path

+ start: str
+ end: str
+ groups: list
+ startTimePeriod: str
+ endTimePeriod: str

+ __init__(self, start, end, startTimePeriod, endTimePeriod): self
+ showEnd(self): end
+ showEndTimePeriod(self): endTimePeriod
+ showStart(self): start
+ showStartTimePeriod(self): startTimePeriod
+ describePaths(self): print out path details

networkModel

+ omega: np.array
+ networkName: str
+ nodes: list
+ paths: list
+ timePeriods: list
+ nTimePeriods: int
+ nYears: int
+ popLenDistbiomass: np.array
+ eggProducingGroupLenDist: np.array

+ __init__(self, networkName, nYears, omega, timePeriods): self
+ initializePaths(self): self
+ describePaths(self): print paths
+ addNodeList(self, nodeList): self
+ nNodes(self): nNodes
+ runNetworkSimulation(self): self
+ describeNetwork(self): print about network

helpers

+ pathOutListFunction(pathsOut, pathsOutProb): dict
+ initializeModelFromCSVs(dfNetwork, dfNode, dfGroups): networkModel

≪builds≫

node

+ nodeName: str
+ groups: list
+ pathsOut: dict
+ pathsIn: list
+ nodeBiomass: float
+ timePeriod: str

+ __init__(self, nodeName, timePeriod): self
+ createNodePop(self, nYears): self.nodePop
+ showTimePeriod(self): timePeriod
+ listGroups(self): groups
+ describeNodes(self): list of nodes
+ addPathsOut(self, pathsOut): self.pathsOut
+ addPathsInt(self, pathsIn): self.pathsIn
+ addGroups(self, groupName): self.groupName
+ nGroups(self): len(nGroups)
+ showNodeName(self): nodeName
+ calculateNodePopulation(self): self.NodePop
+ plotNodePop(self, nYears): plot
+ plotNodeGroups(self, nYears): plot

linearRecruitment

+ juvenile: np.array
+ lenghtWeight: np.array
+ survival: float or np.array
+ probabilityReproducing: float or np.array
+ eggTransition: float
+ eggPerkg: float

+ __init__(self, omega, lengthWeight, probabilityReproducing, eggTransition, eggPerkg, muJ, sigmaJ): self
+ __self__(self, omega, omegaPrime): np.array

probabilityOfReproducing

+ alphaL: float
+ betaL: float
+ minL: float
+ maxL: float

+ __call__(self, z): float

lengthWeight

+ alphaLW: float
+ betaLW: float

+ __init__(self, alphaLW, betaLW): self
+ __call__(self, omega): float

densityNone

+ __call__(self, biomass): 1

densityNegExp

+ a: float
+ b: float

+ __init__(self, a, b): self
+ __call__(self, biomass): float

survival

+ alphaS: float
+ betaS: float
+ minS: float
+ maxS: float

+ __call__(self, z): float

logistic

+ alphaL: float
+ betaL: float
+ minL: float
+ maxL: float

+ __init__(self, aG, kG, sigmaG): self
+ __call__(z): float

growthVB

+ aG: float
+ kG: float
+ sigmaG: float

+ __init__(self, aG, kG, sigmaG): self
+ __call__(self, tzIn, zPrimeIn): float

group

+ popLenDist: np.array
+ groupProduceEggs: Bool
+ groupImpactSexRatio: Bool
+ groupImpactViability: Bool
+ nYears: int
+ groupSex: str
+ groupName: str
+ popSize:np.array
+ popLenDist: np.array
+ survival: np.array or float
+ growth: np.array
+ recruitment: np.array
+ density: np.array
+ lengthWeight: np.array
+ groupOffspringPfemale: float
+ groupOffspringViability = float
+ omega: np.array
+ hWidth: float
+ pulseIntroductionString: np.array

+ __init__(self, groupName, popSize0, popLengDist0, omega, nYears, survival, growth, recruitment, density, lengthweight): self
+ timeStepGroup(self, t, tPlusOne, pReferenceGroupBirth, offspringViability, recruitGroup, popLenDistbiomass, referenceSex): self
+ movement(self, immigration, emigration, t): self
+ showGroupPopSize(self): popSize
+ showGroupProduceEggs(self): groupProduceEggs
+ showGroupName(self): groupName
+ showGroupSex(self): groupSex
+ plotLengthTime(self): plots
+ plotPop(self): plot