3.0

```
library(stringi)
library(dplyr)

setwd("/home/bartoszukm/Sages/ml-1m/")

movies_text <- stri_read_lines("movies.dat")
movies_text <- stri_replace_all_fixed(movies_text, "::", ";")
stri_write_lines(movies_text, "movies_oneSep.dat")

movies <- read.csv2("movies_oneSep.dat", header=FALSE)
head(movies)
#movies <- movies[,-1]
#head(movies)
colnames(movies)<-c("movieID", "title", "category")
movies$movieID <- as.integer(as.character(movies$movieID))
head(movies)
# mozna zamieniac przy uzyciu pakietu stringi lub recznie - np w notatniku (znajdz i
# zamien) lub w bashu:
# sed 's/::/@/g' ratings.dat > wynik.dat
movies_text <- stri_read_lines("ratings.dat")
movies_text <- stri_replace_all_fixed(movies_text, "::", ";")
stri_write_lines(movies_text, "ratings_oneSep.dat")

ratings <- read.csv2("ratings_oneSep.dat", header=FALSE)
head(ratings)
colnames(ratings)<-c("userID", "movieID", "Rating", "Timestamp")
head(ratings)

movies_text <- stri_read_lines("users.dat")
movies_text <- stri_replace_all_fixed(movies_text, "::", ";")
stri_write_lines(movies_text, "users_oneSep.dat")

users <- read.csv2("users_oneSep.dat", header=FALSE)
head(users)
colnames(users)<-c("userID","Gender","Age","Occupation","Zip-code")
head(users)

#zadanie zero: rok filmu
#movies <- mutate(movies, year = stri_sub(title,-5,-2))
movies <- movies %>% mutate(year = as.numeric(stri_sub(title,-5,-2)))
#A ile jest wszystkich filmow
nrow(movies)
movies %>% summarise(licznosc = n())
#B
#zliczamy filmy po latach, sortujemy po liczbie wystapien
#obcinamy do 5 wynikow
movies %>% count(year) %>% arrange(desc(n)) %>% slice(1:5)
```

```r
#C
# grupujemy po płci i zliczamy
users %>% group_by(Gender) %>% summarise(count=n())
# lub zwyczajnie zliczamy po płci
users %>% count(Gender)

#D
head(movies)
kategorie <- unlist(stri_split_fixed(movies$category, "|"))
kategorieLicznosc <- sort(table(kategorie))

#E
# laczymy zbiory filmow i ocen - nie podajemy po jakim warunku ma się odbywać łączenie
# ale ponieważ oba zbiory mają kolumny "movieid" to funkcja inner_join sama wybiera
# właśnie tę kolumnę do połączenia zbiorów
# następnie przy pomocy select ograniczamy się do dwóch kolumn - tytuł i ocena
# później grupujemy po tytułach, w celu zagregowania ocen do średniej wartości
# oraz zliczenia liczby wystąpień (punkt F)
# sortujemy malejąco po średniej, odsiewamy te filmy, które mają mniej niż 100 ocen
# i zostawiamy tylko 100 najlepszych, które następnie zapisujemy do zmiennej ranking
inner_join(movies, ratings) %>% select(title, rank) %>%
  group_by(title) %>% summarise(srednia = mean(rank), ile=n()) %>%
  arrange(desc(srednia)) %>% filter(ile > 100) %>% slice(1:100) -> ranking
write.csv(ranking, "ranking.csv")

#rozwiązanie numer 2
#najpierw wszystkie oceny grupujemy po id filmu, następnie uśredniamy i zliczamy oceny
