# Simultaneous Adversarial Knapsack

*Ben Ewing*

*November 25, 2018*

## Setup

Libraries.

```
# For data manipulation
library(dplyr)
library(purrr)
library(tidyr)
# For kurtosis and skew functions
library(e1071)
# RcppAlgos::comboGeneral is ~80x faster than utils::combn in my quick test, it also seems to beat
# data.table::CJ. It can also run in parallel, but this doesn't seem to help on Windows. Might be
# worth trying on the Econ linux machines?
library(RcppAlgos)
# For compiling functions to bytecode, this usually nets a tiny speedup
library(compiler)
# For linear programming
library(ROI)
library(ROI.plugin.glpk)
library(ompr)
library(ompr.roi)
```

Load knapsack functions.

```
source("../r/knapsack_functions_mat.R")
```

Simulation settings.

```
# number of draws from each items distribution, used to estimated utility of the item
n_draws <- 1000
```

## Premise

There are $n$ players (indexed by $i$), each with budget $b_i$. There are $m$ items (indexed by $j$) with price $p_j$ and value $v_j$, which is a random variable with a known distribution. Unless otherwise stated, we assume that all $v_j$ are uncorrelated. Any additional beliefs players hold about an item's value are captured in an error term $\epsilon_{i,j}$. Unless otherwise stated, we assume the error for all players and items has mean 0.

The player who forms the knapsack with the highest value wins. Players only care about winning: utility of 1 for winning, 0 otherwise. Players form knapsacks simultaneously, and values are only revealed after formation.

## Case 1

- $n = 2$ (two players)
  - $b_i = 1$ for all $i$ (all have a budget of one)

- $m > 1$ (more than one item)
  - all $v_j$ are equal in distribution (all items are the same)
  - $p_j = 1$ for all $j$ (all items have a price of one)

```
n_players <- 2
knap_size <- 1

tmp <- matrix(c(
  rnorm(n_draws, 0, 1),
  rnorm(n_draws, 0, 1)
), ncol = 2) %>%
  form_knapsacks(knap_size) %>%
  estimate_knapsack_utility(n_players)

tmp %>%
  group_by(knap_id) %>%
  summarise_if(is.numeric, mean) %>%
  arrange(utility)
```

```
## # A tibble: 2 x 9
##   knap_id n_tie_strats win_n win_prop    mean    sd skewness kurtosis
##     <int>        <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>    <dbl>
## 1       1        0.667  162     0.162 -0.0470  1.04  -0.0222    0.137
## 2       2        0.667  171.    0.171  0.0120  1.01   0.102     0.245
## # ... with 1 more variable: utility <dbl>
```

Because all items are equally likely to take on the same value, players will just try not to choose the same item. In other words, players will play a mixed strategy where each item has an equal chance of being selected: $\frac{1}{m}$.

## Case 2

- $n = 2$ (two players)
  - $b_i = 1$ for all $i$ (all have a budget of one)
- $m > 1$ (more than one item)
  - All $v_j have the same expectation, different variance, and are symmetric.
  - $p_j = 1$ for all $j$ (all items have a price of one)

```
n_players <- 2
knap_size <- 1

tmp <- matrix(c(
  rnorm(n_draws, 0, 1),
  rnorm(n_draws, 0, 3)
), ncol = 2) %>%
  form_knapsacks(knap_size) %>%
  estimate_knapsack_utility(n_players)

tmp %>%
  group_by(knap_id) %>%
  summarise_if(is.numeric, mean) %>%
  arrange(utility)
```

```
## # A tibble: 2 x 9
##   knap_id n_tie_strats win_n win_prop    mean    sd skewness kurtosis
##     <int>        <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>    <dbl>
## 1       2        0.667  160.    0.160 -0.0287  2.96   0.138    -0.226
## 2       1        0.667  173     0.173  0.0250  1.02   0.0824    0.311
## # ... with 1 more variable: utility <dbl>
```

With two players, variance does not matter so long as expectation is approximately the same. Players will again play a mixed strategy in an attempt to avoid choosing the same items.

