Main

Runs Everything

# Libraries/Classes calling on Main

None

# Internal Library/Class dependencies

Validate/Read

Flows Library

AlternativeSummary Class

TotalRequiredFlows Class

TotalOptionalFlows Class

SensitivitySummary Class

UncertaintySummary Class

cashFlows Library

Analysis Class

Alternative Class

BCN Class

Sensitivity Class

Scenario Class

# External Library Dependencies

None

# Functions

Main() – runs everything

# Pseudo Code

Optional inputs are in italics

Begin Pseudocode

Import Validate/Read

Import Flows Library

Import AlternativeSummary Class

Import TotalRequiredFlows Class

Import TotalOptionalFLows Class

Import SensitivitySummary Class

Import UncertaintySummary Class

Import cashFlows Library

Import Analysis Class

Import Alternative Class

Import BCN Class

Import Sensitivity Class

Import Scenario Class

main()

JSON file passes through authentication and initiates code

Call to validate/read to validate file and generate user defined objects

// Generate BCN level Flows (could be separated out as a separate method/library if desired)

For bcn in BCN object

bcnNonDiscFlow, bcnDiscFlow, quantList = cashFlows.bcnFlow(analysis.discountRate,bcnObject,studyPeriod,timestepCount)

initialize bcnFlow object variables via

bcnStorage(bcnName,altID,type,subtype,tag,bcnNonDiscFlow,bcnDiscFlow, quantList, quantUnt,quantList)

// Generate Total Cash Flows usng totalFlows from cashFlows library

For each Analysis.altID

cashFlows.totalFlows(bcnStorageObjectList,altID)

// Calculate measures

Find baseline Alternative

calculate measures for baseline (those relative to baseline can be left blank, the write to file section will deal with the specifics of blank output)

Store results in alternativeSummary object

For all non-baseline alternatives

Calculate measures

Store results in alternativeSummary object

// Sensitivity Calculations

Loop through all sensitivity objects

If sensitivity.globalVarBool == True

Loop through all BCN storage objects

Find the associated bcnStorage object for each user BCN object

Update bcnStorage.sensBool to True

If sensitivity.varName == “Inflation Rate”

Update the inflation rate from analysis.inflationRate using the information from the sensitivity object (specifically Change Type and Change Value)

Call to discounting library to calculate new discount rate using the updated inflation rate

Calculate cash flow using the new global variable via

bcnNonDiscFlow, bcnDiscFlow, quantList = cashFlows.bcnFlow(updated discountRate,bcnObject,studyPeriod,timestepCount)

For the associated bcnStorage object

update bcnStorage.sensFlowNonDisc = bcnNonDiscFlow

update bcnStorage.sensFlowDisc = bcnDiscFlow

update bcnStorage.sensFlowQuantList = quantList

Generate total cash flows via

Loop through all altIDs

cashFlows.totalFlows(bcnStorageObjectList,altID)

Calculate measures

Find baseline Alternative

For totalRequiredFlow Object corresponding to baseAltID and with sensBool == true

totalCostsBase = sumCosts(totReqFlow.totCostsDisc)

totalBenefitsBase = sumBenefits(totReqFlow.totCostsDisc)

totalInvBase = sumInv(totReqFlow.totCostsDisc)

totalNonInvBase = sumNonInv(totReqFlow.totCostsDisc)

irrBase = meas(altID)

Store results in senstivitySummary object using the calculated measures and the Sensitivity object ID

For all non-baseline alternatives

For totalRequiredFlow objects whose altID does NOT equal the baseAltID

totalCosts = sumCosts(totReqFlow.totCostsDisc)

totalBenefits = sumBenefits(totReqFlow.totCostsDisc)

totalInv = sumInv(totReqFlow.totCostsDisc)

totalNonInv = sumNonInv(totReqFlow.totCostsDisc)

netBenefits(totalBenefits,totalCosts,totalBenefitsBase,totalCostsBase)

netSavings = netSavings(totalCosts,totalCostsBase)

bcr = measBCR(netSavings,totalCostsInv,

totalCostsInvBase)

sir = measSIR(totalCostsInv,totalCostsNonInv,

totalCostsInvBase,totalCostsNonInvBase)

airr = measAIRR(analysis.reinvestRate,sir)

initialize list of tags

tagList == []

totalTagVal = []

quantUnitList = []

for each totalOptionalFlow for associated with the altID

if tag is in totalOptionalFlow append to tagList

for each tag in tagList

delQ = measDeltaQ (altID,altIDBase,tag)

NSPerQ = measNSPerQ(netSavings,altID,

altIDBase,tag)

NSPerPctQ = measNSPerPctQ(altID,

altIDBase,tag)

NSElas = measNSElasticity (altID,altIDBase,tag)

totalQuant, quantUnits = totalQuant(altID,tag)

append totalQuant to totalTagVal

append quantUnits to quantUnitList

irr = meas(altID)

spp = measSPP(altID)

dpp = measDPP(altID)

Store results in senstivitySummary object using the calculated measures and the Sensitivity object ID

If sensitivity.globalVarBool == false

Using the sensitivity.bcnID variable, find the associated bcnStorage object

For that object

Update bcnStorage.sensBool to True

Using sensitivity.varName, sensitivity.diffType and sensitivity.diffVal calculate altered value for bcn.value and

* + - 1. Store the old value of the variable
      2. use bcn.update to update the variable to the calculated value

calculate the new flow using the following but only for the altered bcn

bcnNonDiscFlow, bcnDiscFlow, quantList = cashFlows.bcnFlow(updated discountRate,bcnObject,studyPeriod,timestepCount)

For the associated bcnStorage object

update bcnStorage.sensFlowNonDisc = bcnNonDiscFlow

update bcnStorage.sensFlowDisc = bcnDiscFlow

update bcnStorage.sensFlowQuantList = quantList

Generate total cash flows via

Loop through all altIDs in the sensitivity.altID list

cashFlows.totalFlows(bcnStorageObjectList,altID)

for those altIDs not in the sensitivity.altID list copy the values from the totalFlows objects into a new totalFlows object except sensBool == true

if baseAltID is in sensitivity.altID list, calculate measures as above (for global variable)

else,

pull

totalCostsBase

totalBenefitsBase

totalInvBase

totalNonInvBase

from the totalRequiredFlows object with altID == baseAltID and sensBool == false

calculate measures for those altIDs in sensitivity.altID list using the pulled from the baseAltID w/ sensBool == true as the baseline measures

loop through bcnStorage objects that were altered and set sensBool == false and all sensFlows to lists of zeros

// Write to file

End main()

End Pseudocode