

TP3: Randonnée et optimisation

Evan Voyles

May 25

```
library(tidyverse)
library(purrr)
library(tibble)
library(ggplot2)
library(pracma)
```

Un randonneur se trouve face à un dilemme classique : il aimerait emmener avec lui un maximum d'affaires qui pourraient lui servir mais ne souhaite pas alourdir inutilement son sac, qui ne doit pas dépasser un certain poids P . Il cherche donc à déterminer la combinaison d'affaire qui vérifierait le meilleur rapport utilité/légereté. Pour ce faire, il associe à chacune de ses M affaires, son poids p_i et sa valeur v_i pour $i \in \{1, \dots, M\}$. L'objectif est de trouver la configuration qui maximise la valeur du sac à dos sans dépasser P .

1. On considère un vecteur σ (qui contient 0 ou 1 pour l'entrée i selon que l'objet i ait été choisi pour remplir le sac à dos). On choisit initialement un vecteur σ tel que

```
# A partir d'une configuration \sigma, proposer \sigma' en modifiant aleatoirement l'un des
# \sigma_i par (1 - \sigma_i)
propose_sigma_prime <- function(sig) {

  n <- length(sig)
  i <- sample(1:n, 1)

  sig[i] <- 1 - sig[i]
  sig
}

# Return the total weight of the items carried
total_weight <- function(sig, poids) {
  sum(sig * poids)
}

total_valeur <- function(sig, valeur) {
  sum(sig * valeur)
}

# We'll use the transition matrix  $Q(\cdot | V) = N(V, 1)$ 
# En sachant que  $\pi(\sigma') / \pi(\sigma) = \exp(V_{\{\sigma'\}} - V_{\{\sigma\}})$ ,
# On implemente l'algorithme de metropolis hastings comme suivant:
taux_accept <- function(sig, sig_prime, valeur) {
  min(exp(total_valeur(sig, valeur) - total_valeur(sig_prime, valeur)), 1)
}
```

```
item <- c("map", "water", "sandwich", "glucose", "tin", "banana", "apple", "cheese",
         "beer", "suntan_cream", "camera", "T-shirts", "trousers", "umbrella",
```

```

      "waterproof_trousers", "waterproof_overclothes", "note-case", "sunglasses", "towel",
      "socks", "book")
p <- c(0.05, 1, 0.7, 0.1, 0.5, 0.2, 0.3, 0.4, 0.5, 0.3, 1, 0.8, 0.4, 0.7, 0.4, 0.3, 0.4, 0.1, 0.3, 0.4,
v <- c(150, 300, 160, 60, 45, 60, 40, 30, 180, 70, 30, 100, 10, 40, 70, 75, 50, 80, 12, 50, 30)
P <- 7
n_it <- 1e3
n_items <- length(item)

next_sigma <- function(sig, poids, valeur, P) {

  sig_prime <- propose_sigma_prime(sig)

  if (total_weight(sig_prime, poids) > P) {
    return(sig)
  }

  r <- taux_accept(sig, sig_prime, valeur)

  if (runif(1) < r) {
    # Then we accept sig_prime
    return(sig)
  } else {
    # We keep sig
    return(sig_prime)
  }
}

sim_traj <- function(n) {
  sigma0 <- rep(0, n_items)
  traj <- list(sigma0)

  for (i in 2:n) {
    traj[[i]] <- next_sigma(traj[[i - 1]], p, v, P)
  }

  traj
}

sim_traj_valeur <- function(n) {

  traj <- sim_traj(n)
  val <- map_dbl(traj, total_valeur, v)

  list(t = traj, v = val)
}

val_traj <- sim_traj_valeur(1000)

# Get the n indice of the max value in a val_traj
get_max_val <- function(n) {

```

```

val_traj <- sim_traj_valeur(n)
max_val <- max(val_traj$v)
i_max <- detect_index(val_traj$v, function(x) { x == max_val })

  list(max = max_val, i = i_max, sigma = val_traj$t[[i_max]])
}

# Now re run this algorithm 100 times with n = 1e5.
# Figure out the max of all values

n_traj <- 1e3
n <- 1e2

max_val_traj <- list()

for (i in 1:n_traj) {
  max_val_traj[[i]] <- get_max_val(n)
}

# Now extrac the max values
max_vals <- map_dbl(max_val_traj, function(traj) { traj$max })
max_sigmas <- map(max_val_traj, function(traj) { traj$sigma })
max(max_vals)

```

```
## [1] 1532
```

```

# Consistently, the MAX value that I can get is 1532
# Let's find the index of 1532

# Function factory
equals <- function(k) {
  function(x) {
    x == k
  }
}

i_1532 <- detect_index(max_vals, equals(1532))

winning_configuration <- max_sigmas[[i_1532]]

total_valeur(winning_configuration, v)

```

```
## [1] 1532
```

```
total_weight(winning_configuration, p)
```

```
## [1] 6.75
```

```
item[as.logical(winning_configuration)]
```

```

## [1] "map"           "water"         "sandwich"
## [4] "glucose"       "tin"           "banana"
## [7] "apple"         "cheese"        "beer"
## [10] "suntan_cream"  "T-shirts"      "waterproof_trousers"
## [13] "waterproof_overclothes" "note-case"     "sunglasses"
## [16] "towel"         "socks"

```