ceRtainty: Certainty Equivalent in R

Ariel Soto-Caro
asotocaro@ufl.edu - arsoto@udec.cl
University of Florida & Universidad de Concepcion

June 14, 2019

1 Introduction

The Certainty Equivalent analysis for financial projects and, specifically agricultural treatments, is so far implemented by commercial software, to be installed as plugins into Microsoft Excel. The `ceRtainty` is a small package that allows performing this analysis through ***R***, which is free and where the data management procedures are incredibly more flexible and powerful than excel sheets.

The Stochastic Efficiency with respect to a Function (SERF) method presented by (Hardaker et al. 2004) is currently a widely used approach in the risk-efficient evaluation treatments. Mainly SERF offers a bounded estimation, allowing researchers or analysts to use any utility function to represent the decision-maker risk profile. Additionally, unlike other methods, the analyst doesn't need to know a priori decision-maker's risk aversion coefficient (Richardson and Outlaw 2008).

SERF methodology has been used to rank treatments either in controlled or in non-controlled experiments. The SERF requires only a small set of data and makes a minimum of assumptions. Specifically, the critical data is a time-series vector of profit values for each treatment and a utility function that represents a farmer's risk preference. With those elements, we can compute the Certainty Equivalence (CE) and the Risk Premium (RP) for every treatment.

The CE represents the sure amount for which a decision-maker remains indifferent between his/her current wealth and the risky outcome (Hardaker et al. 2004). The RP is the cost of the risk measure in terms of the production yield. SERF provides to decision-makers a very intuitive tool to make risk-efficient decisions by simply choosing the best-ranked treatment based on the CEs.

The `ceRtainty` package computes the CE values from a profit dataset and its respective risk premiums. Through this vignette, I will present how to use this package.

2 Instalation

The package is installed by the command:

```
> install.packages("ceRtainty")
```

When the package is already installed, should be loaded by:

```
> library(ceRtainty)
```

For a complete functionality, `ceRtainty` also requieres:

- **dplyr:** to organize the data.
- **tidyr:** to organize the data.
- **RColorBrewer:** to prepare the plots.

2.1 Functionality

At this time `ceRtainty` package is able to perform two measures:

- The Certainty Equivalent values (`certainty` function) for a set of profit's treatments or projects, and produces three objects: a table with CE values (`CE_values`), a vector with the risk aversion coefficients (RAC) values (`RAC`), and a plot with the CE values (`CE_plot()`).
- The Risk Premium values (`premium` function), using a data set of CE values and choosing a project or treatment as a base. This measure requires a `certainty` object to perform the computation. This function also creates three objects: a table of RP values [regarding an arbitrary treatment/project selected by the analyist] (`PremiumRisk`), a table with RP in terms of percentage of change regarding the base treatment/project (`PremiumRiskPer100`), and a plot of absolute values of the RP (`RP_plot()`).

Also is necessary to consider, even though CE can be computed for an extensive set of utility functions, the current version of `ceRtainty` package allows two of the most used functions (this list will increase in following versions):

- Power Utility function
- Exponential Negative Utility function.

2.2 Data set in ceRtainty

The `ceRtainty` package offers a unique (but small) dataset collected from agricultural experimental trials that were used and presented by Soto-Caro, Wu, and Guan (2019). This data set considers three pesticide treatments plus a non-treated (control) case, in strawberry fields, for seasons 2014-15 & 2015-16. Such experiments were conducted by the Gulf Coast Research and Education Center, University of Florida. Each value represents the farmer's profit, and there are four values for each season.

The data can be invoked in following way:

```
> data("profitSWG")
> head(profitSWG)
     control   fracture  milstop serenade
1   539.4105 4808.04725 6277.187 6625.230
2  9923.6830 3838.34400 8709.806 7738.285
3  4136.7481  984.85261 2495.159 4479.759
4  -464.1482   22.20683 3077.641 5823.031
5 12744.1448 7932.79606 1375.915 5425.849
6  3923.3339 6313.64550 1299.130 5325.025
```

3 Computing the CEs

The function to perform the certainty equivalent is `certainty()`. Is very important to be clear that, depending on the utility function, the RAC should be a relative or an absolute value:

$$r_a(w) = \frac{r_r(w)}{w}$$

Where w is the original or initial agent's wealth, r_a is the absolute RAC, and r_r is the relative RAC. Power Utility function uses relative RAC, and Exponential Negative function uses absolute RAC. This is selected into the `certainty` function automatically when the user defines the parameter `utility`. That is why the user can print the RAC values employed into the CE computation, as in the following example:

