R code and associated data for calculation of multi-criteria decision analysis for nutrient management in San Francisco Bay.

Note: To work properly, most of the code has to be run in the given order –you cannot skip to the code chunk that is most interesting to you and run it. This document was designed as an R Markdown file.

Save the following tables as csv files in your working directory:

Attribute tables (These have the basic details and best and worst values of the attributes which are used to measure fulfillment of each objective. SH3 and SH7 had different sets of objectives than the other stakeholders, so they require their own attribute tables.)

- attrs.nutrients.csv

name	main objective	fundamental objective	attribute	unit	worst	best
wq	ecosystem	good nutrient management	Water quality	%	95	5
hab	ecosystem	good nutrient management	Wetland habitat	hectares	0	4800
supply	resource recovery	good nutrient management	Water supply	MGD	0	190
recovery	resource recovery	good nutrient management	Nutrient recovery	kg/year	0	8500000
CEC	resource recovery	good nutrient management	CECs	kg/year	137	42
adapt	intergen equity	good nutrient management	Ease of adaptation	%	0	100
slr	intergen equity	good nutrient management	Sea level rise	scale_constr	-10	10
CO2	intergen equity	good nutrient management	CO2 emissions	tonnes CO2 eq/year	900000	195000
ease_use	social support	good nutrient management	Ease of use	%	0	100
access	social support	good nutrient management	Shoreline access	Integer	0	17
permit	social support	good nutrient management	Permitting	%	0	100
reliable	costs	good nutrient management	Reliability	%	50	99
costs	costs	good nutrient management	Costs	$	8000000000	0

- attrs.nutrients.SH3.csv

name	main objective	fundamental objective	attribute	unit	worst	best
supply	resource recovery	good nutrient management	Water supply	MGD	0	190
recovery	resource recovery	good nutrient management	Nutrient recovery	kg/year	0	8500000

CEC	resource recovery	good nutrient management	CECs	kg/year	137	42
adapt	intergen equity	good nutrient management	Ease of adaptation	%	0	100
slr	intergen equity	good nutrient management	Sea level rise	scale_constr	-10	10
CO2	intergen equity	good nutrient management	CO2 emissions	tonnes CO2 eq/year	900000	200000
access	social support	good nutrient management	Shoreline access	Integer	0	17
permit	social support	good nutrient management	Permitting	%	0	100
reliable	costs	good nutrient management	Reliability	%	50	99
costs	costs	good nutrient management	Costs	$	8000000000	0

- attrs.nutrients.SH7.csv

name	**main objective**	**fundamental objective**	**attribute**	**unit**	**worst**	**best**
wq	ecosystem	good nutrient management	Water quality	%	95	5
hab	ecosystem	good nutrient management	Wetland habitat	hectares	0	5200
supply	resource recovery	good nutrient management	Water supply	MGD	0	190
CEC	resource recovery	good nutrient management	CECs	kg/year	137	22
adapt	intergen equity	good nutrient management	Ease of adaptation	%	0	100
slr	intergen equity	good nutrient management	Sea level rise	scale_constr	-10	10
CO2	intergen equity	good nutrient management	CO2 emissions	tonnes CO2 eq/year	900000	180000
ease_use	social support	good nutrient management	Ease of use	%	0	100
access	social support	good nutrient management	Shoreline access	Integer	0	17
permit	social support	good nutrient management	Permitting	%	0	100
reliable	costs	good nutrient management	Reliability	%	50	100
costs	costs	good nutrient management	Costs	$	8000000000	0

Data for modeling outcomes of management options

- wetland_suitability_area.csv

Rank	Burlingame	EBDA	Millbrae	Palo Alto	San Jose - Santa Clara	San Mateo	SFO Airport	SF (Southeast Plant)	South SF and San Bruno	Sunnyvale	Silicon Valley Clean Water
1 (m2)	158285.05	3772022.22	598712.18	297677.34	4087131.37	28066.77	537308.17	13255.86	489604.22	1471529.65	19262.82
2 (m2)		4666197.41			3494102.46					1818.46	4625.93
3 (m2)	212269.81	30375516.51	154476.9	734748.49	5117377.65	170212	195642.06	954970.01	250842.84	1000000	207574.96

- Dry season discharge summary_BACWA_2012_2016.csv

Unit	Parameter	Burlingame WWTP	EBDA Outfall	Millbrae WWTP	Palo Alto WQCP	San Jose /Santa Clara	San Mateo WWTP	SF Arprt Mel Leong	SF-SE Plant	South San Francisco	Sunnyvale WPCP	SVCW WWTP
mgd	Flow	2.62	53.24	1.33	20.73	78.67	9.40	1.04	54.10	8.11	8.35	11.91
kg/day	Ammonia, Total (as N)	249.49	6526.18	234.32	17.29	182.22	1357.60	196.06	8659.56	809.32	14.32	2006.75
kg/day	TKN	294.21	7207.77	270.36	35.65	458.23	1570.73	172.08	9083.59	973.18	108.12	2066.59
kg/day	Nitrite Plus Nitrate (as N)	69.48	766.65	2.06	2329.68	4331.04	37.27	20.92	752.39	122.35	388.47	68.56
kg/day	Nitrogen, Total (as N)	363.69	7953.08	272.41	2365.42	4789.27	1608.01	193.01	9836.13	1095.53	516.76	2140.87
kg/day	Phosphorus, Total (as P)	56.47	495.82	15.79	390.28	234.58	129.09	13.44	171.79	158.52	185.60	183.25
kg/day	Orthophosphate, Dissolved (as P)	75.17	481.63	15.14	376.68	216.15	152.63	14.34	312.50	167.30	176.95	266.11

Parameters for modeling uncertainty

- uncertainty_distributions.csv

	do.nothing	wetland.levee	wetland.openwater	recycle.irrig	recycle.brine	urine.early	urine.incentive	opt	level2	level3
wq	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax
hab	Determined	Minmax	Minmax	Determined	Determined	Determined	Determined	Determined	Determined	Determined
supply	Determined	Determined	Determined	Minmax	Minmax	Determined	Determined	Determined	Determined	Determined
recovery	Determined	Determined	Determined	Minmax	Determined	Minmax	Minmax	Determined	Determined	Determined
CEC	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax
adapt	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax

	do.nothing	wetland.levee	wetland.openwater	recycle.irrig	recycle.brine	urine.early	urine.incentive	opt	level2	level3
slr	Determined	Determined	Determined	Determined	Determined	Determined	Determined	Determined	Determined	Determined
CO2	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Determined	Minmax	Minmax
ease_use	Determined	Determined	Determined	Determined	Minmax	Minmax	Minmax	Determined	Determined	Determined
access	Determined	Minmax	Minmax	Determined	Determined	Determined	Determined	Determined	Determined	Determined
permit	Determined	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax
reliable	Determined	Minmax	Minmax	Minmax	Normal	Minmax	Minmax	Normal	Normal	Normal
costs	Determined	Minmax	Minmax	Beta	Minmax	Minmax	Minmax	Normal	Normal	Normal

Stakeholder preferences

- pars.nutrients.sensitivity.csv

name	val.avg	val.SH7	val.SH3	val.SH2	val.SH8	val.SH5	val.SH1	val.SH4	val.SH6	val.SH9
w.ecosystem	0.26	0.2	NA	0.28	0.3	0.19	0.26	0.26	0.29	0.29
w.wq	0.57	0.5	NA	0.67	0.43	0.53	0.56	0.63	0.57	0.67
w.hab	0.43	0.5	NA	0.33	0.57	0.47	0.44	0.38	0.43	0.33
w.r.recovery	0.16	0.2	0.08	0.25	0.12	0.18	0.18	0.21	0.06	0.14
w.supply	0.36	0.5	0	0.5	0.53	0.34	0.23	0.3	0.44	0.36
w.recovery	0.26	NA	0	0.45	0.13	0.28	0.32	0.2	0.33	0.36
w.CEC	0.41	0.5	1	0.05	0.34	0.38	0.45	0.5	0.22	0.27
w.equity	0.24	0.2	0.36	0.25	0.19	0.25	0.19	0.21	0.32	0.14
w.adapt	0.46	0.32	0.95	0.45	0.36	0.39	0.45	0.48	0.29	0.44
w.slr	0.33	0.36	0	0.36	0.45	0.33	0.23	0.29	0.57	0.33
w.CO2	0.2	0.32	0.05	0.17	0.17	0.3	0.3	0.21	0.11	0.21
w.social	0.17	0.2	0.16	0.08	0.13	0.23	0.15	0.16	0.16	0.21
w.ease_use	0.24	0.31	NA	0.5	0.11	0.15	0.36	0.32	0.1	0.07
w.access	0.28	0.31	0.09	0.25	0.37	0.11	0.18	0.45	0.5	0.21
w.permit	0.51	0.38	0.91	0.25	0.53	0.74	0.45	0.23	0.4	0.71
w.tot.costs	0.21	0.2	0.4	0.14	0.25	0.16	0.22	0.16	0.16	0.21
w.reliable	0.59	0.47	0.67	0.5	0.69	0.67	0.63	0.63	0.53	0.57
w.costs	0.41	0.53	0.33	0.5	0.31	0.33	0.38	0.38	0.47	0.43
c.ecosystem	0	0	0	0	0	0	0	0	0	0
c.wq	0	0	0	0	0	0	0	0	0	0
c.hab	0	0	0	0	0	0	0	0	0	0

c.r.recovery	0	0	0	0	0	0	0	0	0	0
c.supply	0	0	0	0	0	0	0	0	0	0
c.recovery	0	0	0	0	0	0	0	0	0	0
c.CEC	0	0	0	0	0	0	0	0	0	0
c.equity	0	0	0	0	0	0	0	0	0	0
c.adapt	0	0	0	0	0	0	0	0	0	0
c.slr	0	0	0	0	0	0	0	0	0	0
c.CO2	0	0	0	0	0	0	0	0	0	0
c.social	0	0	0	0	0	0	0	0	0	0
c.ease_use	0	0	0	0	0	0	0	0	0	0
c.access	0	0	0	0	0	0	0	0	0	0
c.permit	0	0	0	0	0	0	0	0	0	0
c.tot.costs	0	0	0	0	0	0	0	0	0	0
c.reliable	0	0	0	0	0	0	0	0	0	0
c.costs	0	0	0	0	0	0	0	0	0	0
r.nutrients.uncertainty.midnodes	0	0	0	0	0	0	0	0	0	0
add.ecosystem	1	1	1	1	1	1	1	1	1	1
add.wq	1	1	1	1	1	1	1	1	1	1
add.hab	1	1	1	1	1	1	1	1	1	1
add.r.recovery	1	1	1	1	1	1	1	1	1	1
add.supply	1	1	1	1	1	1	1	1	1	1
add.recovery	1	1	1	1	1	1	1	1	1	1
add.CEC	1	1	1	1	1	1	1	1	1	1
add.equity	1	1	1	1	1	1	1	1	1	1
add.adapt	1	1	1	1	1	1	1	1	1	1
add.slr	1	1	1	1	1	1	1	1	1	1
add.CO2	1	1	1	1	1	1	1	1	1	1
add.social	1	1	1	1	1	1	1	1	1	1
add.ease_use	1	1	1	1	1	1	1	1	1	1
add.access	1	1	1	1	1	1	1	1	1	1
add.permit	1	1	1	1	1	1	1	1	1	1
add.tot.costs	1	1	1	1	1	1	1	1	1	1
add.reliable	1	1	1	1	1	1	1	1	1	1
add.costs	1	1	1	1	1	1	1	1	1	1
min.ecosystem	0	0	0	0	0	0	0	0	0	0
min.wq	0	0	0	0	0	0	0	0	0	0
min.hab	0	0	0	0	0	0	0	0	0	0
min.r.recovery	0	0	0	0	0	0	0	0	0	0

min.supply	0	0	0	0	0	0	0	0	0	0
min.recovery	0	0	0	0	0	0	0	0	0	0
min.CEC	0	0	0	0	0	0	0	0	0	0
min.equity	0	0	0	0	0	0	0	0	0	0
min.adapt	0	0	0	0	0	0	0	0	0	0
min.slr	0	0	0	0	0	0	0	0	0	0
min.CO2	0	0	0	0	0	0	0	0	0	0
min.social	0	0	0	0	0	0	0	0	0	0
min.ease_use	0	0	0	0	0	0	0	0	0	0
min.access	0	0	0	0	0	0	0	0	0	0
min.permit	0	0	0	0	0	0	0	0	0	0
min.tot.costs	0	0	0	0	0	0	0	0	0	0
min.reliable	0	0	0	0	0	0	0	0	0	0
min.costs	0	0	0	0	0	0	0	0	0	0
cd.ecosystem	0	0	0	0	0	0	0	0	0	0
cd.wq	0	0	0	0	0	0	0	0	0	0
cd.hab	0	0	0	0	0	0	0	0	0	0
cd.r.recovery	0	0	0	0	0	0	0	0	0	0
cd.supply	0	0	0	0	0	0	0	0	0	0
cd.recovery	0	0	0	0	0	0	0	0	0	0
cd.CEC	0	0	0	0	0	0	0	0	0	0
cd.equity	0	0	0	0	0	0	0	0	0	0
cd.adapt	0	0	0	0	0	0	0	0	0	0
cd.slr	0	0	0	0	0	0	0	0	0	0
cd.CO2	0	0	0	0	0	0	0	0	0	0
cd.social	0	0	0	0	0	0	0	0	0	0
cd.ease_use	0	0	0	0	0	0	0	0	0	0
cd.access	0	0	0	0	0	0	0	0	0	0
cd.permit	0	0	0	0	0	0	0	0	0	0
cd.tot.costs	0	0	0	0	0	0	0	0	0	0
cd.reliable	0	0	0	0	0	0	0	0	0	0
cd.costs	0	0	0	0	0	0	0	0	0	0

- Inputs/Stakeholder points.csv

Name_ID	Name_R	Role	Water quality	Wetland habitat	Water supply	Nutrient recovery	CECs	Ease of adaptation	Sea level rise	CO2 emissions	Ease of use	Shoreline access	Permitting	Reliability	Costs	Ecosystem	Improve.wastewater	Intergen.equity	Social.support	Low.costs
xx	SH7	WW	75	75	75	NA	75	75	85	75	75	75	90	90	100	75	75	78.33	80	95
xx	SH3	WW	NA	NA	0	0	20	90	0	4.5	NA	4	40	100	50	NA	6.66	31.5	22	75
xx	SH2	WW	100	50	90	81	9	90	72	36	30	15	15	50	50	75	60	66	20	50
xx	SH8	Eco	75	100	40	10	26	52	65	26	9	31.5	45	85	38.25	87.5	25.33	47.66	28.5	61.625
xx	SH5	Eco	75	67.5	63	52.5	70	100	85	70	18	13.5	90	65	32.5	71.25	61.83	85	40.5	48.75
xx	SH1	WW	100	80	35	49	70	75	37.5	52.5	48	24	60	85	51	90	51.33	55	44	68
xx	SH4	Reg	100	60	48	32	80	80	48	40	42	60	30	60	36	80	53.33	56	44	48
xx	SH6	Reg	90	67.5	25	18.75	12.5	50	100	25	10	50	40	50	45	78.75	18.75	58.33	33.33	47.5
xx	SH9	Reg	100	50	50	50	37.5	50	37.5	25	7.5	22.5	75	75	56.25	75	45.83	37.5	35	65.625

Install the following packages (this only has to be done once).

```
install.packages("utility")
install.packages("fitdistrplus")
install.packages("truncnorm")
install.packages("RColorBrewer")
install.packages("plyr")
```

Load the following libraries (this has to be done every time you run the code).

```
library(utility)
library(fitdistrplus)
library(truncnorm)
library(RColorBrewer)
library(plyr)
```

Set the scenario parameters. Population can vary between .87 and 1.6, climate impacts between 0 (no impact of sea level rise on existing wastewater treatment plant operations) and 5 (large impact), and ecological threshold is 0 (current ecological resilience to nutrient loading) or 1 (increased ecological sensitivity to nutrient loading). For the ‘Status quo’ scenario, there is 33% population growth (1.33 multiplier), 0 impact of climate change, and ecological threshold is 1.

```
scenario<-c(population= 1.33,
            climate.impact = 0,
            eco.threshold = 1)
```

Load attribute tables.

```
attrs.nutrients <- read.table(paste("attrs.nutrients.csv", sep = ""), header
= TRUE, sep=",")
head(attrs.nutrients)
class(attrs.nutrients$worst)
attrs.nutrients$name <- as.character(attrs.nutrients$name)

attrs.nutrients.SH3 <- read.table(paste("attrs.nutrients.SH3.csv", sep = ""),
header = TRUE, sep=",")
attrs.nutrients.SH3$name <- as.character(attrs.nutrients.SH3$name)

attrs.nutrients.SH7 <- read.table(paste("attrs.nutrients.SH7.csv", sep = ""),
header = TRUE, sep=",")
attrs.nutrients.SH7$name <- as.character(attrs.nutrients.SH7$name)
```

Set aggregation parameters for the objectives hierarchy.

```
aggregation.parameters<-function(add,min,cd){
  pref.nutrients.sensitivity <-
read.table(paste("pars.nutrients.sensitivity.csv", sep = ""), header = TRUE,
sep=",")
  pref.nutrients.sensitivity[38:55,2:11]<-add ##additive
  pref.nutrients.sensitivity[56:73,2:11]<-min ##minimum
  pref.nutrients.sensitivity[74:91,2:11]<-cd ##cob douglass
  assign("pref.nutrients.sensitivity", pref.nutrients.sensitivity,envir =
globalenv())
  assign("aggregation.all",c(add,min,cd),envir = globalenv())
}

aggregation.parameters(add=1,min=0,cd=0)
pref.nutrients.sensitivity
aggregation.all
```

Specify aggregation type for objectives hierarchy. Add = additive aggregation; Min = minimum aggregation; cd = Cobb-Douglas aggregation.

```
aggregation.parameters(add=1,min=0,cd=0)
```