# dopiero po tym łączymy ze zbiorem filmów, zostawiamy tytuły i oceny, filtrujemy,
# sortujemy i zostawiamy top 100
ratings %>% group_by(movieid) %>%
  summarise(srednia = mean(rank), ile=n()) %>% inner_join(movies) %>%
  select(title, srednia, ile) %>% filter(ile > 100) %>%
  arrange(desc(srednia)) %>% slice(1:100)

# porównaj czas działaniu obu rozwiązań przy pomocy pakietu microbenchmark
# zastanów się dlaczego drugie rozwiązanie jest szybsze (na moim komputerze trzykrotnie)
microbenchmark(inner_join(movies, ratings) %>% select(title, rank) %>%
                 group_by(title) %>% summarise(srednia = mean(rank), ile=n()) %>%
                 arrange(desc(srednia)) %>% filter(ile > 100) %>% slice(1:100),

               ratings %>% group_by(movieid) %>%
                 summarise(srednia = mean(rank), ile=n()) %>% inner_join(movies) %>%
                 select(title, srednia, ile) %>% filter(ile > 100) %>%
                 arrange(desc(srednia)) %>% slice(1:100), times=10)


#G
#podejście nr 1 - podzielić dane według grup (kobiety/mężczyźni) i liczyć osobno średnie
#dla każdego z podzbiorów
movies_ratings_users <- dplyr::inner_join(movies_ratings, users, by=c("userID"="UserID"))
```

```r
head(movies_ratings_users)
ocenySrednie <- movies_ratings_users %>% group_by(movieID, Gender) %>%  summarise(ocena =
mean(Rating), count = n())
ocenySrednie <- ocenySrednie %>% filter(count >= 100)
ocenySrednieF <- ocenySrednie %>% filter(Gender=="F")
ocenySrednieF <- ocenySrednieF %>% ungroup() %>% arrange(desc(ocena))
print(dplyr::inner_join(ocenySrednieF, movies, by="movieID"), n=40)

ocenySrednieM <- ocenySrednie %>% filter(Gender=="M")
ocenySrednieM <- ocenySrednieM  %>% ungroup() %>% arrange(desc(ocena))
print(dplyr::inner_join(ocenySrednieM, movies, by="movieID"), n=40)

# jednak jeśli grup będzie dużo to ręczne dzielenie tego na podzbiory będzie mało efektywne
# wtedy lepiej pogrupować po tej zmiennej:
# najpierw łączymy oceny z użytkownikami - aby wiedzieć która ocena była wystawiona
# przez kogo
# następnie grupujemy po DWÓCH zmiennych - płeć i id filmu - kolejność jest ważna
# możesz sprawdzić co się stanie jak odwrócimy
# dalej to już standard - grupowanie ocen, filtracja po ilości ocen
# sortowanie, wycinanie top 10 - wycinanie działa tutaj PER GRUPA
# a mamy dwie - F/M więc z każdej zostawia po 10 dlatego wynikowy zbiór ma 20 wierszy
ratings %>% inner_join(users) %>% group_by(Gender, movieid) %>%
  summarise(srednia = mean(rank), ile=n()) %>% filter(ile > 100) %>%
  arrange(desc(srednia)) %>% slice(1:10)%>% inner_join(movies) %>%
  select(title, srednia, Gender)


#H

#do użytkowników dołączam oceny, zliczam ile było w danej grupie wiekowej
#ocen dla danego id filmu, dołączam filmy, selecet ogranicza kolumny do istotnych
