Measures Library

Calculate measures based on the calculated cash flows.

# Libraries/Classes calling on BCN Class

Main

# Internal Library/Class dependencies

totReqCashFlows

totOptionalCashFlows

# External Library Dependencies

numpy

# Functions

Total Benefits, Costs, Non-investment Costs, and investment costs are already calculated and stored in the totReqFlows objects and do not need to be recalculated here.

# Pseudo Code

Begin Pseudocode

Import totReqCashFlows

Import totOptionalCashFlows

**checkCosts(totReqFlow object)**

if totReqFlow.totCostDisc != totReqFlow.totCostDiscInv + totReqFlow.totCostDiscNonInv

error in calculation

**end checkCosts**

**sumCosts(totReqFlow.totCostsDisc)**

calculate

Return totalBenefits

**end sumCosts**

**sumBenefits(totReqFlow.totCostsDisc)**

calculate

Return totalCosts

**end sumBenefits**

**sumInv(totReqFlow.totCostsDisc)**

calculate

Return totalCostsInv

**end sumInv**

**sumNonInv(totReqFlow.totCostsDisc)**

**end sumNonInv**

**netBenefits(altID,altIDBase)**

pull appropriate summations using sumCosts and sumBenefits

Return netBeneftis

**End netBenefits**

**netSavings(altID,altIDBase)**

pull appropriate summations using sumCosts

Return netSavings

**End netSavings**

**measBCR(altID,altIDBase)**

pull appropriate summations using netSavings, sumCostsInv

If denominator is not zero

If denominator > 0 and Numerator > 0

Return measBCR

Elseif Denominator < 0 and Numerator > 0, then BCR is infinity

Elseif Denominator < 0 and Numerator < 0, then BCR “Not Calculated”

Else

Return BCR incalculable as investment costs are equal

**End measBCR**

**measSIR(altID,altIDBase)**

pull appropriate summations using sumCostsInv, sumCostsNonInv

if denominator is not zero

If Delta I > 0 and savings in non-investment costs > 0.

Return measSIR

Elseif Denominator < 0, and numerator > 0, then SIR is infinity

Elseif Denominator < 0, and numerator < 0, then “Not Calculated”

Else

Return SIR incalculable as investment costs are equal

**End measSIR**

**measAIRR(altID,altIDBase)**

calculate SIR for altID

If SIR > 0,

Return measAIRR

Elseif SIR is negative, then do not calculate AIRR.

**end measAIRR**

**measDeltaQ (altID,altIDBase,tag)**

pull totalTagQ from altID and altIDBase for chosen tag

Calculate

Return measDeltaQ

**End measDeltaQ**

**measNSPerQ (altID,altIDBase,tag)**

get measDeltaQ(altID,altIDBase,tag) for chosen tag and netSavings(altID,altIDBase)

Calculate

Return measNSPerQ

**End measNSPerQ**

**measNSPerPctQ (altID,altIDBase,tag)**

get measDeltaQ(altID,altIDBase,tag) for chosen tag and netSavings(altID,altIDBase)

pull totalTagQ from altIDBase

Calculate

Return measNSPerPctQ

**End measNSPerPctQ**

**measNSPerPctQ (altID,altIDBase,tag)**

get measDeltaQ(altID,altIDBase,tag) for chosen tag and netSavings(altID,altIDBase)

pull totalTagQ from altIDBase

Calculate

Return measNSPerPctQ

**End measNSPerPctQ**

**measNSElasticity (altID,altIDBase,tag)**

get measDeltaQ(altID,altIDBase,tag) for chosen tag and netSavings(altID,altIDBase)

pull totalTagQ from altIDBase and totalCosts from altIDBase

Calculate

Return measNSElasticity

**End measNSElasticity**

**measIRR (altID)**

Techincally speaking we should be solving this, however the solution to this requires a root finding algorithm and repeatedly updating cash flows to obtain

**Use numpy.irr(totBenefitsNonDisc+totCostsNonDisc) if possible. If numpy.irr doesn’t work for out purposes let me know and we can discuss the appropriate algorithn to calculate this**

Return measIRR

**End measIRR**

**measSPP(altID)**

solve for the first time the following is true

If

Then SPP = T

Else repeat using T= T+1.

Return SPP unless study period is reached then SPP doesn’t exist

**End measSPP**

**measDPP(altID)**

If

Then DPP = T

Else repeat using T= T+1.

Repeat until Condition is true and Report DPP = T

Return DPP unless study period is reached then DPP doesn’t exist

**End measDPP**

**totalQuant(altID,tag)**

initialize quantSum = 0

loop through all total optional flow objects for the given altID

if tag == totalOptionalCashFlow.tag

sum all elements in totalOptionalCashFlow.tag (tagSum)

quantSum = quantSum + tagSum

return quantSum, quantUnits

**end totalQuant**

end Pseudocode