Make the objectives hierarchy for all the stakeholders (except SH3 and SH7– we'll make their objectives hierarchies next).

```
# create lowest level nodes (marginal value functions)
#################################################################

for (i in 1:nrow(attrs.nutrients)) {
  print(paste("No. ", i, sep = ""))
  assign(print(attrs.nutrients$name[i], sep=""),
         utility.endnode.parfun1d.create(name.node = attrs.nutrients$name[i],
                                         name.attrib =
as.character(attrs.nutrients$name[i]) ,
                                         range =
c(min(attrs.nutrients[i,6:7]),max(attrs.nutrients[i,6:7])),
                                         name.fun = "utility.fun.exp",
```

```
                                         par =
c(0,attrs.nutrients$worst[i],attrs.nutrients$best[i]),
                                         names.par =
c(paste("c.",attrs.nutrients$name[i], sep=""),"worst.v","best.v"),
                                         utility = FALSE,
                                         required = FALSE,
                                         col = "black",
                                         shift.levels = 0))
}

# create mid-level aggregation nodes
#################################################################

ecosystem <- utility.aggregation.create(name.node = "ecosystem",
                                        nodes = list(wq, hab),
                                        name.fun = "utility.aggregate.mix",
                                        par= c(1,1,1,0,0),
                                        names.par = c("w.wq","w.hab",

"add.ecosystem","min.ecosystem", "CD.ecosystem"),
                                        required= FALSE)

r.recovery <- utility.aggregation.create(name.node = "r.recovery",
                                         nodes = list(supply, recovery, CEC),
                                         name.fun = "utility.aggregate.mix",
                                         par= c(1,1,1,aggregation.all),
                                         names.par =
c("w.suppy","w.recovery","w.CEC",

"add.r.recovery","min.r.recovery", "CD.r.recovery"),
                                         required= FALSE)

equity <- utility.aggregation.create(name.node = "equity",
                                     nodes = list(adapt, slr, CO2),
                                     name.fun = "utility.aggregate.mix",
                                     par= c(1,1,1,aggregation.all),
                                     names.par = c("w.adapt","w.slr","w.C02",
                                                   "add.equity","min.equity",
"CD.equity"),
                                     required= FALSE)

social <- utility.aggregation.create(name.node = "social",
                                     nodes = list(ease_use, access, permit),
                                     name.fun = "utility.aggregate.mix",
                                     par= c(1,1,1,aggregation.all),
                                     names.par =
c("w.ease_use","w.access","w.permit",
                                                   "add.social","min.social",
"CD.social"),
                                     required= FALSE)

tot.costs <- utility.aggregation.create(name.node = "tot.costs",
                                        nodes = list(reliable, costs),
                                        name.fun = "utility.aggregate.mix",
                                        par= c(1,1,aggregation.all),
                                        names.par =
c("w.reliable","w.costs","w.CEC",
```

```
"add.tot.costs","min.tot.costs", "CD.tot.costs"),
                                        required= FALSE)

#####

## create top-level aggregation node
#################################################################

nutrients.uncertainty.midnodes <- utility.aggregation.create(name.node =
"nutrients.uncertainty.midnodes",
                                                              nodes =
list(ecosystem, r.recovery, equity, social, tot.costs),
                                                              name.fun =
"utility.aggregate.mix",
                                                              par=
c(1,1,1,1,1,aggregation.all),
                                                              names.par = c(
"w.ecosystem", "w.r.recovery", "w.equity", "w.social", "w.tot.costs",

"add.nutrients.uncertainty","min.nutrients.uncertainty",
"CD.nutrients.uncertainty"),
                                                              required= FALSE)
# Convert to utility
nutrients.uncertainty.midnodes.u <-
utility.conversion.parfun.create(name.node =
"nutrients.uncertainty.midnodes.u",
                                                                      node =
nutrients.uncertainty.midnodes,
                                                                      name.fun
= "utility.fun.exp",
                                                                      par =
c(0,0,1),

names.par = c("r.nutrients.uncertainty","worst.u","best.u"),

required= FALSE)
```

Load general objects.

```
criteria <- c("wq", "hab",  "supply",   "recovery", "CEC",  "adapt",
"slr",  "CO2",  "ease_use", "access",   "permit",   "reliable", "costs")
options <- c("Do nothing", "Wetland levee", "Wetland openwater", "Recycle
irrig.", "Recycle brineline", "Urine early", "Urine incentives", "Opt.",
"Level 2", "Level 3")
options.no.brine <- options[c(1:4, 6:10)]
SHs<- c("SH1", "SH2", "SH3", "SH4", "SH5", "SH6", "SH7", "SH8", "SH9")
Criteria.names<- c("Water Quality", "Wetland Habitat", "Water Supply",
"Nutrient Recovery", "CECs",
                   "Ease of Adaptation", "Sea Level Rise", "CO2 Emissions",
"Ease of Use",
                   "Shoreline Access", "Permitting", "Reliability" ,"Costs")
Criteria.names.main.obj <- c("Ecosystem", "Improve wastewater", "Intergen.
Equity", "Social support", "Low costs")
```

Make objectives hierarchy for SH3.

```
# create lowest level nodes (marginal value functions)
#################################################################
for (i in 1:nrow(attrs.nutrients.SH3)) {
  print(paste("No. ", i, sep = ""))
  assign(print(attrs.nutrients.SH3$name[i], sep=""),
         utility.endnode.parfun1d.create(name.node =
attrs.nutrients.SH3$name[i],
                                         name.attrib =
as.character(attrs.nutrients.SH3$name[i]) ,
                                         range =
c(min(attrs.nutrients.SH3[i,6:7]),max(attrs.nutrients.SH3[i,6:7])),
                                         name.fun = "utility.fun.exp",
                                         par =
c(0,attrs.nutrients.SH3$worst[i],attrs.nutrients.SH3$best[i]),
                                         names.par =
c(paste("c.",attrs.nutrients.SH3$name[i], sep=""),"worst.v","best.v"),
                                         utility = FALSE,
                                         required = FALSE,
                                         col = "black",
                                         shift.levels = 0))
}


# create mid-level aggregation nodes
#################################################################

r.recovery <- utility.aggregation.create(name.node = "r.recovery",
                                         nodes = list(supply, recovery, CEC),
                                         name.fun = "utility.aggregate.mix",
                                         par= c(1,1,1,1,0,0),
                                         names.par =
c("w.suppy","w.recovery","w.CEC",

"add.r.recovery","min.r.recovery", "CD.r.recovery"),
                                         required= FALSE)

equity <- utility.aggregation.create(name.node = "equity",
                                     nodes = list(adapt, slr, CO2),
                                     name.fun = "utility.aggregate.mix",
                                     par= c(1,1,1,1,0,0),
                                     names.par = c("w.adapt","w.slr","w.C02",
                                                   "add.equity","min.equity",
"CD.equity"),
                                     required= FALSE)

social.SH3 <- utility.aggregation.create(name.node = "social.SH3",
                                     nodes = list(access, permit),
                                     name.fun = "utility.aggregate.mix",
                                     par= c(1,1,1,0,0),
                                     names.par = c("w.access","w.permit",

"add.social.SH3","min.social.SH3", "CD.social.SH3"),
                                     required= FALSE)

tot.costs <- utility.aggregation.create(name.node = "tot.costs",
                                        nodes = list(reliable, costs),
                                        name.fun = "utility.aggregate.mix",
```

```
                                        par= c(1,1,1,0,0),
                                        names.par =
c("w.reliable","w.costs","w.CEC",

"add.tot.costs","min.tot.costs", "CD.tot.costs"),
                                        required= FALSE)

#####

## create top-level aggregation node
##################################################################

nutrients.sowatt.midnodes.SH3 <- utility.aggregation.create(name.node =
"nutrients.sowatt.midnodes.SH3",
                                                            nodes =
list(r.recovery, equity, social.SH3, tot.costs),
                                                            name.fun =
"utility.aggregate.mix",
                                                            par=
c(1,1,1,1,1,0,1),
                                                            names.par =
c("w.r.recovery", "w.equity", "w.social.SH3", "w.tot.costs",

"add.nutrients.sowatt.midnodes.SH3","min.nutrients.sowatt.midnodes.SH3",
"CD.nutrients.sowatt.midnodes.SH3"),
                                                            required= FALSE)

# Convert to utility
nutrients.sowatt.midnodes.SH3.u <- utility.conversion.parfun.create(name.node
= "nutrients.sowatt.midnodes.SH3.u",
                                                                    node =
nutrients.sowatt.midnodes.SH3,
                                                                    name.fun
= "utility.fun.exp",
                                                                    par =
c(0,0,1),
                                                                    names.par
= c("r.nutrients.sowatt.midnodes.SH3","worst.u","best.u"),
                                                                    required=
FALSE)
```

Make objectives hierarchy for SH7.

```
# create lowest level nodes (marginal value functions)
##################################################################
for (i in 1:nrow(attrs.nutrients.SH7)) {
  print(paste("No. ", i, sep = ""))
  assign(print(attrs.nutrients.SH7$name[i], sep=""),
         utility.endnode.parfun1d.create(name.node =
attrs.nutrients.SH7$name[i],
                                         name.attrib =
as.character(attrs.nutrients.SH7$name[i]) ,
                                         range =
c(min(attrs.nutrients.SH7[i,6:7]),max(attrs.nutrients.SH7[i,6:7])),
                                         name.fun = "utility.fun.exp",
```

```
                                        par =
c(0,attrs.nutrients.SH7$worst[i],attrs.nutrients.SH7$best[i]),
                                        names.par =
c(paste("c.",attrs.nutrients.SH7$name[i], sep=""),"worst.v","best.v"),
                                        utility = FALSE,
                                        required = FALSE,
                                        col = "black",
                                        shift.levels = 0))
}


# create mid-level aggregation nodes
#################################################################

ecosystem <- utility.aggregation.create(name.node = "ecosystem",
                                       nodes = list(wq, hab),
                                       name.fun = "utility.aggregate.mix",
                                       par= c(1,1,1,0,0),
                                       names.par = c("w.wq","w.hab",

"add.ecosystem","min.ecosystem", "CD.ecosystem"),
                                       required= FALSE)

r.recovery.SH7 <- utility.aggregation.create(name.node = "r.recovery.SH7",
                                        nodes = list(supply, CEC),
                                        name.fun = "utility.aggregate.mix",
                                        par= c(1,1,1,0,0),
                                        names.par = c("w.suppy","w.CEC",

"add.r.recovery.SH7","min.r.recovery.SH7", "CD.r.recovery.SH7"),
                                        required= FALSE)

equity <- utility.aggregation.create(name.node = "equity",
                                    nodes = list(adapt, slr, CO2),
                                    name.fun = "utility.aggregate.mix",
                                    par= c(1,1,1,1,0,0),
                                    names.par = c("w.adapt","w.slr","w.C02",
                                                  "add.equity","min.equity",
"CD.equity"),
                                    required= FALSE)

social <- utility.aggregation.create(name.node = "social",
                                    nodes = list(ease_use, access, permit),
                                    name.fun = "utility.aggregate.mix",
                                    par= c(1,1,1,1,0,0),
                                    names.par =
c("w.ease_use","w.access","w.permit",
                                                  "add.social","min.social",
"CD.social"),
                                    required= FALSE)

tot.costs <- utility.aggregation.create(name.node = "tot.costs",
                                       nodes = list(reliable, costs),
                                       name.fun = "utility.aggregate.mix",
                                       par= c(1,1,1,0,0),
                                       names.par =
c("w.reliable","w.costs","w.CEC",
```

```
"add.tot.costs","min.tot.costs", "CD.tot.costs"),
                                        required= FALSE)

#####

## create top-level aggregation node
#################################################################

nutrients.sowatt.midnodes.SH7 <- utility.aggregation.create(name.node =
"nutrients.sowatt.midnodes.SH7",
                                                            nodes =
list(ecosystem, r.recovery.SH7, equity, social, tot.costs),
                                                            name.fun =
"utility.aggregate.mix",
                                                            par=
c(1,1,1,1,1,1,0,1),
                                                            names.par = c(
"w.ecosystem", "w.r.recovery.SH7", "w.equity", "w.social", "w.tot.costs",

"add.nutrients.sowatt.midnodes.SH7","min.nutrients.sowatt.midnodes.SH7",
"CD.nutrients.sowatt.midnodes.SH7"),
                                                            required= FALSE)

# Convert to utility
nutrients.sowatt.midnodes.SH7.u <- utility.conversion.parfun.create(name.node
= "nutrients.sowatt.midnodes.SH7.u",
                                                               node =
nutrients.sowatt.midnodes.SH7,
                                                               name.fun =
"utility.fun.exp",
                                                               par =
c(0,0,1),
                                                               names.par =
c("r.nutrients.sowatt.midnodes.SH7","worst.u","best.u"),
                                                               required=
FALSE)
```

Load data for calculating attributes for all management options.

```
#WETLAND AREA SUITABILITY DATA
##############################
# Load csv of Ian Wren's/SFEI's GIS assessment of wetland area
wetland.area.data <- read.csv ("wetland_suitability_area.csv")
wetland.area.data [is.na(wetland.area.data)] <- 0
wetland.area.data

# Convert to hectares
wetland.ha <- wetland.area.data [,2:12]/10000
wetland.ha

# Convert to acres
wetland.acre <- wetland.area.data [,2:12]/4046.86
wetland.acre

# DISCHARGE DATA
```

```
################


# Load csv of all nutrient loading and flows (2012-2016)#
# units is average in the dry season (BACWA definition of May  1- Sept. 30)

Discharge.data.current <- read.csv("Dry season discharge
summary_BACWA_2012_2016.csv")
Discharge.data.current

# for future scenarios, assume loads and flows scale with population growth,
but concentrations stay the same
Discharge.scenario <-
cbind(Discharge.data.current[,1:2],Discharge.data.current
[,3:13]*scenario[1])
Discharge.scenario

# Calculate dry season average TN concentrations for each plant (mg/L)

Discharge.TN.conc.scenario <- Discharge.scenario
[5,3:13]/Discharge.scenario[1,3:13]*10^6/3785412
Discharge.TN.conc.scenario
```