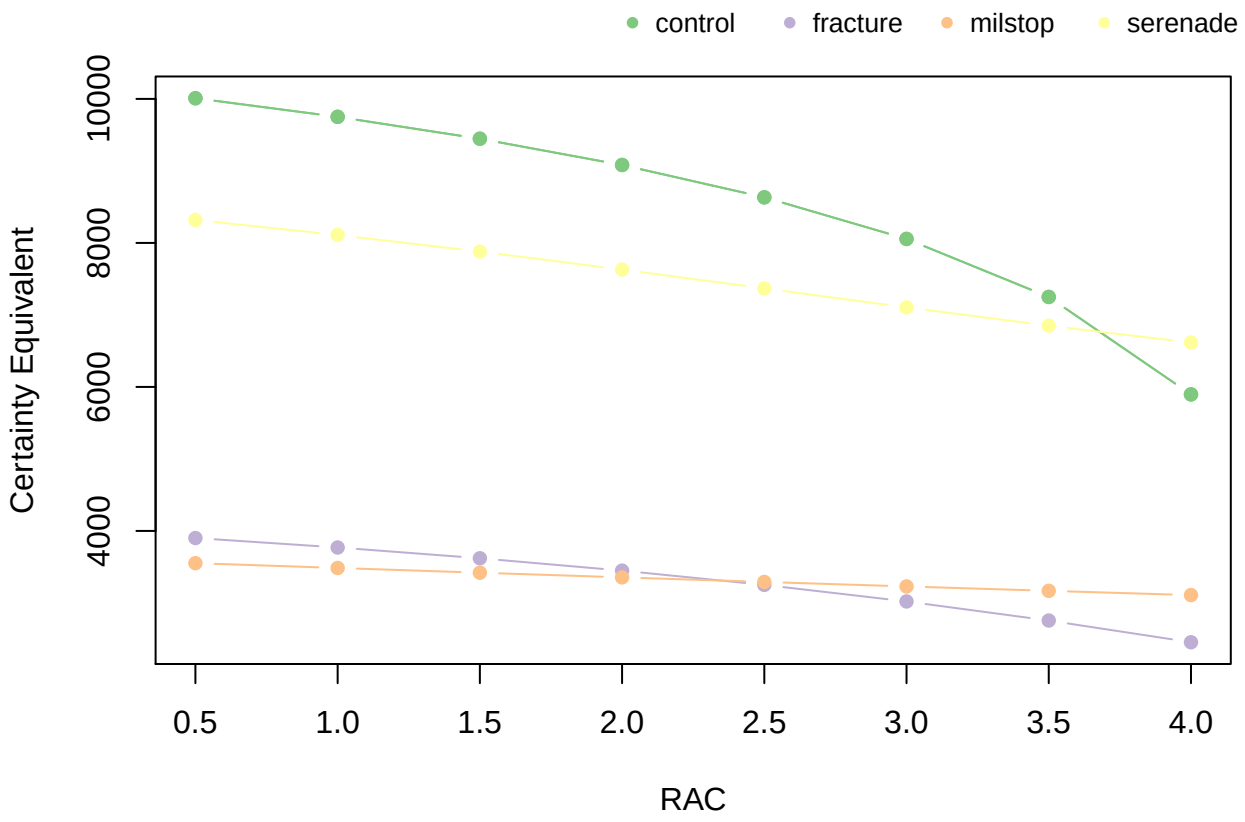
```r
library(ceRtainty)
data("profitSWG")
#
# Computing the CE values, for a RAC range of 0.5-4.0, and Power utility function.
#

# Obtaining the CE table
certainty(data = profitSWG, ival = 0.5, fval = 4, utility = "Power")$CE_values
```

```
##     control fracture  milstop serenade
## 1  5896.290 2454.139 3108.723 6616.198
## 2  7249.502 2755.638 3168.210 6850.728
## 3  8055.270 3020.697 3229.135 7104.565
## 4  8633.257 3250.468 3291.422 7368.019
## 5  9082.234 3448.827 3354.980 7629.706
## 6  9446.761 3620.500 3419.710 7879.504
## 7  9750.981 3770.055 3485.503 8110.506
## 8 10009.421 3901.479 3552.242 8319.343
```

```r
# Obtaining the RAC vector
certainty(data=profitSWG,ival=.5,fval=4,utility="Power")$RAC
```

```
## $racVector
##      control  fracture   milstop   serenade
## 1  0.0161051 0.9016105 0.9016105  0.0161051
## 2 -0.9677898 0.8032210 0.8032210 -0.9677898
## 3 -1.9516847 0.7048315 0.7048315 -1.9516847
## 4 -2.9355796 0.6064420 0.6064420 -2.9355796
## 5 -3.9194745 0.5080526 0.5080526 -3.9194745
## 6 -4.9033694 0.4096631 0.4096631 -4.9033694
## 7 -5.8872643 0.3112736 0.3112736 -5.8872643
## 8 -6.8711592 0.2128841 0.2128841 -6.8711592
##
## $rac_ini
## [1] 0.5
##
## $rac_fin
## [1] 4
##
## $rac_len
## [1] 8
```

```r
# Performing the CE plot
certainty(data=profitSWG,ival=.5,fval=4,utility="Power")$CE_plot()
```



