.R”, “session\_calculation\_delay.R”), and others (“session\_misc.R”, “session\_popover.R”).

Main calculation. File “session\_calculation\_main.R” calls “session\_calculation\_steps.R”, “session\_cash\_flow.R” and “session\_dashboard.R”, sequentially carrying out the main calculation. The functions in “session\_calculation\_main.R” react to any change in user input (i.e., list “input”). About half of the outputs in Partial Budget is calculated in this procedure, and the rest is calculated in “session\_partial\_budget.R”, which is invoked when the user views the Partial Budget pane.

Rendering. Most of the output (stored in list “output”) is processed in “session\_render\_base.R”. It handles some minor formatting and conditional adjustments. All rendering items are reactive and react to the change in input or other calculated reactive values.

Additional features. ‘’session\_summary.R’ simply sorts out the results from three base investment profiles in tables and charts and its functions are activated when the Summary tab is viewed. “session\_sensitivity.R” triggers the main calculation by substituting a value of one variable with an alternative value. When the user sets the year of investment in milking system greater than zero, “session\_calculation\_delay.R” inserts some additional variables describing the budget calculation prior to the beginning of that investment.

Others. “session\_misc.R” implements some axillary functions that update tabs (to synchronize a selected profile across the series of tabs in the Data Entry) and process the download and upload of data inputs. “session\_popover.R” displays popovers in UI using the variables and conditions from the server side.

4. Coding Notes

* For the three base profiles of Robots, Retrofit Parlors, and New Parlors, function calls are used to generate profile-specific UI components. On the serve-side, this is done using lapply().
* Reactive-value lists, “ans” and “sum”, are used to store calculation results. For each profile x, ans[[x]] contains slightly more than 100 calculated variables. There are 3 base profiles with each having 10 sensitivity simulations, implying that “ans” stores slightly over 3000 variables that are all reactive to user inputs.