Calculate the parameters for each management option.

```
#do nothing
###########
do.nothing.load <- sum (Discharge.scenario [5, 3:13]) # in kg/day
SMX <- 200 # ng/L SMX in secondary treated effluent
# here do.nothing.CEC is how much SMX is produced
do.nothing.CEC <- sum((data.frame(Discharge.scenario [1, c(3:5, 8:11, 13)] *
SMX,
                                  Discharge.scenario [1, c(6,7,12)] * .85 *
SMX)*
                         365 * 10^6 /.26417 / 10^9 / 10^3)) # final unit is
kg/year SMX, days/year, gal/day, L/gal, g/ng, kg/g
do.nothing.CEC.WWTPs <- (data.frame(Discharge.scenario [1, c(3:5, 8:11, 13)]
* SMX,
                                    Discharge.scenario [1, c(6,7,12)] * .85 *
SMX)*
                           365 * 10^6 /.26417 / 10^9 / 10^3)
do.nothing.reliable <- 100 # realiably nothing.
do.nothing.cost <- 0
do.nothing.CO2 <- ((.5 # in kg CO2e/m3, for secondary treatment, model as
minmax distribution +/- 30%
                    * sum(data.frame(Discharge.scenario [1, c(3:5, 8:11,
13)]))
                    + (0.5 + 0.32) * sum( data.frame(Discharge.scenario [1,
c(6,7,12)]))) # these already have tertiary treatment
                   * 365 * 3785.41 / 1000) # final units is tonnes CO2e/year,
days/year, m3/million gallons, tonnes/kg
do.nothing.habitat <- 0
do.nothing.supply <- 22
do.nothing.n.recovery <- 0
do.nothing.adapt <- 100
do.nothing.slr <- -5 - 1 * scenario[2]
```

```
do.nothing.ease.of.use <- 100
do.nothing.access <- 0
do.nothing.permit <- 100

#wetland.levee
#############

wetland.levee.potential <- colSums (wetland.acre [2:3,])  #area -- sum of
ranking 2 and 3, in acres
MGD.acre <- .042 # MGD/acre for wetland levee
wetland.levee.area <-  apply(cbind (t(Discharge.scenario [1, 3:13]),
wetland.levee.potential * MGD.acre), 1, min)/MGD.acre # units are mgd. Uses
two columns, left column is total flow, right column is flow treatable within
area, 'apply' takes the minimum of these two to find treated flow given area
constraints

#wetland.levee.load = mass removed by treatment, kg/day by levee wetlands
treated.flow.levee <- apply(cbind (t(Discharge.scenario [1, 3:13]),
wetland.levee.potential * MGD.acre), 1, min) # units are mgd. Uses two
columns, left column is total flow, right column is flow treatable within
area, 'apply' takes the minimum of these two to find treated flow given area
constraints, assuming all TN goes through wetland
wetland.levee.load <- sum (treated.flow.levee # per treatment plant
                          * (Discharge.TN.conc.scenario - 2) # mg TN /L
after treatment
                          * 3785411.78 / 1000000) # final unit is kg/day,
L/MG, kg/mg

# wetland.levee.cec = mass SMX removed by treatment

wetland.levee.CEC <- do.nothing.CEC - (sum(treated.flow.levee * (0.9 * SMX) *
153 # assume 90% removal, dry season removal only (5 months), assume
significantly less in wet season
                                          + treated.flow.levee * (0.3 * SMX)
* (365-153)) # assume 30% removal in wet season
                                      * 3785411.78 / 10^12) # final unit is
kg/year, L/MG, ng/kg
wetland.levee.CEC.removal <- (sum(treated.flow.levee * (0.9 * SMX) * 153 #
assume 90% removal, dry season removal only (5 months), assume significantly
less in wet season
                                  + treated.flow.levee * (0.3 * SMX) * (365-
153)) # assume 30% removal in wet season
                              * 3785411.78 / 10^12)
wetland.levee.CEC/do.nothing.CEC

wetland.levee.reliable <- 77 - 0.1 * scenario[2] # assuming wetland
reliability would be very slightly affected (1% decrease) per unit of
reliability.climate.impact
wetland.levee.cost <- sum (wetland.levee.area * 650000) # $/acre
wetland.levee.CO2 <- do.nothing.CO2 + sum (wetland.levee.area *
                                            (((137 + 25 * 4 + 298 * .13) #
in mg/m2/hour, operations
                                              + (1/8 * (137 + 25 * 4 + 298 *
.13))) # mg/m2/hour, construction
                                             * 4046.86 * 24 * 365 / 10 ^ 9))
# final unit is tonnes CO2e/year, m2/acre, hours/day, days/year, tonnes/mg
wetland.levee.habitat <- sum(wetland.levee.area)
```

```
wetland.levee.supply <- do.nothing.supply + 0
wetland.levee.n.recovery <- 0
wetland.levee.adapt <- 52 ### vary from 40-65
wetland.levee.slr <- 8 - 1 * scenario[2]
wetland.levee.ease.of.use <- 100
wetland.levee.access <- 11
wetland.levee.permit <- 30 # vary from 5-60

#wetland.openwater
#################
wetland.openwater.area <- 0.6 * colSums(wetland.area.data [,2:12]) # square
meters, area -- input parameter, assume 60% of ranking 1, 2 and 3 available
#wetland.openwater.load = mass removed by openwater wetland treatment kg TN
/day
openwater.flow <- (Discharge.scenario [1,3:13]
                   * 3785.41178 * 365) # final unit is m3/year, m3/MG,
days/year
# Cout/Cin = (1 + ((59.4 * area/(6.4*flow)))^(-6.4) # per treatment plant,
area is in m2, C in is influent/effluent nitrate concentration - assume all
TN is converted to nitrate (mass/m3), flow is m3/year, assume all flow goes
through wetlands
# this below maximizes removal for open water wetlands (using all area for
each). It also seeks to optimize flow to maximize removal through wetlands of
the largest area. The model shows you should put all the flow through for all
areas to get maximum removal
test.seq<-NULL
discharge.seq<-NULL
Cout.Cin<-NULL
wetland.openwater.load<-NULL # kg/day
wetland.openwater.flow<-NULL
wetland.openwater.flow.proportion<-NULL
Cout.Cin.all<-NULL
for( i in 1:11 ){
  test.seq<- seq(0,as.numeric(openwater.flow[i]),
by=as.numeric(openwater.flow[i]/100))
  discharge.seq <- seq(0,as.numeric(Discharge.scenario[5,i+2]),
by=as.numeric(Discharge.scenario[5,i+2]/100))
  Cout.Cin = (1 + ((59.4 * wetland.openwater.area[i])/(6.4*test.seq)))^(-6.4)
  wetland.openwater.load[i]<-((1-
Cout.Cin)*Discharge.scenario[5,i+2]*(0.01*0:100))[which.max((1-
Cout.Cin)*Discharge.scenario[5,i+2]*(0.01*0:100))]
  wetland.openwater.flow[i]<-(which.max((1-
Cout.Cin)*Discharge.scenario[5,i+2]*(0.01*0:100))-
1)/100*Discharge.scenario[1,i+2]
  wetland.openwater.flow.proportion[i]<-(which.max((1-
Cout.Cin)*Discharge.scenario[5,i+2]*(0.01*0:100))-1)/100
  Cout.Cin.all[i]<-Cout.Cin[101]
}
names(wetland.openwater.load)<-colnames(Discharge.scenario[3:13]) # units is
kg/day removed
names(wetland.openwater.flow)<-colnames(Discharge.scenario[3:13])
names(wetland.openwater.flow.proportion)<-colnames(Discharge.scenario[3:13])
# the result of this crazy for loop is that it treats most N to put all the
flow through at all the sites
# now try solving for A -- basically set Cout/Cin to .1 (90% removal) unless
there is not enough area to do so, in which case maximize area
```

```
# this gives amount TN removed (in kg/day, wetland.openwater.load), as well
as Cout/Cin,  given 90% removal or max wetland area if not enough space for
90% removal
# wetland.openwater.area.TN gives area for 90% nitrogen removal or max space
(if 90% N removal requires more space than availabe), in m2
wetland.openwater.load=NULL
Cout.Cin.all<-NULL
wetland.openwater.area.TN<- NULL

for(i in 1:11){
  Area.90 = (((exp(log(0.1)/-6.4))-1)*6.4*Discharge.scenario[1,i +2]*
3785.41178 * 365)/59.4 # unit is meters2
  openwater.area.actual <- ifelse(Area.90 >
wetland.openwater.area[i],wetland.openwater.area[i], Area.90)
  Cout.Cin = (1 + ((59.4 * openwater.area.actual)/(6.4*Discharge.scenario[1,i
+2]* 3785.41178 * 365)))^(-6.4)
  wetland.openwater.load[i]<-((1-Cout.Cin)*Discharge.scenario[5,i+2]) # units
is kg/day, assume TN is treated (and all ammonia is converted to nitrate
prior to wetland)
  Cout.Cin.all[i]<-Cout.Cin
  wetland.openwater.area.TN [i] <- openwater.area.actual
}
names(wetland.openwater.area.TN)<-colnames(Discharge.scenario[3:13]) # units
are m2
names(wetland.openwater.load)<-colnames(Discharge.scenario[3:13]) # units is
kg/day removed
names(Cout.Cin.all)<-colnames(Discharge.scenario[3:13])
wetland.openwater.load
Cout.Cin.all
wetland.openwater.area.TN

#wetland.openwater.cec = kg of SMX removed by treatment
# A90 = .87/(z * kphoto), hectares/MGD for 90% removal of SMX, z = depth (m)
= .3, kphoto = transformation rate/day, assume .25
z <- .3 # meters
kphoto.summer <- .25 #/day (assume pH is 8, [DOC] = 7 mg/L, latitude is ~40)
kphoto.winter <- .1 #.day (assume pH is 8, [DOC] = 7 mg/L, latitude is ~40)
A90.SMX.summer = .87/(z * kphoto.summer) * 10000 # units is m2/MGD, area
needed to remove 90% of SMX, given summer conditions
A90.SMX.winter = .87/(z * kphoto.winter) * 10000 # units is m2/MGD, area
needed to remove 90% of SMX, given winter conditions
# assume SMX removal scales linearly with area, assume 6 months of summer and
6 months of winter conditions
A90.SMX = mean(c(A90.SMX.summer,A90.SMX.winter)) # units is m2/MGD
# wetland.openwater.cec = total amount of SMX removed by treatment, final
unit is kg/year
wetland.openwater.cec.all <- data.frame(NULL)

for (i in 1:11){
  wetland.openwater.cec.all [1,i]<- (wetland.openwater.area.TN[i]/(A90.SMX *
Discharge.scenario [1, i +2] ))*.9*do.nothing.CEC.WWTPs[i]
}
wetland.openwater.CEC <- do.nothing.CEC - sum(wetland.openwater.cec.all)

wetland.openwater.CEC/do.nothing.CEC # this is interesting...basically you
need about twice as much space to treat SMX to 90% removal than nitrogen...
```

```
# assume SMX removal scales linearly with area
wetland.openwater.reliable <- 77 - 0.3 * scenario[2] # assuming wetland
reliability would be slightly affected (3% decrease) per unit of
climate.impact
wetland.openwater.cost <- 200000 * sum(wetland.openwater.area.TN) / 4046.86 #
final unit is $, m2/acre
wetland.openwater.CO2 <- do.nothing.CO2 + ((sum(wetland.openwater.area.TN) *
                                              137 + (25 * 4) + (298 * .13) #
in mg/m2/hour
                                            + (1/8)  * (137 + (25 * 4) + (298
* .13))) # in mg/m2/hour
                                          /(10^9) * 24 * 365) # final unit
is tonnes CO2e/year, mg/tonne, hr/day, days/year
wetland.openwater.habitat <- sum(wetland.openwater.area.TN/10000) /2 # assume
this is half as good as sub-surface flow for habitat
wetland.openwater.supply <- 0 + do.nothing.supply
wetland.openwater.n.recovery <- 0
wetland.openwater.adapt <- 42 ### vary from 30-55
wetland.openwater.slr <- 5 - 1 * scenario[2]
wetland.openwater.ease.of.use <- 100
wetland.openwater.access <- 11  ## ask Felix, should by number of treatment
plants
wetland.openwater.permit <- 30 # vary from 1-60

#recycle.irrig
##############

recycle.irrig.amount<-  c(0, (1/6) * Discharge.scenario [1,4], 0, 10, 45, 10,
0, 2, 5, 10, 0) #amount of water recycled for irrigation, MGD per treatment
plant
names (recycle.irrig.amount) <- colnames(Discharge.scenario [3:13])
recycle.irrig.load <- recycle.irrig.amount * Discharge.TN.conc.scenario /
(10^6) * 3785411.78  # final unit is kg/day, mg/kg, L/million gallons
recycle.irrig.CEC <- do.nothing.CEC - (sum(recycle.irrig.amount) * SMX # ng/L
                                       * 3785411.78 * 365 / (10^12)) # final
unit is kg/year, liters/ million gallons, days/year, ng/kg. Assume all gets
recycled in the wet season also
recycle.irrig.CEC.removal <- (sum(recycle.irrig.amount) * SMX # ng/L
                              * 3785411.78 * 365 / (10^12))
recycle.irrig.reliable <- 96 - .5 * scenario [2]
recycle.irrig.cost <- ((sum(recycle.irrig.amount)
                        * 930) # $/AF/year
                       * 3.06888785 * 365 * 30) # final unit is $ over 30
years
recycle.irrig.CO2 <- do.nothing.CO2 + (sum(recycle.irrig.amount) * 1023 # g
CO2e/m3
                                       * 3785.41178 * 365 / (10^6)) # final
unit is final unit is tonnes CO2e/year, m3/million gallons, days/year,
g/tonne

recycle.irrig.habitat <- 0
recycle.irrig.supply <- sum(recycle.irrig.amount) + do.nothing.supply
recycle.irrig.n.recovery <- sum(recycle.irrig.load) * 365
recycle.irrig.adapt <- 45 ### vary from 30-60
recycle.irrig.slr <- -3 - 1 * scenario[2]
recycle.irrig.ease.of.use <- 100
recycle.irrig.access <- 0
```

```
recycle.irrig.permit <- 70 # vary from 55-85

# recycle.brineline
#################
names (Discharge.scenario)
recycle.ipr <- c(.25  * Discharge.scenario [1,3], 0, .25 * Discharge.scenario
[1,5], .25 * Discharge.scenario [1,6], 35, 0, 0, 20, .25 * Discharge.scenario
[1,11], .25 * Discharge.scenario [1,12], .25 * Discharge.scenario [1,13]) #
in MGD, vector of amounts from wwtps with IPR
recycle.dpr <- c(0,0,0,0,8,0,0,0,0,0,0) # MGD
recycle.brineline.amount  <- recycle.ipr + recycle.dpr #amount of water
recycled for potable reuse, MGD
recycle.brine <- .2 * sum (recycle.brineline.amount) # amount of brine, MGD,
assume 80% efficiency of RO
recycle.brineline.load <- (sum(recycle.brineline.amount *
Discharge.TN.conc.scenario)
                           * 3785411.78 / (10^6)) # final unit is kg/day,
L/million gallons, mg/kg
recycle.brineline.CEC <- do.nothing.CEC - (sum(recycle.brineline.amount) *
SMX #ng/L
                                            * 3785411.78 * 365 / (10^12)) #
final unit is kg/year
recycle.brineline.reliable <- 98 - .5 * scenario [2] # mean of normal
distribution with SD .01
pumping.energy <- ((1/2655220) # kwh/ft-lbs of lift assume weight of brine =
weight of water
                   * recycle.brine #units is MGD
                   * 2000 # ft, height over the peninsula
                   * (8.34 * 10^6) *365 / 1000) # final unit is Mwh/year
pumping.energy.CO2 <- (pumping.energy * 320896  # unit here after
multiplication is g CO2 e/ million gallons (conversion from
west.berkeley.edu)
                       * recycle.brine / (10^6) * 365) # final unit is tonnes
CO2 e/year, g/tonne, days/year
# run through WEST (west.berkeley.edu), get 11866708406 g CO23/million
gallons
pipe.length <- 36960 # ft across peninsula (roughly Millbrae to Pacifica)
pipe.CO2 <- (pipe.length * 358  # unit here after multiplication is g CO2 e/
million gallons (conversion from west.berkeley.edu)
             * recycle.brine / (10^6) * 365) #final unit is tonnes CO2
e/year, g/tonne, days/year
pumping.CO2 <- pumping.energy.CO2 + pipe.CO2 # unit is tonnes CO2e/year
# unit is CO2e/year (need CO2 e/kwh in CA)
recycle.brineline.CO2 <- do.nothing.CO2 + ((sum(recycle.brineline.amount) *
(438 + 25 * (.88)) # kg CO2e/466 m3
                                             * 3785.41178 * 365 / 466 / (10^3)
# m3/million gallons, days/year, m3/kg, kg/ tonne
                                             + pumping.CO2)) # final unit is
tonnes CO2 e/year
pumping.cost <- ((1/2655220) # kwh/ft-lbs of lift assume weight of brine =
weight of water
                 * 0.2 # $/kwh for energy
                 * recycle.brine #units is MGD
                 * 2000 # ft, height over the peninsula
                 * (8.34 * 10^6) *365*30) # final unit is $ for 30 years,
pounds/million gallons, days/year, years
```

```
recycle.brineline.cost <- ((sum(recycle.ipr) * 3.06888785 * 365 * 30  # 
AF/million gallons, days/year, years
                            * (700 + mean(c( 120, 750, 1250)))) # $/AF, 
advanced treatment plus conveyance to drinking water
                           + (sum(recycle.dpr) * 3.06888785 * 365 * 30 # 
AF/million gallons, days/year, years
                              * (700 + 120)) # $/AF, advanced treatment plus 
conveyance to drinking water
                           + (recycle.brine * 115 * 3.06888785 * 365 * 30 ) # 
final unit is $/AF, cost of pipe for amount of water, AF/million gallons, 
days/year, year
                           + 2000000 * (pipe.length/5280) # cost of brineline 
pipe per distance, ft/mile
                           +  pumping.cost) # final unit is $ for 30 years of 
operation
recycle.brineline.habitat <- 0
recycle.brineline.supply <- sum(recycle.brineline.amount) + do.nothing.supply
recycle.brineline.n.recovery <- 0
recycle.brineline.adapt <- 50 ### vary from 40-60
recycle.brineline.slr <- -5 - 1 * scenario[2]
recycle.brineline.ease.of.use <- 65
recycle.brineline.access <- 0
recycle.brineline.permit <- 40 # vary from 30-50

#urine.early
##########

# proportion of population that is early adopters (tech and ed industries)
Early.adopt <- .039
# grams of N excreted in urine
N.excreted <- 10.5 # g N/person/day
# Proportion of a person's urine recovered by urine source separation
U.recovery.toilet <- .5
# Proportion of N recovered by ion exchange resin
N.recovery.resin <- .9
toilet.use <- .5 # percent of the time that people with urine-separating 
toilets use them
population <-  scenario [1] * c(30000, 957000, 22850, 217330, 1498700, 0.5 * 
(143100 + 156500), 10000, 670400, 110000, 148000, 200600) # shown in order of 
WWTPs in Discharge.scenario.
names(population)<- colnames(Discharge.scenario[3:13])
urine.early <- Early.adopt * sum(population) + ifelse((scenario 
[1])>1,(scenario [1]-1)*(sum(population)/scenario[1]),0) # if population 
grows, it's early adopters + whatever population growth. If population 
declines, it's just early adopters
urine.early.load <- (urine.early *  N.excreted *  N.recovery.resin * 
U.recovery.toilet * toilet.use
                     / 1000 ) # final unit is kg/day, g/kg

total.CEC <- sum(data.frame(Discharge.scenario [1, c(3:5, 8:11, 13)] * SMX,
                            Discharge.scenario [1, c(6,7,12)] * .85 * SMX)* 
                   365 * 10^6 /.26417 / 10^9 / 10^3) # final unit is kg/year 
SMX, days/year, gal/day, L/gal, g/ng, kg/g
urine.early.CEC.removal <- urine.early/sum(population) * total.CEC * .84
urine.early.CEC <- do.nothing.CEC - urine.early.CEC.removal # proportion of 
SMX excreted in urine vs. feces, final unit is kg/year of SMX removed
urine.early.reliable <- 66 # +/- 20%
```

```
urine.early.cost <- (26 # $/m3 of urine
                     * urine.early * 1.4 # L urine excreted/person/day
                     * toilet.use
                     * 365 * 30 / 1000) # final units is $ over 30 years,
days/year, yrs, L/m3
urine.early.CO2 <- do.nothing.CO2 + (urine.early * 1.4 # L urine
excreted/person/day
                                     * toilet.use
                                     * 5.5 # kg CO2 e/m3 urine treated
                                     / 1000 / 1000) # final units is tonnes
CO2e/year, kg/tonne, L/m3
urine.early.habitat <- 0
urine.early.supply <- 0 + do.nothing.supply
urine.early.n.recovery <- urine.early.load * 365
urine.early.adapt <- 85 ### vary from 75-95
urine.early.slr <- 0
urine.early.ease.of.use <- 35 ## vary from 25-45
urine.early.access <- 0
urine.early.permit <- 40 # vary from 30-50


#urine.incentives
###############

urine.incentives <- (0.3 * sum(population)) + ifelse((scenario
[1])>1,(scenario [1]-1)*(sum(population)/scenario[1]),0) # 30% of current
population + all population growth
#urine.incentives.load = kg TN/day removed by urine source separation
toilet.use.incentives <- .75 # percent of the time that people with urine-
separating toilets use them
urine.incentives.load <- (urine.incentives *  N.excreted *  N.recovery.resin
* U.recovery.toilet * toilet.use.incentives
                          / 1000 ) # final unit is kg/day, g/kg
urine.incentives.CEC.removal <- (urine.incentives/sum(population) * total.CEC
* .84)
urine.incentives.CEC <- do.nothing.CEC - urine.incentives.CEC.removal #
proportion of SMX excreted in urine vs. feces, final unit is kg/year of SMX
removed
urine.incentives.reliable <- 76 # +/- 20%
incentive1 <- 3000 # initial retrofit incentive
incentive2 <- (2 * 52 * 30) #  weekly reward for turning in cartridge for 30
year lifecycle, $ reward, weeks/year, years
household <- 2.7
outreach <- 200000 * 11 # ask Felix about this... 11 is number of
participating utilities
urine.incentives.cost <- (((26 # $/m3 of urine
                            * urine.incentives * 1.4 # L urine
excreted/person/day
                            * toilet.use.incentives
                            * 365 * 30 / 1000) # final units is $ over 30
years, days/year, yrs, L/m3
                           + (incentive1 * (0.3 *
(sum(population)/scenario[1]) / household )) # $/household to install toilet
for all existing population
                           + (incentive2 * urine.incentives/household) #
people per household, initial incentive only for retrofits, not new housing
stock
```

```
                           + outreach)) # final unit is $
urine.incentives.CO2 <- do.nothing.CO2 + (urine.incentives * 1.4 # L urine
excreted/person/day
                                          * toilet.use.incentives
                                          * 5.5 # kg CO2 e/m3 urine treated
                                          / 1000 / 1000) # final units is
tonnes CO2e/year, kg/tonne, L/m3)
urine.incentives.habitat <- 0
urine.incentives.supply <- 0 + do.nothing.supply
urine.incentives.n.recovery <- urine.incentives.load * 365
urine.incentives.adapt <- 55 ### vary from 45-65
urine.incentives.slr <- 0
urine.incentives.ease.of.use <- 35 ## vary from 25-45
urine.incentives.access <- 0
urine.incentives.permit <- 40 # vary from 30-50

# opt
####
opt <- # binary yes or no
  opt.load <- sum(Discharge.scenario [5,3:13]) * .1 # assuming optimization
removes 10%, final unit is kg/day
opt.CEC <- do.nothing.CEC # CECs removed by optimization is 0
opt.cost <- sum(Discharge.scenario[1, 3:13]) * 1000000 * 0.5 #assume $0.5/gpd
(estimated from HDR draft report).
opt.reliable<- 98 # normal with sd .1
opt.CO2 <- do.nothing.CO2 + 0
opt.habitat <- 0
opt.supply <- 0 + do.nothing.supply
opt.n.recovery <- 0
opt.adapt <- 75 ### vary from 60-90
opt.slr <- -5 -1 * scenario [2]
opt.ease.of.use <- 100
opt.access <- 0
opt.permit <- 90 # vary from 80-100


# Level2
Level2 <- # binary yes or no
  #Level2.load = amount TN (kg/day) removed by Level 2 treatment
  Level2.load <- (sum(Discharge.scenario [5,3:13]) - (sum(Discharge.scenario
[1,3:13]) * 15 # mg/L
                                                        * 3785411.78 / (10^6)))
# final units is kg/day, L/million gallons, mg/kg
#Level2.CEC = additonal amount SMX (kg/year) removed by Level 2 treatment,
assumes no additional removal from plants that already have tertiary
treatment
Level2.CEC <- do.nothing.CEC - (sum(data.frame(Discharge.scenario [1, c(3:5,
8:11, 13)])) * .15 * SMX
                              * 3785411.78 / (10^12) * 365) # final unit is
kg SMX/year, L/million gallons, ng/kg, days/year
Level2.cost <- sum(Discharge.scenario[1, 3:13]) * 1000000 * 7.5 # $7.5/gpd
dry season flow-weighted average from HDR draft report to BACWA
Level2.reliable <- 98 # normal with SD .1
Level2.CO2 <- do.nothing.CO2 + (sum(data.frame(Discharge.scenario [1, c(3:5,
8:11, 13)])) * 0.32 # kg CO2 e/m3, for all the plants that don't already have
tertiary treatment
```

```
                                * 3785.41178 * 365 / 1000) # final unit is
tonnes CO2 e/year, m3/million gallons, days/year, kg/tonne
Level2.habitat <- 0
Level2.supply <- 0 + do.nothing.supply
Level2.n.recovery <- 0
Level2.adapt <- 55 ### vary from 30-75
Level2.slr <- -5 -1 * scenario [2]
Level2.ease.of.use <- 100
Level2.access <- 0
Level2.permit <- 90 # vary from 80-100

# Level3
Level3 <- # binary yes or no
  #Level3.load = amount TN (kg/day) removed by Level 3 treatment
  Level3.load <- (sum(Discharge.scenario [5,3:13]) - (sum(Discharge.scenario
[1,3:13]) * 6 # mg/L
                                                      * 3785411.78 / (10^6)))
# final units is kg/day, L/million gallons, mg/kg
Level3.CEC <- do.nothing.CEC - (sum(data.frame(Discharge.scenario [1, c(3:5,
8:11, 13)])) * .15 * SMX
                                * 3785411.78 / (10^12) * 365) # final unit is
kg SMX/year, L/million gallons, ng/kg, days/year
Level3.cost <- sum(Discharge.scenario[1, 3:13]) * 1000000 * 9.8 # $9.8/gpd
dry season flow-weighted average from HDR draft report to BACWA
Level3.reliable <- 98 # normal with SD .01
Level3.CO2 <- do.nothing.CO2 + (sum(data.frame(Discharge.scenario [1, 3:13]))
* 0.8 # kg CO2 e/m3, for all the plants that don't already have tertiary
treatment
                                * 3785.41178 * 365 / 1000) # final unit is
tonnes CO2 e/year, m3/million gallons, days/year, kg/tonne
Level3.habitat <- 0
Level3.supply <- 0 + do.nothing.supply
Level3.n.recovery <- 0
Level3.adapt <- 10 ### vary from 5-15
Level3.slr <- -5 -1 * scenario [2]
Level3.ease.of.use <- 100
Level3.access <- 0
Level3.permit <- 90 # vary from 80-100
```