#kolejne grupowanie po grupie i roku, następnie liczenie średniej trochę
# "na piechotę" - bo nie ma obserwacji pojedynczo tylko są zagregowane
# 1921 rok był 3 razy, 1922 też 3 itd zamiast 1921, 1921, 1921, 1922, 1922...
users %>% inner_join(ratings) %>% count(Age, movieid) %>% inner_join(movies) %>%
  select(Age, n, year) %>% group_by(Age,year) %>% summarise(n=sum(n)) %>%
  mutate(iloczyn = n * year) %>% summarise(n=sum(n), iloczyn=sum(iloczyn)) %>%
  mutate(sredniRok = iloczyn/n) %>% select(Age, sredniRok)

# tutaj średnia liczona przy pomocy mean ale za to dwa cięższe joiny
# na nie zagregowanych danych - to rozwiązanie będzie wolniejsze im większe są zbiory
select(users, Age, userid) %>% inner_join(select(ratings, userid, movieid)) %>%
   inner_join(select(movies, movieid, year)) %>% group_by(Age) %>%
summarise(sredniRok=mean(year))

#I
#najpierw nalezy "rozciągnąć" ramkę danych
expand_row <- function(x){
  expand <- strsplit(x[3], "\\|")
  num <- length(expand)
```

```
  movieid <- rep(x[1],num)
  return(data.frame(movieid=as.integer(movieid),genre=expand[[1]], row.names = NULL))
}
expanded <- apply(movies,1,expand_row)
df <- do.call("rbind", expanded)

# do ramki dołączamy oceny i użytkowników, zliczamy częstości, grupujemy po gatunku
# aby w ramach grup posortować malejąco po częstości (n) i ograniczyć się do trzech
# rekordów per grupa
df %>% inner_join(ratings) %>% select(genre, userid) %>% inner_join(users) %>%
  select(genre, Age) %>% count(genre, Age) %>% group_by(genre) %>%  arrange(desc(n)) %>%
  slice(1:3)

#J
# podobnie jak wcześniej łączymy wszystkie trzy zbiory
# zliczamy ile było rekordów dla każdej z wartości par (gatunek, płeć)
# grupujemy po płci aby w grupach posortować i zwrócić tylko trzy wyniki
df %>% inner_join(ratings) %>% select(genre, userid) %>% inner_join(users) %>%
  select(genre, Gender) %>% count(genre, Gender) %>% group_by(Gender) %>%
  arrange(desc(n)) %>% slice(1:3)
```

3.1

```
rozwin <- function(macierz, nazwy)
{


  df1 <- data.frame(a = as.vector(macierz),
                    b = rep(dimnames(macierz)[[2]], each = nrow(macierz)),
                    c = rep(dimnames(macierz)[[1]], times = nrow(macierz)))
  colnames(df1) <- nazwy
  df1
}

rozwin(WorldPhones, c("co", "gdzie", "kiedy"))
```

3.2

```
zwin <- function(ramka)
{
  mat <- matrix(ramka[,1], ncol = length(levels(ramka[,2])), nrow = length(levels(ramka[,3])))
  colnames(mat) <- as.character(unique(ramka[,2]))
  rownames(mat) <- as.character(unique(ramka[,3]))
  mat
}
ramka <- rozwin(WorldPhones, c("co", "gdzie", "kiedy"))
zwin(ramka)
```

3.3

```
uniqueRows.loop<-function(df){
   #funkcja dziala poprawnie na posortowanych danych, tj. z zalozeniem,
   #ze powtarzajace sie wiersze wystepuja zawsze kolo siebie.
   stopifnot(is.data.frame(df))
   stopifnot(length(data.frame(df))>0)

   if(length(df)>=2){
   n<-length(df)
   vec<-numeric(nrow(df))
   previous<-0
   for(i in seq_along(df[,1])){
      nextt<-as.integer(df[i,])
      if(sum(previous==nextt)!=n) vec[i]<-row.names(df[i,])
      previous<-nextt}
   ndf<-na.omit(df[vec,])
   row.names(ndf)<-1:length(vec[vec!=0])
   }
   else{
   vec<-numeric(nrow(df))
   previous<-0
   for(i in seq_along(row(df))){
      nextt<-as.integer(df[[1]][i])
      if(sum(previous==nextt)!=1) vec[i]<-row.names(df)[i]
      previous<-nextt}
   ndf<-data.frame(df[[1]][row.names(df[1])==vec])
   names(ndf)<-names(df)
   }
   return(ndf)
}

uniqueRows<-function(df){
   #W tresci bylo tylko o nieuzywaniu funkcji duplicated(), wiec jest to
   #akceptowalne rozwiazanie zdaje sie? :) Jakby co, to jest moja glowna
   #funkcja: pozostale sa po to, jakby ta sie nie liczyla.