## Case 3

- $n = 2$ (two players)
  - $b_i = 1$ for all $i$ (all have a budget of one)
- $m > 1$ (more than one item)
  - All $v_j have different expectation, different variance, and are symmetric.
  - $p_j = 1$ for all $j$ (all items have a price of one)

```
n_players <- 2
knap_size <- 1

tmp <- matrix(c(
  rnorm(n_draws, 0, 1),
  rnorm(n_draws, 5, 3),
  rnorm(n_draws, 10, 5)
), ncol = 3) %>%
  form_knapsacks(knap_size) %>%
  estimate_knapsack_utility(n_players)

tmp %>%
  group_by(knap_id) %>%
  summarise_if(is.numeric, mean) %>%
  arrange(utility)
```

```
## # A tibble: 3 x 9
##   knap_id n_tie_strats win_n win_prop    mean    sd skewness kurtosis
##     <int>        <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>    <dbl>
## 1       1          0.5   21    0.0210 5.77e-3 0.996 -0.00619   -0.157
## 2       2          0.5  284.   0.284  4.92e+0 3.02  -0.0434    -0.108
## 3       3          0.5  445.   0.445  1.00e+1 4.97  -0.0744    -0.171
## # ... with 1 more variable: utility <dbl>
```

```
strat_minimax_2(tmp)$solution
```

```
##   row_pr[1]   row_pr[2]   row_pr[3] row_utility
##           0           1           0           0
```

Players will choose the item with the highest expectation, variance does not matter.

## Case 4

- $n = 3$ (three players)
  - $b_i = 1$ for all $i$ (all have a budget of one)
- $m > n$ (more than one item)
  - All $v_j have the same expectation, different variance, and are symmetric.
  - $p_j = 1$ for all $j$ (all items have a price of one)

```
n_players <- 3
knap_size <- 1

tmp <- matrix(c(
  rnorm(n_draws, 5, 1),
  rnorm(n_draws, 5, 3),
  rnorm(n_draws, 5, 5),
  rnorm(n_draws, 5, 7),
  rnorm(n_draws, 5, 9)
), ncol = 5) %>%
  form_knapsacks(knap_size) %>%
  estimate_knapsack_utility(n_players)

tmp %>%
  group_by(knap_id) %>%
  summarise_if(is.numeric, mean) %>%
  arrange(utility)
```

```
## # A tibble: 5 x 9
##   knap_id n_tie_strats win_n win_prop  mean    sd skewness kurtosis utility
##     <int>        <dbl> <dbl>    <dbl> <dbl> <dbl>    <dbl>    <dbl>   <dbl>
## 1       1        0.667  347.    0.347  4.96  1.00  -0.132   0.310   -0.0956
## 2       2        0.667  367.    0.367  5.06  2.94   0.0371 -0.0417  -0.0394
## 3       3        0.667  390.    0.390  5.18  5.07  -0.0164 -0.198    0.0237
## 4       4        0.667  392.    0.392  5.09  6.97   0.0746  0.214    0.0341
## 5       5        0.667  409.    0.409  5.31  8.90  -0.0843 -0.00989  0.0771
```

```
strat_minimax_3(tmp)$solution
```

```
##   row_pr[1]   row_pr[2]   row_pr[3]   row_pr[4]   row_pr[5] row_utility
##   0.4504988   0.0000000   0.3863392   0.0000000   0.1631619  -0.1830450
```

No idea. It looks like players randomize over every item, except that with the highest expectation?

## Case 5

- $n = 3$ (three players)
  - $b_i = 1$ for all $i$ (all have a budget of one)
- $m > n$ (more than one item)
  - All $v_j have different expectation, different variance, and are symmetric.
  - $p_j = 1$ for all $j$ (all items have a price of one)

```
n_players <- 3
knap_size <- 1

tmp <- matrix(c(
  rnorm(n_draws, 5, 1),
  rnorm(n_draws, 6, 3),
  rnorm(n_draws, 7, 5),
  rnorm(n_draws, 8, 7),
  rnorm(n_draws, 9, 3)
), ncol = 5) %>%
  form_knapsacks(knap_size) %>%
  estimate_knapsack_utility(n_players)

tmp %>%
  group_by(knap_id) %>%
  summarise_if(is.numeric, mean) %>%
  arrange(utility)
```

```
## # A tibble: 5 x 9
##   knap_id n_tie_strats win_n win_prop  mean    sd skewness kurtosis utility
##     <int>        <dbl> <dbl>    <dbl> <dbl> <dbl>    <dbl>    <dbl>   <dbl>
## 1       1        0.667  188.    0.188  5.03 0.994   0.0328   0.260  -0.456
## 2       2        0.667  310.    0.310  6.00 2.94   -0.115   -0.0173 -0.171
## 3       3        0.667  394.    0.394  7.07 5.09   -0.0748  -0.0356  0.0333
## 4       4        0.667  459.    0.459  8.15 7.08   -0.145   -0.0637  0.168
## 5       5        0.667  552.    0.552  9.01 3.02   -0.0690  -0.0779  0.372
```

```
strat_minimax_3(tmp)$solution
```

```
##   row_pr[1]   row_pr[2]   row_pr[3]   row_pr[4]   row_pr[5] row_utility
##  0.00000000  0.00000000  0.00000000  0.93567251  0.06432749 -0.18850000
```