Calculate percent change in nutrient loading from current conditions per option.

```
# do nothing
do.nothing.load.change <- (do.nothing.load - sum((Discharge.data.current
[5,3:13])))/sum(Discharge.data.current [5,3:13]) * 100
# wetland levee
wetland.levee.load.change <- ((do.nothing.load - wetland.levee.load) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
#wetland openwater
wetland.openwater.load.change <- ((do.nothing.load -
sum(wetland.openwater.load)) - sum(Discharge.data.current [5,3:13])) /
sum(Discharge.data.current [5,3:13]) * 100
#recycle irrigation
recycle.irrig.load.change <- ((do.nothing.load - sum(recycle.irrig.load)) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
```

```
# recycle brineline
recycle.brineline.load.change <- ((do.nothing.load - recycle.brineline.load)
- sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current
[5,3:13]) * 100
# urine early
urine.early.load.change <- ((do.nothing.load - urine.early.load) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
# urine incentives ### hmm, something looks wrong here...should be much more
reduction because is supposed to be all new growth + 30% of existing...
urine.incentives.load.change <- ((do.nothing.load - urine.incentives.load) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
# opt
opt.load.change <- ((do.nothing.load - sum(opt.load)) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
# level 2
Level2.load.change <- ((do.nothing.load - Level2.load) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
# level 3
Level3.load.change <- ((do.nothing.load - Level3.load) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
```

Create objects for the different parameter vectors.

```
load.change.par <-
c(do.nothing.load.change,wetland.levee.load.change,wetland.openwater.load.cha
nge,
recycle.irrig.load.change,recycle.brineline.load.change,urine.early.load.chan
ge, urine.incentives.load.change, opt.load.change,Level2.load.change,
Level3.load.change)
names (load.change.par) <- options

#translating from load.change to wq
###################################

## very basic, assuming linear fit from data points
#data points if no threshold
percent.change <- c(0, -100, 100)
wq.graph <- c(10, 5, 95)

plot(percent.change, wq.graph)
abline(lm(wq.graph ~ percent.change))
regmodel=lm(wq.graph ~percent.change) #fit a regression model
summary(regmodel) #get results from fitting the regression model

wq.par<-NULL
if (scenario [3] == 0) wq.par <- regmodel$coefficients[2]* load.change.par +
regmodel$coefficients[1]

#wq.par
# data points if threshold
# very basic, assuming linear fit based on elicitation from SFEI
```

```
percent.change.thresh <- c(33, 60, -13, -35, -35, -20, -20, -60, -60, -75)
wq.graph.thresh <- c(75, 85, 70, 35, 35, 40, 40, 30, 30, 10)
plot (percent.change.thresh, wq.graph.thresh)
abline(lm(wq.graph.thresh~percent.change.thresh))
regmodel.thresh=lm(wq.graph.thresh~percent.change.thresh) #fit a regression
model
summary(regmodel.thresh) #get results from fitting the regression model

if (scenario [3] == 1) wq.par <- regmodel.thresh$coefficients[2]*
load.change.par + regmodel.thresh$coefficients[1] #simply linear with y =
mx+b
wq.par
for (i in 1:10){
  if (wq.par [i] <= 5) wq.par[i] <- 5
  if (wq.par [i] >=95) wq.par[i] <- 95

}
wq.par

#habitat
habitat.par <- c(do.nothing.habitat, wetland.levee.habitat,
wetland.openwater.habitat, recycle.irrig.habitat, recycle.brineline.habitat,
urine.early.habitat, urine.incentives.habitat, opt.habitat, Level2.habitat,
Level3.habitat)
names (habitat.par) <- options
# water supply
supply.par <- c(do.nothing.supply, wetland.levee.supply,
wetland.openwater.supply, recycle.irrig.supply, recycle.brineline.supply,
urine.early.supply, urine.incentives.supply, opt.supply, Level2.supply,
Level3.supply)
names (supply.par) <- options
# nutrient recovery
n.recovery.par <- c(do.nothing.n.recovery, wetland.levee.n.recovery,
wetland.openwater.n.recovery, recycle.irrig.n.recovery,
recycle.brineline.n.recovery, urine.early.n.recovery,
urine.incentives.n.recovery, opt.n.recovery, Level2.n.recovery,
Level3.n.recovery)
names (n.recovery.par) <- options
# CECs - this is amount that gets through after treatment
CEC.par <- c(do.nothing.CEC, wetland.levee.CEC, wetland.openwater.CEC,
recycle.irrig.CEC, recycle.brineline.CEC, urine.early.CEC,
urine.incentives.CEC, opt.CEC, Level2.CEC, Level3.CEC)
names (CEC.par) <- options
# ease of adaptation
adapt.par <- c(do.nothing.adapt, wetland.levee.adapt,
wetland.openwater.adapt, recycle.irrig.adapt, recycle.brineline.adapt,
urine.early.adapt, urine.incentives.adapt, opt.adapt, Level2.adapt,
Level3.adapt)
names (adapt.par) <- options
# resilience to sea level rise
slr.par <- c(do.nothing.slr, wetland.levee.slr, wetland.openwater.slr,
recycle.irrig.slr, recycle.brineline.slr, urine.early.slr,
urine.incentives.slr, opt.slr, Level2.slr, Level3.slr)
names (slr.par) <- options
# GHG emissions
```

```
CO2.par <- c(do.nothing.CO2, wetland.levee.CO2, wetland.openwater.CO2,
recycle.irrig.CO2, recycle.brineline.CO2, urine.early.CO2,
urine.incentives.CO2, opt.CO2, Level2.CO2, Level3.CO2)
names (CO2.par) <- options
# ease of use
ease.of.use.par <- c(do.nothing.ease.of.use, wetland.levee.ease.of.use,
wetland.openwater.ease.of.use, recycle.irrig.ease.of.use,
recycle.brineline.ease.of.use, urine.early.ease.of.use,
urine.incentives.ease.of.use, opt.ease.of.use, Level2.ease.of.use,
Level3.ease.of.use)
names (ease.of.use.par) <- options
# shoreline access
access.par <- c(do.nothing.access, wetland.levee.access,
wetland.openwater.access, recycle.irrig.access, recycle.brineline.access,
urine.early.access, urine.incentives.access, opt.access, Level2.access,
Level3.access)
names (access.par) <- options
# permitting
permit.par <- c(do.nothing.permit, wetland.levee.permit,
wetland.openwater.permit, recycle.irrig.permit, recycle.brineline.permit,
urine.early.permit, urine.incentives.permit, opt.permit, Level2.permit,
Level3.permit)
names (permit.par) <- options
#reliability
reliable.par <- c(do.nothing.reliable, wetland.levee.reliable,
wetland.openwater.reliable, recycle.irrig.reliable,
recycle.brineline.reliable, urine.early.reliable, urine.incentives.reliable,
opt.reliable, Level2.reliable, Level3.reliable)
names (reliable.par) <- options
#costs
cost.par <- c(do.nothing.cost, wetland.levee.cost, wetland.openwater.cost,
recycle.irrig.cost, recycle.brineline.cost, urine.early.cost,
urine.incentives.cost, opt.cost, Level2.cost, Level3.cost)
names (cost.par) <- options
# cost per capita
cost.per.cap <- cost.par/sum(population)
names(cost.per.cap) <- options

criteria.calcs<- data.frame(cbind(wq.par, habitat.par, supply.par,
n.recovery.par, CEC.par, adapt.par, slr.par, CO2.par, ease.of.use.par,
access.par, permit.par, reliable.par, cost.par))
```

Calculate and plot cost-efficiency of each option.

```
# cost efficiency (cost.par/nitrogen removal)
n.removal <- 365*30* c(0, wetland.levee.load, sum(wetland.openwater.load),
sum(recycle.irrig.load), recycle.brineline.load, urine.early.load,
urine.incentives.load, opt.load, Level2.load, Level3.load)

cost.efficiency <- cost.par / n.removal
cost.efficiency2 <- n.removal/cost.par
cost.efficiency3 <- n.removal[c(1:4, 6:10)]/cost.par[c(1:4, 6:10)] # no
brineline
cost.efficiency4 <- cost.par[c(1:4, 6:10)]/n.removal[c(1:4, 6:10)] #no
brineline
names(cost.efficiency2) <- options
```

```
names(cost.efficiency3) <- options.no.brine
names(cost.efficiency4) <- options.no.brine

par(mfrow = c(1,1))
par(mar=c(10,4.1,5,2),
    oma = c(0,0,0,0),
    xpd = T)
barplot(cost.efficiency, main = "Cost efficiency ($/kg TN removal)", las = 2)
barplot(cost.efficiency2, main = "Cost efficiency (kg TN removal/$)", las =
2, col = brewer.pal(10, "Paired"))
barplot(cost.efficiency3, main = "Cost efficiency (kg TN removal/$)", las =
2, col = brewer.pal(9, "Paired"), ylim = c(0,.5))
barplot(cost.efficiency4, main = "Cost efficiency ($/kg TN removal)", las =
2, col = brewer.pal(9, "Paired"), ylim = c(0,70))
```