The initial and final values of the absolute RAC vector are the parameters `ival` and `fval` respectively. The parameter `wealth` is 0 by default. The code computes one observation of the CE for each observation on the profit vectors. That means, when the profit vector has a length of n, then `certainty` compute a RAC vector with n values between `ival` and `fval`.

When the analysis considers too many projects or treatments simultaneously, the plot could be unclear and analyze the table instead is recommended. It is possible to perform this analysis with only one project, but with at least three observation per project.

4 Computing the RP

The risk premium is a measure to compare among CEs. In consequence, before to compute the RP is required to compute the CE values. Therefore, the procedure is: compute the CEs, stored them in an object, and then considered such an object as the dataset for the `premium` function. An example of this is presented below:

```
library(ceRtainty)
data("profitSWG")
#
# Computing and storing the CE values using Power utility function
#
ces <- certainty(data = profitSWG, ival = 0.5, fval = 4, utility = "Power")

ces_values  <- ces$CE_values  # store CE table
ces_rac     <- ces$RAC        # store RAC vector
```

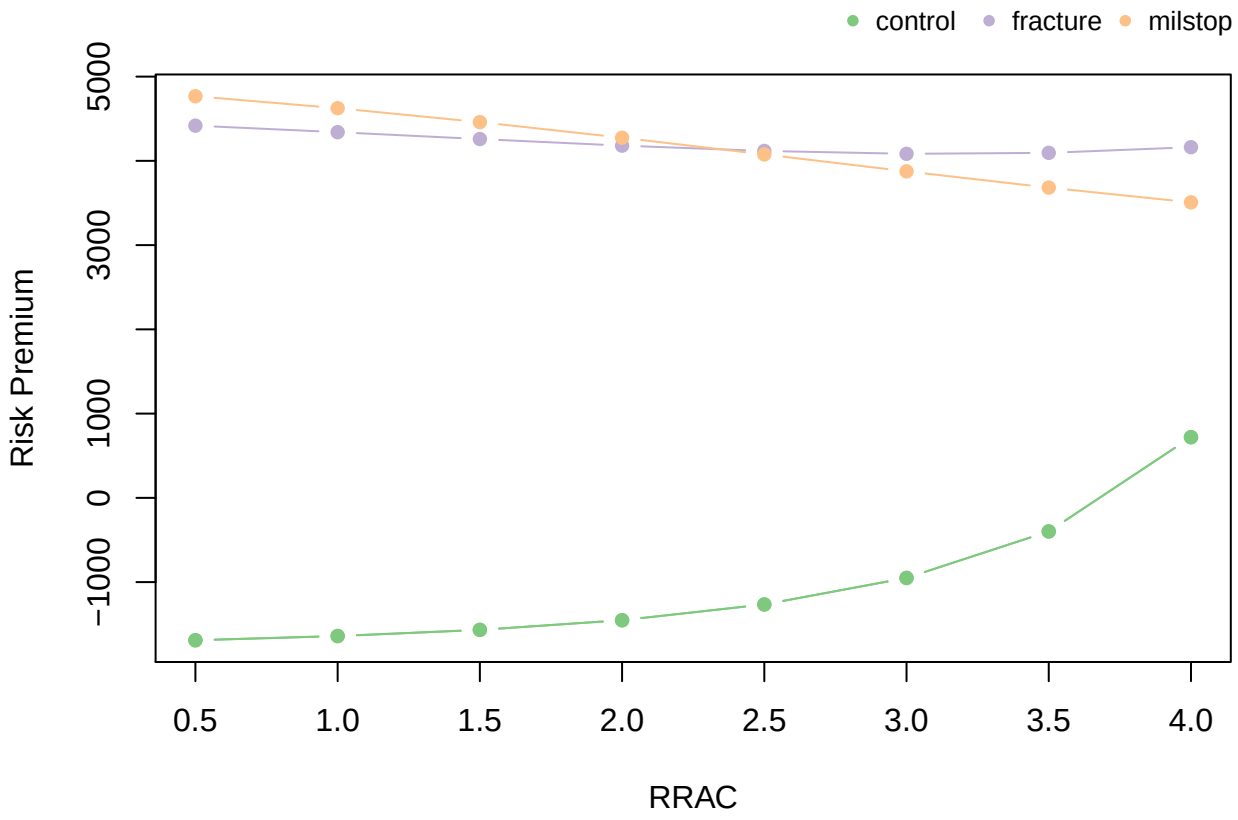
```r
# Computing the RP values respect to SERENADE treatment
premium(tbase = "serenade",ce_data = ces_values, rac = ces_rac, utility = "Power")$PremiumRisk
```

```
##      control fracture  milstop
## 1   719.9080 4162.059 3507.475
## 2  -398.7736 4095.090 3682.518
## 3  -950.7043 4083.869 3875.430
## 4 -1265.2373 4117.552 4076.598
## 5 -1452.5287 4180.878 4274.726
## 6 -1567.2573 4259.004 4459.794
## 7 -1640.4758 4340.451 4625.003
## 8 -1690.0783 4417.864 4767.101
```

```r
# Computing the RP values in percentage respect to SERENADE treatment
premium(tbase = "serenade",ce_data = ces_values, rac = ces_rac, utility = "Power")$PremiumRiskPer100
```

```
##      control  fracture   milstop
## 1 -12.209508 -169.5935 -112.8269
## 2   5.500704 -148.6077 -116.2334
## 3  11.802266 -135.1962 -120.0145
## 4  14.655388 -126.6757 -123.8552
## 5  15.993077 -121.2261 -127.4144
## 6  16.590420 -117.6358 -130.4144
## 7  16.823700 -115.1296 -132.6926
## 8  16.884876 -113.2356 -134.1998
```

```r
# Plotting the RP absolute values
premium(tbase = "serenade",ce_data = ces_values, rac = ces_rac, utility = "Power")$RP_plot()
```



