Final project

Natalia Sarabia Vásquez

November 30, 2021

1 Project

Generate chromosomes and set the population matrix

2 Creating the generation

```
# Generate the genes and set the chromosomes:
create_population <- function(chromosome_length, population_size){
  n <- chromosome_length * population_size
  chromosome <- as.vector(sample(0:1, n, replace=TRUE))
  population <- as.data.frame(matrix(chromosome, nrow = population_size, ncol = chromosome_length))
  names(population) <- dput(paste0('gen_', seq(1,chromosome_length,1)))
  return(population)
}
```

3 Computing the fitness

```
# Function to identify the genes that would be active in each model given a chromosome:
find_genes <- function(chromosome, variables_names){
  variables <- variables_names[grep(1,as.vector(chromosome))]
  return(variables)
}

# Function to construct the formulas given the active genes of a chromosome:
set_formulas <- function(active_genes, name_y){
  formulas <- as.formula(paste(name_y, paste(active_genes, sep = "", collapse = " + "), sep = " ~ "))
  return(formulas)
}

# Function to compute the fitness (AIC) given a formula and a dataset
# By default, it will fit a linear regression. Although, it can receive the parameters
# for a generalized linear model
fitness <- function(formula, data, ...){
   fitness <- AIC(glm(formula = formula, data = data, ...))
   return(fitness)
}

# Compute the fitness of an entire generation:
# The result is sort by the fittest individual to the least fit
get_fitness <- function(X, name_y, generation){
```

```
  data_names <- names(X)[!names(X) %in% c(name_y)]
  variables <- apply(generation, 1, find_genes, data_names)
  formulas <- lapply(variables, set_formulas, name_y)
  fitness <- lapply(formulas, fitness, X)
  return(unlist(fitness))
}

# Be careful, this returns your generation sorted by fitness
gather_fitness_generation <- function(generation,fitness_scores){
  gathered <- cbind(generation,fitness_scores) %>%
  arrange(desc(fitness_scores))
  return(gathered)
}
```

4 Test the funtion

Data to try out the code:

```
set.seed(123)

# Number of genes per chromosome: number of  covariates in a linear model
chromosome_length <- 10

# Population size
# The paper recommends fo binary encoding of chromosomes to choose P to satisfy $C \leq P \leq 2C$
population_size <- sample(chromosome_length:(2*chromosome_length), 1, replace=TRUE)

my_generation <- create_population(chromosome_length, population_size)
```

```
## c("gen_1", "gen_2", "gen_3", "gen_4", "gen_5", "gen_6", "gen_7",
## "gen_8", "gen_9", "gen_10")
```

```
head(my_generation)
```

```
##   gen_1 gen_2 gen_3 gen_4 gen_5 gen_6 gen_7 gen_8 gen_9 gen_10
## 1     1     0     1     1     0     0     1     0     0      1
## 2     0     1     0     0     0     0     1     1     0      1
## 3     1     0     1     1     0     1     0     0     1      0
## 4     1     0     0     1     0     0     1     1     1      0
## 5     1     0     1     0     1     0     1     1     0      1
## 6     0     0     0     0     1     0     1     0     1      1
```

```
x <- as.data.frame(matrix(runif(100*(chromosome_length+1),0,1),ncol=(chromosome_length+1),nrow=100))
names(x) <- letters[1:(chromosome_length+1)]
head(x)
```

```
##           a          b          c         d          e          f         g
## 1 0.2197676 0.68637508 0.70399206 0.3219374 0.03736996 0.37681642 0.8719988
## 2 0.3694889 0.05284394 0.10380669 0.8911143 0.51880492 0.04210805 0.6078680
## 3 0.9842192 0.39522013 0.03372777 0.6262569 0.67901342 0.36441108 0.7562034
## 4 0.1542023 0.47784538 0.99940453 0.3029049 0.90323356 0.27375127 0.8472413
## 5 0.0910440 0.56025326 0.03487480 0.3882047 0.02552670 0.85046748 0.6127796
## 6 0.1419069 0.69826159 0.33839128 0.1604751 0.98907827 0.36240171 0.7932242
##           h         i         j         k
## 1 0.99103432 0.8565970 0.3947590 0.3135179
```

```
## 2 0.74320492 0.6979466 0.8112373 0.1383901
## 3 0.07585713 0.6844865 0.3185631 0.0580307
## 4 0.45116891 0.3480151 0.5814473 0.9310996
## 5 0.05353693 0.5546818 0.4554890 0.5683380
## 6 0.33955551 0.1372436 0.2688034 0.1177704
```

```
fitness_scores <- get_fitness(x,"a",my_generation)
head(fitness_scores)
```

```
## [1] 25.92196 22.38374 26.19539 21.74139 23.95598 27.13783
```

```
my_generation_info <- gather_fitness_generation(my_generation, fitness_scores)
head(my_generation_info)
```

```
##   gen_1 gen_2 gen_3 gen_4 gen_5 gen_6 gen_7 gen_8 gen_9 gen_10 fitness_scores
## 1     0     0     0     0     1     0     1     0     1      1       27.13783
## 2     1     1     0     1     1     0     0     0     1      0       26.69476
## 3     1     0     1     1     0     1     0     0     1      0       26.19539
## 4     0     1     1     0     0     0     1     1     1      1       25.95421
## 5     1     0     1     1     0     0     1     0     0      1       25.92196
## 6     0     0     0     0     1     0     1     1     1      1       24.59081
```

```
# fit one by lm
AIC(lm(a ~ f+h+j+k, data = x ))
```

```
## [1] 27.13783
```