Set parameters for modeling uncertainty for each option.

```
uncertainty.distributions<- read.csv("uncertainty_distributions.csv",
row.names = 1)

#do nothing
###########

#do.nothing.load
do.nothing.load.change.par1<- .8  * do.nothing.load.change
do.nothing.load.change.par2<-1.2 * do.nothing.load.change

#do.nothing.wq
do.nothing.wq.par1<- .8 * wq.par[1]
do.nothing.wq.par2<- 1.2 * wq.par[1]

# here do.nothing.CEC is actually how much SMX is produced
#do.nothing.CEC
do.nothing.CEC.par1<- .8 * sum(do.nothing.CEC)
do.nothing.CEC.par2<-  1.2 * sum(do.nothing.CEC)

#do.nothing.reliable
do.nothing.reliable.par1<- do.nothing.reliable
do.nothing.reliable.par2<- do.nothing.reliable

#do.nothing.cost
do.nothing.cost.par1<- do.nothing.cost
do.nothing.cost.par2<- do.nothing.cost

#do.nothing.CO2
do.nothing.CO2.par1<- .7 * do.nothing.CO2
do.nothing.CO2.par2<- 1.3 * do.nothing.CO2

#do.nothing.habitat
do.nothing.habitat.par1<- do.nothing.habitat
do.nothing.habitat.par2<- do.nothing.habitat

#do.nothing.supply
do.nothing.supply.par1<- do.nothing.supply
do.nothing.supply.par2<- do.nothing.supply
```

```
#do.nothing.n.recovery
do.nothing.n.recovery.par1<- do.nothing.n.recovery
do.nothing.n.recovery.par2<- do.nothing.n.recovery

#do.nothing.adapt
do.nothing.adapt.par1<- do.nothing.adapt
do.nothing.adapt.par2<- do.nothing.adapt

#do.nothing.slr
do.nothing.slr.par1<- do.nothing.slr
do.nothing.slr.par2<- do.nothing.slr

#do.nothing.ease.of.use
do.nothing.ease.of.use.par1<- do.nothing.ease.of.use
do.nothing.ease.of.use.par2<- do.nothing.ease.of.use

#do.nothing.access
do.nothing.access.par1<- do.nothing.access
do.nothing.access.par2<- do.nothing.access

#do.nothing.permit
do.nothing.permit.par1<-do.nothing.permit
do.nothing.permit.par2<- do.nothing.permit

# WETLAND LEVEE
###############
wetland.levee.dist<- c("Minmax", "Minmax", "Determined", "Determined", "")
MGD.acre <- (.042) # MGD/acre for wetland levee
treated.flow.levee <- apply(cbind (t(Discharge.scenario [1, 3:13]),
wetland.levee.potential * MGD.acre), 1, min) # units are mgd. Uses two
columns, left column is total flow, right column is flow treatable within
area, 'apply' takes the minimum of these two to find treated flow given area
constraints, assuming all TN goes through wetland
wetland.levee.load <- sum (treated.flow.levee # per treatment plant
                           * (Discharge.TN.conc.scenario - 2) # mg TN /L
after treatment
                           * 3785411.78 / 1000000) # final unit is kg/day,
L/MG, kg/mg

MGD.acre.8 <- .8 * (.042) # MGD/acre for wetland levee
treated.flow.levee.8 <- apply(cbind (t(Discharge.scenario [1, 3:13]),
wetland.levee.potential * MGD.acre.8), 1, min) # units are mgd. Uses two
columns, left column is total flow, right column is flow treatable within
area, 'apply' takes the minimum of these two to find treated flow given area
constraints, assuming all TN goes through wetland
wetland.levee.load.8 <- sum (treated.flow.levee.8 # per treatment plant
                             * (Discharge.TN.conc.scenario - 2) # mg TN /L
after treatment
                             * 3785411.78 / 1000000) # final unit is kg/day,
L/MG, kg/mg
MGD.acre1.2 <- 1.2 * (.042) # MGD/acre for wetland levee
treated.flow.levee1.2 <- apply(cbind (t(Discharge.scenario [1, 3:13]),
wetland.levee.potential * MGD.acre1.2), 1, min) # units are mgd. Uses two
columns, left column is total flow, right column is flow treatable within
area, 'apply' takes the minimum of these two to find treated flow given area
constraints, assuming all TN goes through wetland
```

```
wetland.levee.load1.2 <- sum (treated.flow.levee1.2 # per treatment plant
                              * (Discharge.TN.conc.scenario - 2) # mg TN /L
after treatment
                              * 3785411.78 / 1000000) # final unit is kg/day,
L/MG, kg/mg

wetland.levee.load.change <- ((do.nothing.load - wetland.levee.load) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
wetland.levee.load.change.8 <- ((do.nothing.load - wetland.levee.load.8) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
wetland.levee.load.change1.2 <- ((do.nothing.load - wetland.levee.load1.2) -
sum(Discharge.data.current [5,3:13])) / sum(Discharge.data.current [5,3:13])
* 100
levee.load.change.variation <-
c(wetland.levee.load.change,wetland.levee.load.change.8,wetland.levee.load.ch
ange1.2)
names(levee.load.change.variation)<- c("0.042 MGD/acre", "0.0336 MGD/acre",
"0.0504 MGD/acre")
wetland.levee.load.change.8/wetland.levee.load.change
wetland.levee.load.change1.2/wetland.levee.load.change

# vary from - 20% to + 17 % based on +/- 20% change to area needed for
treatment
#wetland.levee.load.change: +/- 20%, uniform distribution
wetland.levee.load.change.par1 <- .8 * wetland.levee.load.change
wetland.levee.load.change.par2 <- 1.17* wetland.levee.load.change

#wetland levee.wq
wetland.levee.wq.par1 <- .8 * wq.par[2]
wetland.levee.wq.par2 <- 1.2 * wq.par[2]

# wetland.levee.cec
wetland.levee.CEC.par1 <- do.nothing.CEC.par1 - (1.2 *
wetland.levee.CEC.removal)
wetland.levee.CEC.par2 <- do.nothing.CEC.par2 - (.8 *
wetland.levee.CEC.removal)

#wetland.levee.reliable
wetland.levee.reliable.par1 <- .8 * wetland.levee.reliable
wetland.levee.reliable1.par2 <- 1.2 * wetland.levee.reliable

#wetland.levee.cost
wetland.levee.cost.par1 <- sum (wetland.levee.area * 100000) # $/acre
wetland.levee.cost.par2 <- sum (wetland.levee.area * 1000000) # $/acre

#wetland.levee.CO2
wetland.levee.CO2.par1 <- .8 * wetland.levee.CO2
wetland.levee.CO2.par2 <- 1.2 * wetland.levee.CO2

#wetland.levee.habitat
wetland.levee.habitat.par1 <- .75 * wetland.levee.habitat
wetland.levee.habitat.par2 <- wetland.levee.habitat

#wetland.levee.supply
wetland.levee.supply.par1<- wetland.levee.supply
```

```
wetland.levee.supply.par2<- wetland.levee.supply

#wetland.levee.n.recovery
wetland.levee.n.recovery.par1<- wetland.levee.n.recovery
wetland.levee.n.recovery.par2<- wetland.levee.n.recovery

#wetland.levee.adapt
wetland.levee.adapt.par1 <- 40
wetland.levee.adapt.par2 <- 65

#wetland.levee.slr
wetland.levee.slr.par1<- wetland.levee.slr
wetland.levee.slr.par2<- wetland.levee.slr

#wetland.levee.ease.of.use
wetland.levee.ease.of.use.par1<- wetland.levee.ease.of.use
wetland.levee.ease.of.use.par2<- wetland.levee.ease.of.use

#wetland.levee.access
wetland.levee.access.par1 <- 6
wetland.levee.access.par2 <- 11

#wetland.levee.permit
wetland.levee.permit.par1 <- 5
wetland.levee.permit.par2 <- 60

# WETLAND OPENWATER
###################

#wetland.openwater.load.change
wetland.openwater.load.change.par1 <- 0.8 * wetland.openwater.load.change
wetland.openwater.load.change.par2 <- 1.2 * wetland.openwater.load.change

#wetland.openwater.wq
wetland.openwater.wq.par1 <- .8 * wq.par [3]
wetland.openwater.wq.par2 <- 1.2 * wq.par [3]

#wetland.openwater.CEC
wetland.openwater.CEC.par1 <- .8 * wetland.openwater.CEC
wetland.openwater.CEC.par2 <- 1.2 * wetland.openwater.CEC

#wetland.openwater.reliable
wetland.openwater.reliable.par1 <- .8 * wetland.openwater.reliable
wetland.openwater.reliable.par2 <- 1.2 * wetland.openwater.reliable

#wetland.openwater.cost
wetland.openwater.cost.par1 <- .8 * wetland.openwater.cost
wetland.openwater.cost.par2 <- 1.2 * wetland.openwater.cost + 300000 *
sum(wetland.openwater.area.TN/4046.86) # conversion for m2 to acres, extra
$300,000/acre for land costs on top end

#wetland.openwater.CO2: model with +/- 20%
wetland.openwater.CO2.par1 <- .8 * wetland.openwater.CO2
wetland.openwater.CO2.par2 <- 1.2 * wetland.openwater.CO2

#wetland.openwater.habitat
wetland.openwater.habitat.par1 <- .4 * sum(wetland.openwater.area.TN/10000)
```

```
wetland.openwater.habitat.par2 <- .7 * sum(wetland.openwater.area.TN/10000)

#wetland.openwater.supply
wetland.openwater.supply.par1<- wetland.openwater.supply
wetland.openwater.supply.par2<- wetland.openwater.supply

#wetland.openwater.n.recovery
wetland.openwater.n.recovery.par1<- wetland.openwater.n.recovery
wetland.openwater.n.recovery.par2<- wetland.openwater.n.recovery

#wetland.openwater.adapt
wetland.openwater.adapt.par1 <- 30
wetland.openwater.adapt.par2 <- 55

#wetland.openwater.slr
wetland.openwater.slr.par1<- wetland.openwater.slr
wetland.openwater.slr.par2<- wetland.openwater.slr

#wetland.openwater.ease.of.use
wetland.openwater.ease.of.use.par1<- wetland.openwater.ease.of.use
wetland.openwater.ease.of.use.par2<- wetland.openwater.ease.of.use

#wetland.openwater.access
wetland.openwater.access.par1 <- 6
wetland.openwater.access.par2 <- 11

#wetland.openwater.permit
wetland.openwater.permit.par1 <- 1
wetland.openwater.permit.par2 <- 60

#RECYCLE IRRIGATION
###################

#recycle.irrig.load.change
recycle.irrig.load.change.par1 <- .8 * recycle.irrig.load.change
recycle.irrig.load.change.par2 <- 1.2 * recycle.irrig.load.change

#recycle.irrig.wq
recycle.irrig.wq.par1<- .8 * wq.par[4]
recycle.irrig.wq.par2<- 1.2 * wq.par[4]

# recycle.irrig.CEC
recycle.irrig.CEC.par1 <- do.nothing.CEC.par1 - ( .8 *
recycle.irrig.CEC.removal)
recycle.irrig.CEC.par2 <- do.nothing.CEC.par2 - (1.2 *
recycle.irrig.CEC.removal)

# recycle.irrig.reliable
recycle.irrig.reliable.par1 <- .8* recycle.irrig.reliable
recycle.irrig.reliable.par2 <- 1.2 * recycle.irrig.reliable

irrig.cost.Bay.Area <- c(522.88, 493.12, 1379.31, 242.42, 1200.87, 387.0,
1972.39, 1041.67, 258.75, 601.31, 130.03, 963.39, 665.00, 797.78, 645.99,
133.33, 4385.96) # $/ AF
irrig.scaled <- (irrig.cost.Bay.Area-
min(irrig.cost.Bay.Area))/(max(irrig.cost.Bay.Area)-
min(irrig.cost.Bay.Area))
```

```
irrig.scaled2 <- (irrig.cost.Bay.Area-min(irrig.cost.Bay.Area)+
.00001)/(max(irrig.cost.Bay.Area)) # this one worked, because it couldn't be
at 0

sort (irrig.scaled2)

irrig.beta.fit<- fitdistr (irrig.scaled2, "beta", start = list(shape1=2,
shape2=5), control=list(trace=1, REPORT=1))
str(irrig.beta.fit)
irrig.beta.fit$estimate [1]
irrig.beta <- rbeta(10000, irrig.beta.fit$estimate [1],
irrig.beta.fit$estimate [2])

irrig.descaled<- irrig.beta* max(irrig.cost.Bay.Area)+
min(irrig.cost.Bay.Area)

recycle.irrig.cost.par1<- irrig.beta.fit$estimate [1] # these are the beta
parameters, but scaled 0-1
recycle.irrig.cost.par2<- irrig.beta.fit$estimate [2]

irrig.quant <- qbeta(c(.1, .9), 0.31317989, 1.16505723)
irrig.quant.descaled<- irrig.quant* max(irrig.cost.Bay.Area)+
min(irrig.cost.Bay.Area)
irrig.quant.descaled[1]

recycle.irrig.cost.quant10 <- ((sum(recycle.irrig.amount)
                                * irrig.quant.descaled[1]) # $/AF/year
                               * 3.06888785 * 365 * 30) # final unit is $

recycle.irrig.cost.quant90 <- ((sum(recycle.irrig.amount)
                                * irrig.quant.descaled[2]) # $/AF/year
                               * 3.06888785 * 365 * 30) # final unit is $

# recycle.irrig.CO2
recycle.irrig.CO2.par1 <- .8 * recycle.irrig.CO2
recycle.irrig.CO2.par2 <- 1.2 * recycle.irrig.CO2

# recycle.irrig.habitat
recycle.irrig.habitat.par1<- recycle.irrig.habitat
recycle.irrig.habitat.par2<- recycle.irrig.habitat

#recycle.irrig.supply
recycle.irrig.supply.par1<- .8 * recycle.irrig.supply
recycle.irrig.supply.par2<- 1.2 * recycle.irrig.supply

#recycle.irrig.n.recovery
recycle.irrig.n.recovery.par1<- .8 * recycle.irrig.n.recovery
recycle.irrig.n.recovery.par2<- 1.2 * recycle.irrig.n.recovery

#recycle.irrig.adapt
recycle.irrig.adapt.par1<- 30
recycle.irrig.adapt.par2 <- 60

#recycle.irrig.slr
recycle.irrig.slr.par1<- recycle.irrig.slr
recycle.irrig.slr.par2<- recycle.irrig.slr
```

```
#recycle.irrig.ease.of.use
recycle.irrig.ease.of.use.par1<- recycle.irrig.ease.of.use
recycle.irrig.ease.of.use.par2<- recycle.irrig.ease.of.use

#recycle.irrig.access
recycle.irrig.access.par1<- recycle.irrig.access
recycle.irrig.access.par2<- recycle.irrig.access

#recycle.irrig.permit
recycle.irrig.permit.par1 <- 65
recycle.irrig.permit.par2 <- 95

# recycle.brineline
#################

#recycle.brineline.load.change
recycle.brineline.load.change.par1 <- .8 * recycle.brineline.load.change
recycle.brineline.load.change.par2 <- 1.2 * recycle.brineline.load.change

#recycle.brineline.wq
recycle.brineline.wq.par1<- .8 * wq.par[5]
recycle.brineline.wq.par2<- 1.2 * wq.par[5]

#recycle.brineline.CEC
recycle.brineline.CEC.par1<- .8 * recycle.brineline.CEC
recycle.brineline.CEC.par2<- 1.2 * recycle.brineline.CEC

#recycle.brineline.reliable
recycle.brineline.reliable.par1 <- recycle.brineline.reliable
recycle.brineline.reliable.par2 <- .01 * recycle.brineline.reliable

pumping.energy <- ((1/2655220) # kwh/ft-lbs of lift assume weight of brine =
weight of water
                   * recycle.brine #units is MGD
                   * 2000 # ft, height over the peninsula
                   * (8.34 * 10^6) *365 / 1000) # final unit is Mwh/year
pumping.energy.CO2 <- (pumping.energy * 320896  # unit here after
multiplication is g CO2 e/ million gallons (conversion from
west.berkeley.edu)
                       * recycle.brine / (10^6) * 365) # final unit is tonnes
CO2 e/year, g/tonne, days/year

pumping.energy.low <- ((1/2655220) # kwh/ft-lbs of lift assume weight of
brine = weight of water
                       * recycle.brine #units is MGD
                       * 1500 # ft, height over the peninsula
                       * (8.34 * 10^6) *365 / 1000) # final unit is Mwh/year
pumping.energy.CO2.low <- (pumping.energy.low * 320896  # unit here after
multiplication is g CO2 e/ million gallons (conversion from
west.berkeley.edu)
                           * recycle.brine / (10^6) * 365) # final unit is
tonnes CO2 e/year, g/tonne, days/year

pumping.energy.high <- ((1/2655220) # kwh/ft-lbs of lift assume weight of
brine = weight of water
                        * recycle.brine #units is MGD
                        * 2500 # ft, height over the peninsula
```

```
                        * (8.34 * 10^6) *365 / 1000) # final unit is Mwh/year
pumping.energy.CO2.high <- (pumping.energy.high * 320896    # unit here after
multiplication is g CO2 e/ million gallons (conversion from
west.berkeley.edu)
                            * recycle.brine / (10^6) * 365) # final unit is
tonnes CO2 e/year, g/tonne, days/year
pumping.energy.CO2.high/pumping.energy.CO2
pumping.energy.CO2.low/pumping.energy.CO2
#...so +/- 25 %

# pipe length isn't a huge CO2 contributer here...mostly pumping energy
pipe.length <- 36960 # ft across peninsula (roughly Millbrae to Pacifica)
pipe.CO2 <- (pipe.length * 358  # unit here after multiplication is g CO2 e/
million gallons (conversion from west.berkeley.edu)
             * recycle.brine / (10^6) * 365) #final unit is tonnes CO2
e/year, g/tonne, days/year
pumping.CO2 <- pumping.energy.CO2 + pipe.CO2 # unit is tonnes CO2e/year

pipe.length.low <- 5* 5280 # ft across peninsula (roughly Millbrae to
Pacifica)
pipe.CO2.low <- (pipe.length.low * 358  # unit here after multiplication is g
CO2 e/ million gallons (conversion from west.berkeley.edu)
                 * recycle.brine / (10^6) * 365) #final unit is tonnes CO2
e/year, g/tonne, days/year
pumping.CO2.low <- pumping.energy.CO2 + pipe.CO2.low # unit is tonnes
CO2e/year

pipe.length.high <- 18* 5280 # ft across peninsula (roughly Millbrae to
Pacifica)
pipe.CO2.high <- (pipe.length.high * 358    # unit here after multiplication
is g CO2 e/ million gallons (conversion from west.berkeley.edu)
                  * recycle.brine / (10^6) * 365) #final unit is tonnes CO2
e/year, g/tonne, days/year
pumping.CO2.high <- pumping.energy.CO2 + pipe.CO2.high # unit is tonnes
CO2e/year

#recycle.brineline.CO2: model with +/- 25%
recycle.brineline.CO2.par1 <- .75 * recycle.brineline.CO2
recycle.brineline.CO2.par2 <- 1.25 * recycle.brineline.CO2

pumping.cost <- ((1/2655220) # kwh/ft-lbs of lift assume weight of brine =
weight of water
                 * 0.2 # $/kwh for energy
                 * recycle.brine #units is MGD
                 * 2000 # ft, height over the peninsula
                 * (8.34 * 10^6) *365*30) # final unit is $ for 30 years,
pounds/million gallons, days/year, years
pumping.cost.low <- ((1/2655220) # kwh/ft-lbs of lift assume weight of brine
= weight of water
                     * 0.05 # $/kwh for energy
                     * recycle.brine #units is MGD
                     * 1500 # ft, height over the peninsula
                     * (8.34 * 10^6) *365*30) # final unit is $ for 30 years,
pounds/million gallons, days/year, years
pumping.cost.high <- ((1/2655220) # kwh/ft-lbs of lift assume weight of brine
= weight of water
                      * 0.3 # $/kwh for energy
```

```
                      * recycle.brine #units is MGD
                      * 2500 # ft, height over the peninsula
                      * (8.34 * 10^6) *365*30) # final unit is $ for 30
years, pounds/million gallons, days/year, years
pumping.cost.low/pumping.cost
pumping.cost.high/pumping.cost
#...more than 85% differences based on height and price of energy
recycle.brineline.cost <- ((sum(recycle.ipr) * 3.06888785 * 365 # AF/million
gallons, days/year
                            * (700 + mean(c( 120, 750, 1250)))) # $/AF,
advanced treatment plus conveyance to drinking water
                           + (sum(recycle.dpr) * 3.06888785 * 365 #
AF/million gallons, days/year
                              * (700 + 120)) # $/AF, advanced treatment plus
conveyance to drinking water
                           + (recycle.brine * 115 * 3.06888785 * 365 ) #
final unit is $/AF, cost of pipe for amount of water, AF/million gallons,
days/year
                           + 2000000 * (pipe.length/5280) # cost of brineline
pipe per distance, ft/mile
                           +  pumping.cost) # final unit is $
recycle.brineline.cost.low <- ((sum(recycle.ipr) * 3.06888785 * 365 #
AF/million gallons, days/year
                                * (700 + 120)) # $/AF, advanced treatment
plus conveyance to drinking water
                               + (sum(recycle.dpr) * 3.06888785 * 365 #
AF/million gallons, days/year
                                  * (700 + 120)) # $/AF, advanced treatment
plus conveyance to drinking water
                               + (recycle.brine * 115 * 3.06888785 * 365 ) #
final unit is $/AF, cost of pipe for amount of water, AF/million gallons,
days/year
                               + 2000000 * (pipe.length.low/5280) # cost of
brineline pipe per distance, ft/mile
                               +  pumping.cost.low) # final unit is $
recycle.brineline.cost.high <- ((sum(recycle.ipr) * 3.06888785 * 365 #
AF/million gallons, days/year
                                 * (700 + 1250)) # $/AF, advanced treatment
plus conveyance to drinking water
                                + (sum(recycle.dpr) * 3.06888785 * 365 #
AF/million gallons, days/year
                                   * (700 + 120)) # $/AF, advanced treatment
plus conveyance to drinking water
                                + (recycle.brine * 115 * 3.06888785 * 365 ) #
final unit is $/AF, cost of pipe for amount of water, AF/million gallons,
days/year
                                + 2000000 * (pipe.length.high/5280) # cost of
brineline pipe per distance, ft/mile
                                +  pumping.cost.high) # final unit is $
recycle.brineline.cost.low/recycle.brineline.cost
recycle.brineline.cost.high/recycle.brineline.cost

#recycle.brineline.cost
recycle.brineline.cost.par1 <- .35 * recycle.brineline.cost
recycle.brineline.cost.par2 <- 1.7 * recycle.brineline.cost

#recycle.brineline.habitat
```

```
recycle.brineline.habitat.par1<- recycle.brineline.habitat
recycle.brineline.habitat.par2<- recycle.brineline.habitat

#recycle.brineline.supply
recycle.brineline.supply.par1 <- .8 * recycle.brineline.supply
recycle.brineline.supply.par2 <- 1.2 * recycle.brineline.supply

#recycle.brineline.n.recovery
recycle.brineline.n.recovery.par1<- recycle.brineline.n.recovery
recycle.brineline.n.recovery.par2<- recycle.brineline.n.recovery

#recycle.brineline.adapt
recycle.brineline.adapt.par1 <- 30
recycle.brineline.adapt.par2 <- 60

#recycle.brineline.slr
recycle.brineline.slr.par1<- recycle.brineline.slr
recycle.brineline.slr.par2<- recycle.brineline.slr

#recycle.brineline.ease.of.use
recycle.brineline.ease.of.use.par1 <- 30
recycle.brineline.ease.of.use.par2 <- 80

#recycle.brineline.access
recycle.brineline.access.par1<- recycle.brineline.access
recycle.brineline.access.par2<- recycle.brineline.access

#recycle.brineline.permit
recycle.brineline.permit.par1 <- 20
recycle.brineline.permit.par2 <- 70

#urine.early
##########

# proportion of population that is early adopters (tech and ed industries)
Early.adopt <- .039
# grams of N excreted in urine
N.excreted <- 10.5 # g N/person/day
# Proportion of a person's urine recovered by urine source separation
U.recovery.toilet <- .5
# Proportion of N recovered by ion exchange resin
N.recovery.resin <- .9
toilet.use <- .5 # percent of the time that people with urine-separating
toilets use them
population <-  scenario [1] * c(30000, 957000, 22850, 217330, 1498700, 0.5 *
(143100 + 156500), 10000, 670400, 110000, 148000, 200600) # shown in order of
WWTPs in Discharge.scenario.
names(population)<- colnames(Discharge.scenario[3:13])
urine.early <- Early.adopt * sum(population) + ifelse((scenario
[1])>1,(scenario [1]-1)*(sum(population)/scenario[1]),0) # if population
grows, it's early adopters + whatever population growth. If population
declines, it's just early adopters
urine.early.load <- (urine.early *  N.excreted *  N.recovery.resin *
U.recovery.toilet * toilet.use
                     / 1000 ) # final unit is kg/day, g/kg

# proportion of population that is early adopters (tech and ed industries)
```

```
Early.adopt.low <- .01
# grams of N excreted in urine
N.excreted <- 10.5 # g N/person/day
# Proportion of a person's urine recovered by urine source separation
U.recovery.toilet <- .5
# Proportion of N recovered by ion exchange resin
N.recovery.resin.low <- .8
toilet.use.low <- .4 # percent of the time that people with urine-separating
toilets use them
population <-  scenario [1] * c(30000, 957000, 22850, 217330, 1498700, 0.5 *
(143100 + 156500), 10000, 670400, 110000, 148000, 200600) # shown in order of
WWTPs in Discharge.scenario.
names(population)<- colnames(Discharge.scenario[3:13])
urine.early.low <- Early.adopt.low * sum(population) + ifelse((scenario
[1])>1,(scenario [1]-1)*(sum(population)/scenario[1]),0) # if population
grows, it's early adopters + whatever population growth. If population
declines, it's just early adopters
urine.early.load.low <- (urine.early.low *  N.excreted *
N.recovery.resin.low * U.recovery.toilet * toilet.use.low
                         / 1000 ) # final unit is kg/day, g/kg

# proportion of population that is early adopters (tech and ed industries)
Early.adopt.high <- .10
# grams of N excreted in urine
N.excreted <- 10.5 # g N/person/day
# Proportion of a person's urine recovered by urine source separation
U.recovery.toilet <- .5
# Proportion of N recovered by ion exchange resin
N.recovery.resin.high <- .99
toilet.use.high <- .8 # percent of the time that people with urine-separating
toilets use them
population <-  scenario [1] * c(30000, 957000, 22850, 217330, 1498700, 0.5 *
(143100 + 156500), 10000, 670400, 110000, 148000, 200600) # shown in order of
WWTPs in Discharge.scenario.
names(population)<- colnames(Discharge.scenario[3:13])
urine.early.high <- Early.adopt.high * sum(population) + ifelse((scenario
[1])>1,(scenario [1]-1)*(sum(population)/scenario[1]),0) # if population
grows, it's early adopters + whatever population growth. If population
declines, it's just early adopters
urine.early.load.high <- (urine.early.high *  N.excreted *
N.recovery.resin.high * U.recovery.toilet * toilet.use.high
                          / 1000 ) # final unit is kg/day, g/kg
urine.early.load.low/urine.early.load
urine.early.load.high/urine.early.load
# vary by - 35% to + 200%, depending on number of early adopters

#urine.early.load.change
urine.early.load.change.par1 <- .65 * urine.early.load.change
urine.early.load.change.par2 <- 2.1 * urine.early.load.change

#urine.early.wq
urine.early.wq.par1<- .8 * wq.par [6]
urine.early.wq.par2<- 1.2 * wq.par [6]

#urine.early.CEC
urine.early.CEC.par1<- do.nothing.CEC.par1 - (2.1 * urine.early.CEC.removal)
urine.early.CEC.par2<- do.nothing.CEC.par2 - (.65 * urine.early.CEC.removal)
```

```
urine.early.cost <- (26 # $/m3 of urine
                     * urine.early * 1.4 # L urine excreted/person/day
                     * toilet.use
                     * 365 * 30 / 1000) # final units is $ over 30 years,
days/year, yrs, L/m3
urine.early.cost.low <- (16 # $/m3 of urine
                         * urine.early.low * 1.4 # L urine
excreted/person/day
                         * toilet.use.low
                         * 365 * 30 / 1000) # final units is $ over 30 years,
days/year, yrs, L/m3
urine.early.cost.high <- (30 # $/m3 of urine
                          * urine.early.high * 1.4 # L urine
excreted/person/day
                          * toilet.use.high
                          * 365 * 30 / 1000) # final units is $ over 30
years, days/year, yrs, L/m3
urine.early.cost.low/urine.early.cost
urine.early.cost.high/urine.early.cost
#big variation (-55% - + %220) depending on adoption rates and cost of
treatment (depends most on number of facilities built)
#urine.early.cost
urine.early.cost.par1 <- .45 * urine.early.cost
urine.early.cost.par2 <- 2.2 * urine.early.cost

#urine.early.CO2
# note: the CO2 number is mostly influenced by do.nothing.CO2, only about 1%
by urine
urine.early.CO2.par1 <- .8 * urine.early.CO2
urine.early.CO2.par2 <- 1.2 * urine.early.CO2

#urine.early.habitat
urine.early.habitat.par1<- urine.early.habitat
urine.early.habitat.par2<- urine.early.habitat

#urine.early.supply
urine.early.supply.par1<- urine.early.supply
urine.early.supply.par2<- urine.early.supply

#urine.early.n.recovery
urine.early.n.recovery.par1<- .65* urine.early.n.recovery
urine.early.n.recovery.par2<- 2.1 *urine.early.n.recovery

#urine.early.adapt
urine.early.adapt.par1 <- 75
urine.early.adapt.par2 <- 95

#urine.early.slr
urine.early.slr.par1<- urine.early.slr
urine.early.slr.par2<- urine.early.slr

#urine.early.ease.of.use
urine.early.ease.of.use.par1 <- 15
urine.early.ease.of.use.par2 <- 50

#urine.early.reliable
```

```
urine.early.reliable.par1<- .8 * urine.early.reliable
urine.early.reliable.par2<- 1.2 * urine.early.reliable

#urine.early.access
urine.early.access.par1<- urine.early.access
urine.early.access.par2<- urine.early.access

#urine.early.permit
urine.early.permit.par1 <- 20
urine.early.permit.par2 <- 70

#urine.incentives
###############

urine.incentives <- (0.3 * sum(population)) + ifelse((scenario
[1])>1,(scenario [1]-1)*(sum(population)/scenario[1]),0) # 30% of current
population + all population growth
urine.incentives.low <- .8 * urine.incentives
urine.incentives.high <- 1.4 * urine.incentives
#urine.incentives.load = kg TN/day removed by urine source separation
toilet.use.incentives <- .75 # percent of the time that people with urine-
separating toilets use them
urine.incentives.load <- (urine.incentives *  N.excreted *  N.recovery.resin
* U.recovery.toilet * toilet.use.incentives
                          / 1000 ) # final unit is kg/day, g/kg

#urine.incentives.load.change
urine.incentives.load.change.par1 <- .8 * urine.incentives.load.change
urine.incentives.load.change.par2 <- 1.4 * urine.incentives.load.change

#urine.incentives.wq
urine.incentives.wq.par1 <- .8 * wq.par [7]
urine.incentives.wq.par2 <- 1.2 * wq.par [7]

#urine.incentives.CEC
urine.incentives.CEC.par1<- do.nothing.CEC.par1 - (1.4 *
urine.incentives.CEC.removal)
urine.incentives.CEC.par2<- do.nothing.CEC.par2 - (.8 *
urine.incentives.CEC.removal)

#urine.incentives.reliable
urine.incentives.reliable.par1<- .8 * urine.incentives.reliable
urine.incentives.reliable.par2<- 1.2 * urine.incentives.reliable

incentive1.low <- 500 # initial retrofit incentive
incentive2.low <- (1 * 52 * 30) #  weekly reward for turning in cartridge for
30 year lifecycle, $ reward, weeks/year, years
#household <- 2.7
outreach.low <- 100000 * 11 # 11 is number of participating utilities
urine.incentives.cost.low <- (((16 # $/m3 of urine
                                * urine.incentives.low * 1.4 # L urine
excreted/person/day
                                * toilet.use.incentives
                                * 365 * 30 / 1000) # final units is $ over 30
years, days/year, yrs, L/m3
```

```
                              + (incentive1.low * (0.3 *
(sum(population)/scenario[1]) / household )) # $/household to install toilet
for all existing population
                              + (incentive2.low *
urine.incentives.low/household) # people per household, initial incentive
only for retrofits, not new housing stock
                              + outreach.low)) # final unit is $

incentive1.high <- 4000 # initial retrofit incentive
incentive2.high <- (4 * 52 * 30) #  weekly reward for turning in cartridge
for 30 year lifecycle, $ reward, weeks/year, years
#household <- 2.7
outreach.high <- 400000 * 11 # 11 is number of participating utilities
urine.incentives.cost.high <- (((30 # $/m3 of urine
                                * urine.incentives.high * 1.4 # L urine
excreted/person/day
                                * toilet.use.incentives
                                * 365 * 30 / 1000) # final units is $ over
30 years, days/year, yrs, L/m3
                               + (incentive1.high * (0.3 *
(sum(population)/scenario[1]) / household )) # $/household to install toilet
for all existing population
                               + (incentive2.high *
urine.incentives.high/household) # people per household, initial incentive
only for retrofits, not new housing stock
                               + outreach.high)) # final unit is $
urine.incentives.cost.low/urine.incentives.cost
urine.incentives.cost.high/urine.incentives.cost

#cost
urine.incentives.cost.par1<- .35 * urine.incentives.cost
urine.incentives.cost.par2<-2.4 * urine.incentives.cost

#urine.incentives.CO2
#again, these variables have very small affect on the outcomes
urine.incentives.CO2.par1<- .8 * urine.incentives.CO2
urine.incentives.CO2.par2<- 1.2 * urine.incentives.CO2

#urine.incentives.habitat
urine.incentives.habitat.par1<- urine.incentives.habitat
urine.incentives.habitat.par2<- urine.incentives.habitat

#urine.incentives.supply
urine.incentives.supply.par1<- urine.incentives.supply
urine.incentives.supply.par2<-urine.incentives.supply

#urine.incentives.n.recovery
urine.incentives.n.recovery.par1<- .8 * urine.incentives.n.recovery
urine.incentives.n.recovery.par2<- 1.5 * urine.incentives.n.recovery

#urine.incentives.adapt
urine.incentives.adapt.par1<- 45
urine.incentives.adapt.par2<-65

#urine.incentives.slr
urine.incentives.slr.par1<- urine.incentives.slr
urine.incentives.slr.par2 <- urine.incentives.slr
```

```
#urine.incentives.ease.of.use
urine.incentives.ease.of.use.par1<- 25
urine.incentives.ease.of.use.par2<- 45

#urine.incentives.access
urine.incentives.access.par1<- urine.incentives.access
urine.incentives.access.par2<- urine.incentives.access

#urine.incentives.permit
urine.incentives.permit.par1<- 30
urine.incentives.permit.par2<- 50

# opt
####

#opt.load
# assuming optimization removes 10%, final unit is kg/day
opt.load.change.par1<- opt.load.change
opt.load.change.par2<- abs(.1 * opt.load.change)

#opt.wq
opt.wq.par1<- .8 * wq.par[8]
opt.wq.par2<- 1.2 * wq.par[8]

#opt.CEC
opt.CEC.par1<- .8 * opt.CEC
opt.CEC.par2<- 1.2 * opt.CEC

#opt.cost
opt.cost.par1<- opt.cost
opt.cost.par2<- .1 * opt.cost

#opt.CO2
opt.CO2.par1<- opt.CO2
opt.CO2.par2<- opt.CO2

#opt.habitat
opt.habitat.par1<- opt.habitat
opt.habitat.par2<- opt.habitat

#opt.supply
opt.supply.par1<- opt.supply
opt.supply.par2<- opt.supply

#opt.n.recovery
opt.n.recovery.par1<- opt.n.recovery
opt.n.recovery.par2<- opt.n.recovery

#opt.adapt
opt.adapt.par1<- 60
opt.adapt.par2<- 90

#opt.slr
opt.slr.par1<- opt.slr
opt.slr.par2<- opt.slr
```

```
#opt.ease.of.use
opt.ease.of.use.par1<- opt.ease.of.use
opt.ease.of.use.par2<- opt.ease.of.use

#opt.access
opt.access.par1<- opt.access
opt.access.par2<- opt.access

#opt.permit
opt.permit.par1<- 80
opt.permit.par2<- 100

#opt.reliable
opt.reliable.par1<- opt.reliable
opt.reliable.par2<- .01 * opt.reliable

# Level2
########

#Level2.load
Level2.load.change.par1<- Level2.load.change
Level2.load.change.par2<- abs(.1 * Level2.load.change)

#Level2.wq
Level2.wq.par1<- .8 * wq.par[9]
Level2.wq.par2<- 1.2 * wq.par[9]

#Level2.CEC
# assumes no additional removal from plants that already have tertiary
treatment
Level2.CEC.par1<- .8 * Level2.CEC
Level2.CEC.par2<- 1.2 * Level2.CEC

#Level2.cost
Level2.cost.par1<- Level2.cost
Level2.cost.par2<- .1 * Level2.cost

#Level2.reliable
Level2.reliable.par1<- Level2.reliable
Level2.reliable.par2<- .01 * Level2.reliable

#Level2.CO2
Level2.CO2.par1<- .8 * Level2.CO2
Level2.CO2.par2<- 1.2 * Level2.CO2

#Level2.habitat
Level2.habitat.par1<- Level2.habitat
Level2.habitat.par2<- Level2.habitat

#Level2.supply
Level2.supply.par1<- Level2.supply
Level2.supply.par2<- Level2.supply

#Level2.n.recovery
Level2.n.recovery.par1<- Level2.n.recovery
Level2.n.recovery.par2<- Level2.n.recovery
```

```
#Level2.adapt
Level2.adapt.par1<- 30
Level2.adapt.par2<- 75

#Level2.slr
Level2.slr.par1<- Level2.slr
Level2.slr.par2<- Level2.slr

#Level2.ease.of.use
Level2.ease.of.use.par1<- Level2.ease.of.use
Level2.ease.of.use.par2<- Level2.ease.of.use

#Level2.access
Level2.access.par1<- Level2.access
Level2.access.par2<- Level2.access

#Level2.permit
Level2.permit.par1<- 80
Level2.permit.par2<- 100

# Level3
########


#Level3.load
Level3.load.change.par1<- Level3.load.change
Level3.load.change.par2<- abs(.1 * Level3.load.change)

#Level3.wq
Level3.wq.par1<- .8 * wq.par[10]
Level3.wq.par2<- 1.2 * wq.par[10]

#Level3.CEC
Level3.CEC.par1<- .8 * Level3.CEC
Level3.CEC.par2<- 1.2 * Level3.CEC

#Level3.cost
Level3.cost.par1<- Level3.cost
Level3.cost.par2<- .1 * Level3.cost

#Level3.reliable
Level3.reliable.par1<- Level3.reliable
Level3.reliable.par2<- .01 * Level3.reliable

#Level3.CO2
Level3.CO2.par1<- .8 * Level3.CO2
Level3.CO2.par2<- 1.2 * Level3.CO2

#Level3.habitat
Level3.habitat.par1<- Level3.habitat
Level3.habitat.par2<- Level3.habitat

#Level3.supply
Level3.supply.par1<- Level3.supply
Level3.supply.par2<- Level3.supply

#Level3.n.recovery
```

```
Level3.n.recovery.par1<- Level3.n.recovery
Level3.n.recovery.par2<- Level3.n.recovery

#Level3.adapt
Level3.adapt.par1<- 5
Level3.adapt.par2<- 15

#Level3.slr
Level3.slr.par1<- Level3.slr
Level3.slr.par2<- Level3.slr

#Level3.ease.of.use
Level3.ease.of.use.par1<- Level3.ease.of.use
Level3.ease.of.use.par2<- Level3.ease.of.use

#Level3.access
Level3.access.par1<- Level3.access
Level3.access.par2<- Level3.access

#Level3.permit
Level3.permit.par1<- 80
Level3.permit.par2<- 100

######Parameter 1--create master dataframe with all par.1 values
par.1.test<-ls(globalenv())[grepl(".par1",ls(globalenv()))]

par.1.master<-cbind(
  mget(par.1.test[grepl("wq", par.1.test)]),
  mget(par.1.test[grepl("habitat", par.1.test)]),
  mget(par.1.test[grepl("supply", par.1.test)]),
  mget(par.1.test[grepl("recovery", par.1.test)]),
  mget(par.1.test[grepl("CEC.par", par.1.test)]),
  mget(par.1.test[grepl("adapt", par.1.test)]),
  mget(par.1.test[grepl("slr", par.1.test)]),
  mget(par.1.test[grepl("CO2", par.1.test)]),
  mget(par.1.test[grepl("ease.of", par.1.test)]),
  mget(par.1.test[grepl("access", par.1.test)]),
  mget(par.1.test[grepl("permit", par.1.test)]),
  mget(par.1.test[grepl("reliable", par.1.test)]),
  mget(par.1.test[grepl("cost.par", par.1.test)]))
par.1.master<-par.1.master[c(1,9,10,6,5,7,8,4,2,3),]
colnames(par.1.master)<-criteria
rownames(par.1.master)<-options
par.1.master[1,1]

######Parameter 2--create master dataframe with all par.2 values
par.2.test<-ls(globalenv())[grepl(".par2",ls(globalenv()))]
par.2.master<-cbind(
  mget(par.2.test[grepl("wq", par.2.test)]),
  mget(par.2.test[grepl("habitat", par.2.test)]),
  mget(par.2.test[grepl("supply", par.2.test)]),
  mget(par.2.test[grepl("recovery", par.2.test)]),
  mget(par.2.test[grepl("CEC.par", par.2.test)]),
  mget(par.2.test[grepl("adapt", par.2.test)]),
  mget(par.2.test[grepl("slr", par.2.test)]),
  mget(par.2.test[grepl("CO2", par.2.test)]),
  mget(par.2.test[grepl("ease.of", par.2.test)]),
```

```
  mget(par.2.test[grepl("access", par.2.test)]),
  mget(par.2.test[grepl("permit", par.2.test)]),
  mget(par.2.test[grepl("reliable", par.2.test)]),
  mget(par.2.test[grepl("cost.par", par.2.test)]))
par.2.master<-par.2.master[c(1,9,10,6,5,7,8,4,2,3),]
colnames(par.2.master)<-criteria
rownames(par.2.master)<-options
par.2.master

# for beta distribution, scalars
################################

Max_scalar_beta<- data.frame(matrix(nrow = 10, ncol = 13))
rownames(Max_scalar_beta)<- options
colnames(Max_scalar_beta)<- criteria

Min_scalar_beta<- data.frame(matrix(nrow = 10, ncol = 13))
rownames(Min_scalar_beta)<- options
colnames(Min_scalar_beta)<- criteria

Max_scalar_beta[4,13]<- ((sum(recycle.irrig.amount)
                          * max(irrig.cost.Bay.Area)) # $/AF/year
                         * 3.06888785 * 365 * 30)

Min_scalar_beta[4,13]<- ((sum(recycle.irrig.amount)
                          * min(irrig.cost.Bay.Area)) # $/AF/year
                         * 3.06888785 * 365 * 30)

#making data sheet from parameters
csv.columns <-c("Scenario", "Criteria", "Code_name",    "Unit",
"Distribution", "Parameter1",   "Parameter2",   "Max_scalar_beta",
"Min_scalar_beta",  "Alt_sample",   "Best", "Worst",    "Description")
master.csv <- data.frame (matrix(ncol=13, nrow=13))
colnames (master.csv) <- csv.columns

best <- c(5,    5200,   190.1,  8500000,    22, 100,    10, 180000, 100,
17, 100,    99, 0)
worst <- c(95,  0,  22, 0,  137,    0,  -10,    900000, 0,  0,  0,  50,
8000000000)

master.csv[,2]<- criteria
master.csv[,3]<- criteria
master.csv[,4]<- c( "%",    "hectares", "MGD",  "kg/year",  "kg/year",  "%",
"scale_constr", "tonnes CO2 eq/year",   "%",    "Integer",  "%",    "%",
"$")
master.csv [, 11] <- best
master.csv[,12]<- worst

master.csv

csv.list<-list(NULL)
for(i in 1:10){
  csv.list[[i]]<-master.csv
}
names(csv.list)<-options
csv.list
```

```
for (j in 1:10){

  for(i in 1:13){
    csv.list[[j]][i,1] <- paste("pop=", scenario[1], ",", "SeaLev=",
scenario[2], "Threshold=", scenario[3])
    csv.list[[j]][i,10]<- criteria.calcs[j,i] # this fills in Alt_sample
    csv.list[[j]][i,6]<- par.1.master[j,i]
    csv.list[[j]][i,7]<- par.2.master[j,i]
    csv.list[[j]][i,5]<- as.character(uncertainty.distributions[i,j])
    csv.list[[j]][i,8]<- Max_scalar_beta[j,i]
    csv.list[[j]][i,9]<- Min_scalar_beta[j,i]
  }
}
csv.list
```

