```
library(rgdal)
library(data.table)
library(readxl)
library(prioritizr)
library(prioritizrdata)

# load planning unit data
data(salt_pu)

# create equal cost data
cost_equal <- salt_pu
cost_equal[cost_equal] <- 1

# load conservation feature data
data(salt_features)


# minimum set
p1 <- problem(salt_pu, salt_features) %>%
  add_min_set_objective() %>%
  add_relative_targets(0.17) %>%
  add_binary_decisions() %>%
  add_default_solver(gap = 0)

# print problem
print(p1)

# solve the problem
s1 <- solve(p1)

plot(s1)



budget <- sum(salt_pu[intersecting_units(salt_pu, s1)])
budget2 <- sum(cost_equal[intersecting_units(cost_equal, s1)])

# maximum utility equal weights
wgts <- rep(1, nlayers(salt_features))
names(wgts) <- names(salt_features)
p2 <- problem(salt_pu, salt_features) %>%
  add_max_utility_objective(budget) %>%
  add_feature_weights(wgts) %>%
  add_binary_decisions() %>%
  add_default_solver(gap = 0)
s2 <- solve(p2, force = TRUE)
plot(s2)


# maximum utility high forest weight
```

```
wgts
wgts <- c(10, 1, 1, 1, 1)

p3 <- problem(salt_pu, salt_features) %>%
  add_max_utility_objective(budget) %>%
  add_feature_weights(wgts) %>%
  add_binary_decisions() %>%
  add_default_solver(gap = 0)
s3 <- solve(p3, force = TRUE)
plot(s3)


# maximum utility high forest weight equal cost
p4 <- problem(cost_equal, salt_features) %>%
  add_max_utility_objective(budget2) %>%
  add_feature_weights(wgts) %>%
  add_binary_decisions() %>%
  add_default_solver(gap = 0)
s4 <- solve(p4, force = TRUE)
plot(s4)
```