The user must incorporate the utility function manually, and have to be the same employed in the CE computation.

5 Generating Risk Aversion Coefficients

According to Hardaker et al. (2004) the Power utility function, which uses relative RAC, must adjust the RAC regarding the number of digits in the profit, following the formula:

$$RAC_a = 1 - RAC^{d(|int(max(\pi))|-1)}$$

Where RAC_a is the adjusted RAC, π is the profit vector, int is the integer operator, and d is operator to obtain the amount or number of digits for a certain number.

In this case, the user must consider the RAC to employ here have to be a relative RAC (RRAC). Is not necessary this adjustment for the absolute RAC (ARAC). When the user performs the CE computation with `certainty` function, the adjusted RAC is computed automatically. But, if the user only wants to calculate the RACa, can employ the function `rac_generator` through setting the initial and final value (`ini` and `fin` respectively) of the RAC vector, and the profit dataset:

```
library(ceRtainty)
data("profitSWG")

rac_generator(data = profitSWG$control, ini = 0.5, fin = 4.0)
```

```
##
```

```
## 1  0.1481608
## 2 -0.7036783
## 3 -1.5555175
## 4 -2.4073566
## 5 -3.2591958
## 6 -4.1110349
## 7 -4.9628741
## 8 -5.8147133
```

References

Hardaker, J. Brian, James W. Richardson, Gudbrand Lien, and Keith D. Schumann. 2004. "Stochastic efficiency analysis with risk aversion bounds: a simplified approach." *The Australian Journal of Agricultural and Resource Economics* 48 (2): 253–70. https://doi.org/10.1111/j.1467-8489.2004.00239.x.

Richardson, James W, and Joe L Outlaw. 2008. "Ranking risky alternatives: innovations in subjective utility analysis." *WIT Transactions on Information and Communication* 39 (1): 213–24.

Soto-Caro, Ariel, Feng Wu, and Zhengfei Guan. 2019. "Evaluating Pest Management Strategies: A Robust Method and Its Application to Strawberry Disease Management." Paper presented at AAEA annual meeting 2019, Atlanta, GA, 21-23 July.