Do a Monte-Carlo style simulation of uncertainty of all the parameters.

```
n <- 1000 #number of runs
column.x<-NULL
option.sample <- data.frame(matrix(NA, nrow = n, ncol = 14))
option.list <-  NULL
boundary.min <- 0
boundary.max <- 0
out.of.bounds.min<-data.frame(matrix(ncol=3))
out.of.bounds.max<-data.frame(matrix(ncol=3))

a.matrix.names <- c("a", criteria)
colnames(option.sample) <- a.matrix.names

for(j in 1:10){# 10 is number of options in the list

  for (i in 1:13){# 13 is number of criteria
    # column.x is to keep everything within the range of best and worst
    # This loop runs an 'n' numbered sample from truncated normal
distribution if "Normal", otherwise a sample from a uniform distribution
    #Parameters are mean and SD for normal distribution, min and max for
uniform distribution. If 'Determined', min = max in uniform

    column.x<-{
      if(csv.list [[j]]$Distribution[i] == "Normal") rtruncnorm(n, a =
min(c(csv.list[[j]]$Best[i],csv.list[[j]]$Worst[i])), b =
max(c(csv.list[[j]]$Best[i],csv.list[[j]]$Worst[i])), mean =
as.numeric(csv.list[[j]]$Parameter1[i]), sd = as.numeric(csv.list
[[j]]$Parameter2[i]))
    else if (csv.list [[j]]$Distribution[i] == "Minmax") runif(n, min =
min(c(as.numeric(csv.list[[j]]$Parameter1[i]),
as.numeric(csv.list[[j]]$Parameter2[i]))), max =
max(c(as.numeric(csv.list[[j]]$Parameter1[i]),
as.numeric(csv.list[[j]]$Parameter2[i]))))
    else if (csv.list [[j]]$Distribution[i] == "Determined") csv.list
[[j]]$Alt_sample [i]
    else if (csv.list [[j]]$Distribution[i] == "Beta") rbeta(n,csv.list
[[j]]$Parameter1[i], csv.list [[j]]$Parameter2[i])*csv.list
[[j]]$Max_scalar_beta[i] + csv.list [[j]]$Min_scalar_beta[i]
    }
```

```
      min.x<-min(c(csv.list [[j]]$Best[i],csv.list [[j]]$Worst[i]))
    max.x<-max(c(csv.list [[j]]$Best[i],csv.list [[j]]$Worst[i]))

    column.x[column.x<min.x]<-csv.list[[j]]$Alt_sample[i]
    column.x[column.x>max.x]<-csv.list[[j]]$Alt_sample[i]

   boundary.min <- boundary.min + sum(column.x<min.x)
   boundary.max <- boundary.max + sum(column.x > max.x)
   out.of.bounds.min[(j-1)*13+i,] <- c(sum(column.x<min.x), options[j], 
criteria[i])
   out.of.bounds.max[(j-1)*13+i,] <- c(sum(column.x>max.x), options[j], 
criteria[i])


    option.sample[,i+1]<-column.x
    option.sample[,1]<-rep(options[j],n)

  }

  option.list[[j]] <- option.sample
}

table(round(option.list[[2]]$access))
names(option.list) <- options

str(option.list)

out.of.bounds.max[out.of.bounds.max[,1]>0,]
out.of.bounds.min[out.of.bounds.min[,1]>0,]

boundary.max
boundary.min

#option list without recycle.brineline
option.list.no.brine<- option.list[c(1:4,6:10)]
str(option.list.no.brine)
```

Make tables of all the attribute values.

```
attr.table<- rbind(apply(data.frame(lapply(option.list.no.brine, `[`, 2)), 2, 
mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 3)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 4)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 5)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 6)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 7)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 8)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 9)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 10)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 11)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 12)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 13)), 2, mean),
apply(data.frame(lapply(option.list.no.brine, `[`, 14)), 2, mean))
rownames(attr.table)<-criteria
colnames(attr.table) <- options.no.brine
```

```
head(attr.table)

#table of mean attribute values with units and brine option
############################################################
attr.table.complete<- rbind(apply(data.frame(lapply(option.list, `[`, 2)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 3)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 4)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 5)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 6)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 7)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 8)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 9)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 10)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 11)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 12)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 13)), 2, mean),
                   apply(data.frame(lapply(option.list, `[`, 14)), 2, mean))
rownames(attr.table.complete)<-criteria
colnames(attr.table.complete) <- options
units.df <- c("% likelihood deviation", "square hectares", "MGD", "Kg N/year", "Kg SMX/year", "% ease of adaptation", "scale -10 to 10", "CO2 eq/year", "% ease of use", "Number of access points", "% ease of permitting", "% of time reliable", "Total present value (30 year life span)")
attr.table.complete.w.units<-data.frame(cbind(units = units.df, attr.table.complete))
rownames(attr.table.complete.w.units)<-NULL
colnames(attr.table.complete.w.units)<-NULL


attr.table.final<-data.frame(matrix(ncol = 11))
for(i in 2:11){
  attr.table.final[1:13,i]<-
signif(as.numeric(as.character(attr.table.complete.w.units[,i])), digits = 3)

}

attr.table.final[,1]<-units.df
rownames(attr.table.final)  <- criteria
colnames(attr.table.final) <- c("units", options)


head(attr.table.final)
```

Plot the uncertainty distributions of attribute values for each of the objectives.

```
#plot attribute values for each criteria

par(mfrow=c(1,1),
    mar = c(2,2,2,2),
    oma = c(4,2,2,2),
    cex.lab = .8)

#wq
boxplot(data.frame(lapply(option.list.no.brine, `[`, 2)), names = options.no.brine,
```

```
        main = "% likelihood of deviating from good water quality",
        ylim = c(100,0))
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#hab
boxplot(data.frame(lapply(option.list.no.brine, `[`, 3)), names =
options.no.brine,
        main = "Hectares of wetland area created")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#supply
boxplot(data.frame(lapply(option.list.no.brine, `[`, 4)), names =
options.no.brine,
        main = "MGD of useable water supplied")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#recovery
boxplot(data.frame(lapply(option.list.no.brine, `[`, 5)), names =
options.no.brine,
        main = "Kg/year of nitrogen recovered for fertilizer")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#CEC
boxplot(data.frame(lapply(option.list.no.brine, `[`, 6)), names =
options.no.brine,
        main = "Kg/year total sulfamethoxazole loading to southern reach of
Bay")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#ADAPT
boxplot(data.frame(lapply(option.list.no.brine, `[`, 7)), names =
options.no.brine,
```

```
        main = "% Ease of adaptation")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#SLR
boxplot(data.frame(lapply(option.list.no.brine, `[`, 8)), names =
options.no.brine,
        main = "Resilience to Sea Level Rise (-10 = highly vulnerable, 0  =
unaffected, 10 = protects other infrastructure)", ylim = c(-10, 10))
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#CO2
boxplot(data.frame(lapply(option.list.no.brine, `[`, 9)), names =
options.no.brine,
        main = "Tonnes CO2 e/year from wastewater treatment")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#EASE of USE
boxplot(data.frame(lapply(option.list.no.brine, `[`, 10)), names =
options.no.brine,
        main = "% Ease of use")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#ACCESS
boxplot(data.frame(lapply(option.list.no.brine, `[`, 11)), names =
options.no.brine,
        main = "Additional shoreline access points")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#PERMIT
boxplot(data.frame(lapply(option.list.no.brine, `[`, 12)), names =
options.no.brine,
        main = "% Ease of permitting")
```

```
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#RELIABLE
boxplot(data.frame(lapply(option.list.no.brine, `[`, 13)), names =
options.no.brine,
        main = "% of time nutrient control operates as expected")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)

#COSTS
boxplot(data.frame(lapply(option.list.no.brine, `[`, 14)), names =
options.no.brine,
        main = "Costs (30 year net present value)")
mtext(paste("Population change =", "", scenario [1]), side = 1, line = 2, cex
= 0.6)
mtext(paste("Climate effect =", "", scenario [2]), side = 1, line = 3, cex =
0.6)
mtext(paste("Ecological threshold =", "", scenario [3]), side = 1, line = 4,
cex = 0.6)
```

Read stakeholder preferences into R.

```
pref.nutrients <- pref.nutrients.sensitivity
head(pref.nutrients)
class(pref.nutrients$val.SH7)
#SH7
pars.nutrients.SH7 <- pref.nutrients$val.SH7
names(pars.nutrients.SH7) <- pref.nutrients$name
pars.nutrients.SH7
sum(pref.nutrients[1:18,3], na.rm = TRUE)
#SH3
pars.nutrients.SH3 <- pref.nutrients$val.SH3
names(pars.nutrients.SH3) <- pref.nutrients$name
pars.nutrients.SH3
sum(pref.nutrients[1:18,4], na.rm = TRUE)
#SH2
pars.nutrients.SH2 <- pref.nutrients$val.SH2
names(pars.nutrients.SH2) <- pref.nutrients$name
pars.nutrients.SH2
sum(pref.nutrients[1:18,5])
#SH8
pars.nutrients.SH8 <- pref.nutrients$val.SH8
names(pars.nutrients.SH8) <- pref.nutrients$name
pars.nutrients.SH8
sum(pref.nutrients[1:18,6])
#SH5
pars.nutrients.SH5 <- pref.nutrients$val.SH5
```

```
names(pars.nutrients.SH5) <- pref.nutrients$name
pars.nutrients.SH5
sum(pref.nutrients[1:18,7])
#SH1
pars.nutrients.SH1 <- pref.nutrients$val.SH1
names(pars.nutrients.SH1) <- pref.nutrients$name
pars.nutrients.SH1
sum(pref.nutrients[1:18,8])
#SH4
pars.nutrients.SH4 <- pref.nutrients$val.SH4
names(pars.nutrients.SH4) <- pref.nutrients$name
pars.nutrients.SH4
sum(pref.nutrients[1:18,9])
#SH6
pars.nutrients.SH6 <- pref.nutrients$val.SH6
names(pars.nutrients.SH6) <- pref.nutrients$name
pars.nutrients.SH6
sum(pref.nutrients[1:18,10])
#SH9
pars.nutrients.SH9 <- pref.nutrients$val.SH9
names(pars.nutrients.SH9) <- pref.nutrients$name
pars.nutrients.SH9
sum(pref.nutrients[1:18,11])

#vector of stakeholder preferences
SH.pars.nutrients.all<- data.frame(cbind(pars.nutrients.SH1,
pars.nutrients.SH2, pars.nutrients.SH3, pars.nutrients.SH4,
pars.nutrients.SH5, pars.nutrients.SH6, pars.nutrients.SH7,
pars.nutrients.SH8, pars.nutrients.SH9))
SH.pars.nutrients.all[2]

#vector of stakeholder preferences with regular objectives hierarchy
(excluding special cases)
SH.pars.nutrients.normal <- SH.pars.nutrients.all[c(1,2,4:6,8,9)]
```

Plot stakeholder points for each criteria.

```
SH.points <- read.csv("Inputs/Stakeholder points.csv")
SH.points
Criteria.names<- c("Water Quality", "Wetland Habitat", "Water Supply",
"Nutrient Recovery", "CECs",
         "Ease of Adaptation", "Sea Level Rise", "CO2 Emissions", "Ease of
Use",
         "Shoreline Access", "Permitting", "Reliability" ,"Costs")
Criteria.names.main.obj <- c("Ecosystem", "Improve wastewater", "Intergen.
Equity", "Social support", "Low costs")
SH.weights.added.full<- read.csv("Inputs/Stakeholder weights.csv")
SH.weights.added<- SH.weights.added.full [1:9, 17:21]

#plot stakeholder points for each criteria (criteria on x axis, points on y
axis)
par(las=2)
par(mar=c(9,4.1,4.1,2.1))
boxplot(SH.points[,4:16],
```

```
        names= c("Water Quality", "Wetland Habitat", "Water Supply",
"Nutrient Recovery", "CECs",
                 "Ease of Adaptation", "Sea Level Rise", "CO2 Emissions",
"Ease of Use",
                 "Shoreline Access", "Permitting", "Reliability" ,"Costs"))

title("Stakeholder points for improvement of each criteria from worst to best
state", cex.main = 1.5,
      ylab = "Stakeholder Preferences (Points)", cex.lab = 1.5)

boxplot(SH.points[,17:21],
        names = c("Ecosystem", "Improve wastewater", "Intergen. Equity",
"Social support", "Low costs"))
title(" Stakeholder average points of objectives within each category",
cex.main = 1.5,
      ylab = "Stakeholder Preferences (Points)", cex.lab = 1.5)

# show in order from highest to lowest median
SH.medians <- apply(SH.points [, 4:16], 2, median, na.rm = T)
point.orders<-order(SH.medians, decreasing = T)

SH.medians.main.obj <- apply(SH.points [, 17:21], 2, median, na.rm = T)
point.orders.main.obj<-order(SH.medians.main.obj, decreasing = T)

par(las=2)
par(mar=c(9,4.1,4.1,2.1))
boxplot(SH.points[,point.orders+3],
        names= Criteria.names[point.orders])

title("Stakeholder points for improvement of each criteria from worst to best
state", cex.main = 1.5,
      ylab = "Stakeholder Preferences (Points)", cex.lab = 1.5)

#Medians of overarching objective categories

par(las=2)
par(mar=c(9,4.1,4.1,2.1))
boxplot(SH.points[,point.orders.main.obj+16],
        names= Criteria.names.main.obj[point.orders.main.obj])

title("Average stakeholder points for criteria within each category",
cex.main = 1.5,
      ylab = "Stakeholder Preferences (Points)", cex.lab = 1.5)

#create weights from the points
point.sums <- apply(SH.points [,4:16], 1, sum, na.rm = T)
SH.weights <- SH.points [,4:16]/ point.sums
apply(SH.weights, 1, sum, na.rm = T) # check that all rows add to 1

  #and for main objectives
point.sums.main.obj <- apply(SH.points [,17:21], 1, sum, na.rm = T)
SH.weights.main.obj <- SH.points [,17:21]/ point.sums.main.obj
apply(SH.weights.main.obj, 1, sum, na.rm = T)

#plot the weights

par(las=2)
```

```
par(mar=c(9,4.1,4.1,2.1))
boxplot(SH.weights, range = 0,
        names= c("Water Quality", "Wetland Habitat", "Water Supply",
"Nutrient Recovery", "CECs",
                 "Ease of Adaptation", "Sea Level Rise", "CO2 Emissions",
"Ease of Use",
                 "Shoreline Access", "Permitting", "Reliability" ,"Costs"),
cex.lab = 1.5)

title("Average stakeholder weights for improvement of each criteria from
worst to best state", cex.main = 1.5,
      ylab = "Stakeholder Preferences (weights)", cex.lab = 1.5)

  #and for main objectives
par(las=2)
par(mar=c(9,4.1,4.1,2.1))
boxplot(SH.weights.main.obj, range = 0,
        names= Criteria.names.main.obj, cex.lab = 1.5)

title("Average stakeholder weights for criteria within each category",
cex.main = 1.5,
      ylab = "Stakeholder Preferences (weights)", cex.lab = 1.5)

# for added main objectives
boxplot(SH.weights.added, range = 0,
        names= Criteria.names.main.obj, cex.lab = 1.5)
title("Sum of stakeholder weights for criteria in each category", cex.main =
1.5,
      ylab = "Stakeholder Preferences (weights)", cex.lab = 1.5)

#plot boxplots from the weights, ordered from highest to lowest median
SH.medians.weights <- apply(SH.weights, 2, median, na.rm = T)
weight.orders<-order(SH.medians.weights, decreasing = T)
par(las=2)
par(mar=c(12,6,4.1,2.1), mgp=c(4,1,0))
boxplot(SH.weights[,weight.orders],
        names= Criteria.names[weight.orders], cex.axis = 1.5)

title("Stakeholder weights for each criteria", cex.main = 1.5,
      ylab = "Stakeholder Preferences (weights)", cex.lab = 1.5)

  #and for main objectives
SH.medians.weights.main.obj <- apply(SH.weights.main.obj, 2, median, na.rm =
T)
weight.orders.main.obj<-order(SH.medians.weights.main.obj, decreasing = T)
par(las=2)
par(mar=c(13,6,4.1,2.1), mgp=c(4,1,0))
boxplot(SH.weights.main.obj[,weight.orders.main.obj],
        names= Criteria.names.main.obj[weight.orders.main.obj], cex.axis =
1.5)

title("Average stakeholder weights for criteria within each category",
cex.main = 1.5,
      ylab = "Stakeholder Preferences (weights)", cex.lab = 1.5)

#and for added main objectives
SH.medians.weights.added <- apply(SH.weights.added, 2, median, na.rm = T)
```

```
weight.orders.added<-order(SH.medians.weights.added, decreasing = T)
par(las=2)
par(mar=c(13,6,4.1,2.1), mgp=c(4,1,0))
boxplot(SH.weights.added[,weight.orders.added],
        names= Criteria.names.main.obj[weight.orders.added], cex.axis = 1.5)

title("Sum of stakeholder weights for criteria within each category",
cex.main = 1.5,
      ylab = "Stakeholder Preferences (weights)", cex.lab = 1.5)
```

Calculate results of MCDA for each stakeholder without brineline option.

```
#SH1
#this one has the code to be able to see sub-node scores for each option
option.scores.unc.no.brine.1 <- matrix(ncol = 9, nrow = n) #n is specified in
the chunk on Monte Carlo simulations of uncertainty
colnames(option.scores.unc.no.brine.1) <- options.no.brine
v.eval.unc.1<-list(NULL)

for (i in 1:9){# 9 is number of options
  v.eval.unc.1[[i]] <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH1))
  option.scores.unc.no.brine.1[,i]<- v.eval.unc.1[[i]][,1]
}
option.scores.unc.no.brine.1
v.eval.unc.1[[6]]
apply(v.eval.unc.1[[5]],2, mean)

##loop with just final scores
for (i in 1:9){# 9 is number of options
  v.eval.unc.1 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH1))
  v.eval.unc.1
  option.scores.unc.no.brine.1[,i]<- v.eval.unc.1 [,1]
}


option.scores.unc.no.brine.1

#find ranking for each option in each row
library(plyr)

Ranking.options.unc.no.brine.1 <- option.scores.unc.no.brine.1
rank(1-option.scores.unc.no.brine.1 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.1 [i,]<-rank(1-option.scores.unc.no.brine.1
[i,])
}

Ranking.options.unc.no.brine.1
head(Ranking.options.unc.no.brine.1)

#Plot rankings
```

```
rank.counts.no.brine.1 <- matrix(rep(0,81), ncol = 9) # 9 is number of
options, 81 is ncol squared
count.test.no.brine.1<-NULL
for(i in 1:9){# the 9 is number of options
  count.test.no.brine.1<-count(Ranking.options.unc.no.brine.1, i)

rank.counts.no.brine.1[as.numeric(as.matrix(count.test.no.brine.1[,1])),i]<-
count.test.no.brine.1[,2]
}
rownames(rank.counts.no.brine.1)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.1)<-options.no.brine
rank.counts.no.brine.1

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.1 <- rank.counts.no.brine.1/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12),
    oma = c(0,0,0,0),
    xpd = T)
barplot(t(rank.counts.prob.no.brine.1), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH1", ylab = "Probability of each
rank", cex.lab = .7, density = 60 , angle=c(120,45,90,11,270, 200, 80, 140,
20) )
legend("topright", inset = c(-.7,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n", density = 60 ,
angle=c(120,45,90,11,270, 200, 80, 140, 20))
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)


#SH2
option.scores.unc.no.brine.2 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.2) <- options.no.brine
for (i in 1:9){
  v.eval.unc.2 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH2))
  v.eval.unc.2
  option.scores.unc.no.brine.2[,i]<- v.eval.unc.2 [,1]
}

option.scores.unc.no.brine.2
head(option.scores.unc.no.brine.2)

## do summary stats

apply(option.scores.unc.no.brine.2, 2, mean)
boxplot(option.scores.unc.no.brine.2)

#find ranking for each option in each row

Ranking.options.unc.no.brine.2 <- option.scores.unc.no.brine.2
rank(1-option.scores.unc.no.brine.2 [1,])
```

```
for (i in 1:n){
  Ranking.options.unc.no.brine.2 [i,]<-rank(1-option.scores.unc.no.brine.2
[i,])
}

Ranking.options.unc.no.brine.2
head(Ranking.options.unc.no.brine.2)

## Plot rankings

rank.counts.no.brine.2 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.2<-NULL
for(i in 1:9){
  count.test.no.brine.2<-count(Ranking.options.unc.no.brine.2, i)

rank.counts.no.brine.2[as.numeric(as.matrix(count.test.no.brine.2[,1])),i]<-
count.test.no.brine.2[,2]
}
rownames(rank.counts.no.brine.2)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.2)<-options.no.brine
rank.counts.no.brine.2

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.2 <- rank.counts.no.brine.2/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
barplot(t(rank.counts.prob.no.brine.2), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH2", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)



#SH3
option.scores.unc.no.brine.3 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.3) <- options.no.brine
for (i in 1:9){
  v.eval.unc.3 <- evaluate(nutrients.sowatt.midnodes.SH3.u, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH3))
  v.eval.unc.3
  option.scores.unc.no.brine.3[,i]<- v.eval.unc.3 [,1]
}

option.scores.unc.no.brine.3
head(option.scores.unc.no.brine.3)

## do summary stats
```

```
apply(option.scores.unc.no.brine.3, 2, mean)
boxplot(option.scores.unc.no.brine.3)

#find ranking for each option in each row

Ranking.options.unc.no.brine.3 <- option.scores.unc.no.brine.3
rank(1-option.scores.unc.no.brine.3 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.3 [i,]<-rank(1-option.scores.unc.no.brine.3
[i,])
}

Ranking.options.unc.no.brine.3
head(Ranking.options.unc.no.brine.3)

## Plot rankings

rank.counts.no.brine.3 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.3<-NULL
for(i in 1:9){
  count.test.no.brine.3<-count(Ranking.options.unc.no.brine.3, i)

rank.counts.no.brine.3[as.numeric(as.matrix(count.test.no.brine.3[,1])),i]<-
count.test.no.brine.3[,2]
}
rownames(rank.counts.no.brine.3)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 3", "Rank 3", "Rank 9")
colnames(rank.counts.no.brine.3)<-options.no.brine
rank.counts.no.brine.3

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.3 <- rank.counts.no.brine.3/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), oma = c(0,0,0,0), xpd = T)
barplot(t(rank.counts.prob.no.brine.3), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH3", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)

#SH4
option.scores.unc.no.brine.4 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.4) <- options.no.brine
for (i in 1:9){
  v.eval.unc.4 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH4))
  v.eval.unc.4
  option.scores.unc.no.brine.4[,i]<- v.eval.unc.4 [,1]
}
```

```
option.scores.unc.no.brine.4
head(option.scores.unc.no.brine.4)

## do  summary stats

apply(option.scores.unc.no.brine.4, 2, mean)
boxplot(option.scores.unc.no.brine.4)

#find ranking for each option in each row

Ranking.options.unc.no.brine.4 <- option.scores.unc.no.brine.4
rank(1-option.scores.unc.no.brine.4 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.4 [i,]<-rank(1-option.scores.unc.no.brine.4
[i,])
}

Ranking.options.unc.no.brine.4
head(Ranking.options.unc.no.brine.4)

## Plot rankings


rank.counts.no.brine.4 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.4<-NULL
for(i in 1:9){
  count.test.no.brine.4<-count(Ranking.options.unc.no.brine.4, i)

rank.counts.no.brine.4[as.numeric(as.matrix(count.test.no.brine.4[,1])),i]<-
count.test.no.brine.4[,2]
}
rownames(rank.counts.no.brine.4)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.4)<-options.no.brine
rank.counts.no.brine.4

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.4 <- rank.counts.no.brine.4/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
barplot(t(rank.counts.prob.no.brine.4), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH4", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)

#SH5

option.scores.unc.no.brine.5 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.5) <- options.no.brine
```

```
for (i in 1:9){
  v.eval.unc.5 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH5))
  v.eval.unc.5
  option.scores.unc.no.brine.5[,i]<- v.eval.unc.5 [,1]
}

option.scores.unc.no.brine.5
head(option.scores.unc.no.brine.5)

## do summary stats
apply(option.scores.unc.no.brine.5, 2, mean)
boxplot(option.scores.unc.no.brine.5)

#find ranking for each option in each row

Ranking.options.unc.no.brine.5 <- option.scores.unc.no.brine.5
rank(1-option.scores.unc.no.brine.5 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.5 [i,]<-rank(1-option.scores.unc.no.brine.5
[i,])
}

Ranking.options.unc.no.brine.5
head(Ranking.options.unc.no.brine.5)

## Plot rankings

rank.counts.no.brine.5 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.5<-NULL
for(i in 1:9){
  count.test.no.brine.5<-count(Ranking.options.unc.no.brine.5, i)

rank.counts.no.brine.5[as.numeric(as.matrix(count.test.no.brine.5[,1])),i]<-
count.test.no.brine.5[,2]
}
rownames(rank.counts.no.brine.5)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.5)<-options.no.brine
rank.counts.no.brine.5

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.5 <- rank.counts.no.brine.5/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
barplot(t(rank.counts.prob.no.brine.5), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH5", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)
```

```
#SH6

option.scores.unc.no.brine.6 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.6) <- options.no.brine
for (i in 1:9){
  v.eval.unc.6 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH6))
  v.eval.unc.6
  option.scores.unc.no.brine.6[,i]<- v.eval.unc.6 [,1]
}

option.scores.unc.no.brine.6
head(option.scores.unc.no.brine.6)

## do summary stats
apply(option.scores.unc.no.brine.6, 2, mean)
boxplot(option.scores.unc.no.brine.6)

#find ranking for each option in each row
Ranking.options.unc.no.brine.6 <- option.scores.unc.no.brine.6
rank(1-option.scores.unc.no.brine.6 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.6 [i,]<-rank(1-option.scores.unc.no.brine.6
[i,])
}

Ranking.options.unc.no.brine.6
head(Ranking.options.unc.no.brine.6)

## Plot rankings
rank.counts.no.brine.6 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.6<-NULL
for(i in 1:9){
  count.test.no.brine.6<-count(Ranking.options.unc.no.brine.6, i)

rank.counts.no.brine.6[as.numeric(as.matrix(count.test.no.brine.6[,1])),i]<-
count.test.no.brine.6[,2]
}
rownames(rank.counts.no.brine.6)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.6)<-options.no.brine
rank.counts.no.brine.6

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.6 <- rank.counts.no.brine.6/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
barplot(t(rank.counts.prob.no.brine.6), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH6", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
```

```
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)

#SH7
option.scores.unc.no.brine.7 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.7) <- options.no.brine
for (i in 1:9){
  v.eval.unc.7 <- evaluate(nutrients.sowatt.midnodes.SH7.u, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH7))
  v.eval.unc.7
  option.scores.unc.no.brine.7[,i]<- v.eval.unc.7 [,1]
}

option.scores.unc.no.brine.7
head(option.scores.unc.no.brine.7)

## do summary stats
apply(option.scores.unc.no.brine.7, 2, mean)
boxplot(option.scores.unc.no.brine.7)

#find ranking for each option in each row
Ranking.options.unc.no.brine.7 <- option.scores.unc.no.brine.7
rank(1-option.scores.unc.no.brine.7 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.7 [i,]<-rank(1-option.scores.unc.no.brine.7
[i,])
}

Ranking.options.unc.no.brine.7
head(Ranking.options.unc.no.brine.7)

## Plot rankings
rank.counts.no.brine.7 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.7<-NULL
for(i in 1:9){
  count.test.no.brine.7<-count(Ranking.options.unc.no.brine.7, i)

rank.counts.no.brine.7[as.numeric(as.matrix(count.test.no.brine.7[,1])),i]<-
count.test.no.brine.7[,2]
}
rownames(rank.counts.no.brine.7)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.7)<-options.no.brine
rank.counts.no.brine.7

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.7 <- rank.counts.no.brine.7/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
barplot(t(rank.counts.prob.no.brine.7), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH7", ylab = "Probability of each
rank" )
```

```
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)

#SH8
option.scores.unc.no.brine.8 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.8) <- options.no.brine
for (i in 1:9){
  v.eval.unc.8 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH8))
  v.eval.unc.8
  option.scores.unc.no.brine.8[,i]<- v.eval.unc.8 [,1]
}

option.scores.unc.no.brine.8
head(option.scores.unc.no.brine.8)

## do summary stats
apply(option.scores.unc.no.brine.8, 2, mean)
boxplot(option.scores.unc.no.brine.8)

#find ranking for each option in each row
Ranking.options.unc.no.brine.8 <- option.scores.unc.no.brine.8
rank(1-option.scores.unc.no.brine.8 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.8 [i,]<-rank(1-option.scores.unc.no.brine.8
[i,])
}

Ranking.options.unc.no.brine.8
head(Ranking.options.unc.no.brine.8)

## Plot rankings
rank.counts.no.brine.8 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.8<-NULL
for(i in 1:9){
  count.test.no.brine.8<-count(Ranking.options.unc.no.brine.8, i)

rank.counts.no.brine.8[as.numeric(as.matrix(count.test.no.brine.8[,1])),i]<-
count.test.no.brine.8[,2]
}
rownames(rank.counts.no.brine.8)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.8)<-options.no.brine
rank.counts.no.brine.8

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.8 <- rank.counts.no.brine.8/n

par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
```

```
barplot(t(rank.counts.prob.no.brine.8), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH8", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)

#SH9
option.scores.unc.no.brine.9 <- matrix(ncol = 9, nrow = n)
colnames(option.scores.unc.no.brine.9) <- options.no.brine
for (i in 1:9){
  v.eval.unc.9 <- evaluate(nutrients.uncertainty.midnodes, attrib =
data.frame(option.list.no.brine[[i]]), par = unlist(pars.nutrients.SH9))
  v.eval.unc.9
  option.scores.unc.no.brine.9[,i]<- v.eval.unc.9 [,1]
}

option.scores.unc.no.brine.9
head(option.scores.unc.no.brine.9)

## do summary stats
apply(option.scores.unc.no.brine.9, 2, mean)
boxplot(option.scores.unc.no.brine.9)

#find ranking for each option in each row
Ranking.options.unc.no.brine.9 <- option.scores.unc.no.brine.9
rank(1-option.scores.unc.no.brine.9 [1,])

for (i in 1:n){
  Ranking.options.unc.no.brine.9 [i,]<-rank(1-option.scores.unc.no.brine.9
[i,])
}

Ranking.options.unc.no.brine.9
head(Ranking.options.unc.no.brine.9)

## Plot rankings
rank.counts.no.brine.9 <- matrix(rep(0,81), ncol = 9)
count.test.no.brine.9<-NULL
for(i in 1:9){
  count.test.no.brine.9<-count(Ranking.options.unc.no.brine.9, i)

rank.counts.no.brine.9[as.numeric(as.matrix(count.test.no.brine.9[,1])),i]<-
count.test.no.brine.9[,2]
}
rownames(rank.counts.no.brine.9)<- c("Rank 1", "Rank 2", "Rank 3", "Rank 4",
"Rank 5", "Rank 6", "Rank 7", "Rank 8", "Rank 9")
colnames(rank.counts.no.brine.9)<-options.no.brine
rank.counts.no.brine.9

#to set as probabilities, divide each value by n
rank.counts.prob.no.brine.9 <- rank.counts.no.brine.9/n
```

```
par(mfrow = c(1,1))
par(mar=c(5.1,4.1,4.1,12), xpd = T)
barplot(t(rank.counts.prob.no.brine.9), col = brewer.pal(10, "Paired"), main
= "Probability of ranks for each option, SH9", ylab = "Probability of each
rank" )
legend("topright", inset = c(-.6,0), legend = options.no.brine, col =
brewer.pal(10, "Paired"), pch = 15, bty = "n")
mtext(paste("Population change", sep = "=", scenario[1]), cex = .5)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .5, side = 1,
line = 3)

mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .5, side
= 1, line = 2)
colors()

#vector of resulting scores with uncertainty for each SH
SH.eval.unc.all <- data.frame(cbind (v.eval.unc.1, v.eval.unc.2,
v.eval.unc.3, v.eval.unc.4, v.eval.unc.5, v.eval.unc.6, v.eval.unc.7,
v.eval.unc.8, v.eval.unc.9))
SH.eval.unc.normal <- SH.eval.unc.all[c(1,2,4:6,8,9),]
```

Plot the top three ranked options for each stakeholder.

```
par(mfrow = c(3,3))
par(oma = c(5,4,4,12),
    mar = c(2,2,2,2),
    xpd = NA,
    las = 1)

barplot(t(rank.counts.prob.no.brine.1)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH1", ylab = "Probability of each rank")
barplot(t(rank.counts.prob.no.brine.2)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH2", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.3)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH3", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.4)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH4", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.5)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH5", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.6)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH6", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.7)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH7", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.8)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH8", ylab = "Probability of each rank" )
barplot(t(rank.counts.prob.no.brine.9)[,1:3], col = brewer.pal(10, "Paired"),
main = "SH9", ylab = "Probability of each rank" )

legend(x=3.4,y=3, inset = c(-1, -.2), y.intersp = .5, legend =
options.no.brine, col = brewer.pal(9, "Paired"), pch = 15, bty = "n", pt.cex
= 3)
mtext(paste("Population change", sep = "=", scenario[1]), side = 1, outer =
TRUE, line = 1, cex = .8)
mtext(paste("Climate effect", sep = "=", scenario [2]), cex = .8, side = 1,
outer = TRUE, line = 2)
```

```
mtext(paste("Ecological threshold", sep = "=", scenario [3]), cex = .8, side
= 1, outer = TRUE, line = 3)
mtext("Top 3 Ranked Options for Each Stakeholder", cex = 1.2, side = 3, outer
= TRUE, line = 1, font=2)
```

Plot overall value of options for stakeholders.

```
#plots of median overall value for all options for each stakeholder (Status
quo scenario)

  ##make dataframes of the median overall values
option.scores.unc.no.brine.medians.sq <-
data.frame(lapply(option.scores.unc.no.brine.list.sq, apply,2,median))
colnames(option.scores.unc.no.brine.medians.sq) <- SHs

##status quo scenario
par(mfrow = c(2,1))
par(oma = c(1,2,1,1),
    mar = c(2,3,2,1),
    xpd = NA)
plot(100,100, xlim=c(1,9), xaxt = 'n', xlab = '', ylab = "Median Overall
Value", ylim = c(0,1), cex.lab = 1.5, main = "Status Quo Scenario", cex.main
= 1.5)
axis(1, at=1:9, labels = SHs)
for(i in 1:length(option.scores.unc.no.brine.medians.sq[,1])){
points(1:length(option.scores.unc.no.brine.medians.sq[,1]),
option.scores.unc.no.brine.medians.sq[i,], col = brewer.pal(9, "Paired")[i],
pch = 16, lwd = 2)
lines(1:length(option.scores.unc.no.brine.medians.sq[,1]),
option.scores.unc.no.brine.medians.sq[i,], col = brewer.pal(9, "Paired")[i],
pch = 16, lwd = 2)
}
par(mar=c(0,0,0,0))
plot(1, type = "n", axes=FALSE, xlab="", ylab="")

legend(x = "topleft",inset = 0,
       legend = options.no.brine,
       col = brewer.pal(9, "Paired"), lwd=5, cex=1, horiz = F, ncol = 3, bty
= "n")
```