   #Pozdrawiam, KF.
   stopifnot(is.data.frame(df))

   mm<-unique(df)
   row.names(mm)<-1:nrow(mm)
   return(mm)
}

uniqueRows.split4.2cols <- function(df){
   #UWAGA, poniewaz funkcja ta bazuje na funkcji split(),
   #dziala ona tylko dla ramek o 2 kolumnach danych. Zaleta jest taka,
   #ze dziala znacznie szybciej niz inne rozwiazania (poza oczywiscie
   #uniqueRows)
   stopifnot(is.data.frame(df));
```

```
   stopifnot(length(df)==2)

   x<-df[[1]];y<-df[[2]]
   M<-lapply(split(y,x),union,NULL)  #grupowanie danych z drugiej ramki
                                     #wzgledem etykiet w pierwszej ramce
                                     #i usuniecie powtorzen (union)
   times<-unlist(lapply(M,length),use.names=FALSE)
   col1<-rep(names(M),times)
   col2<-unlist(M,use.names=F)
   newdf<-data.frame(col1,col2,row.names=seq_along(col2))
   names(newdf)<-names(df)
   return(newdf)
}

uniqueRows.split4many.cols<-function(df){
   #funkcja wykorzystuje fakt, ze sumowanie zbiorow usuwa powtarzajace
   #sie elementy w obu zbiorach, czyli podobnie jak poprzednia funkcja,
   #jednak tu probuje dostosowac ja dla ramek o wiecej, niz 2
   #kolumnach. Do tego potrzeba, by wszystkie kolumny byly
   #faktorami.
   stopifnot(is.data.frame(df))
   stopifnot(length(df)>1)
   stopifnot(all(sapply(df,is.factor)==T))

   d<-split(df,1:nrow(df))    #rozdziela poszczegolne wiersze listy.
                              #Zastosowanie jako drugiego argumentu
                              #w funkcji split wyrazenia: row.names(df)
                              #powodowalo nieprawidlowe sortowanie
                              #wierszy, tj. 1,10,11,2,3,...

   t<-lapply(d,as.integer)    #zamienia dane na wartosci liczbowe
   u<-simplify2array(union(t,NULL))
   w<-lapply(split(u,row(u)),factor)
   names(w)<-names(df)
   restore.levels<-function(x,y){   levels(x)<-levels(y)
                                    return(x)}
   ndf<-mapply(restore.levels,w,df)
   return(data.frame(ndf))
}
```

3.4

```
isNearlyOrthoMatrix <- function(m)
{
   stopifnot(is.matrix(m))
   mprod <- m %*% t(m)
   ind <- which(abs(mprod) > .Machine$double.eps, arr.ind = TRUE)
   all(ind[,1] == ind[,2])
}
```

3.5

```
doStochastycznej<-function(x){
   stopifnot(is.numeric(x), nrow(x)==ncol(x))
   wek<-rep(1, nrow(x))
   d<-as.vector(x%*%wek)
   stopifnot(is.matrix(x), d>0, x>=0)

   x/d
}

dostochastycznej <- function(x){
   stopifnot(is.matrix(x), nrow(x)==ncol(x),is.numeric(x),nrow(x)>0)
   #1 spr czy sa nieujemne
   war<-function(x){
      sum( x[x<0])
   }
   if (war(x)<0) stop(cat("wystepuja ujemne elementy\n"))

   #2 co najmniej jeden el >0 w kazdym wierszu
   if (!(sum(apply(x,1,range)[2,]>0)==ncol(x)))
      stop(cat("nie ma elemtnu dodatniego w wierszu\n"))

   #3 przeskalowanie,w kazdym wierszy szukam sumy i dziele, nie spr
   #warunku na dzielenie przez zero bo to wczensiejsze funkcje zapewniaja
   stochastyczna<- x/apply(x,1,sum)
   return(stochastyczna)
}
```

3.6

```
agregation<-function(df,kol,cz,fun,...)
{

   stopifnot(is.data.frame(df),length(as.vector(df))>=1, is.character(kol),
             is.character(cz),length(kol)==1,length(cz)==1,is.function(fun),
             any(names(df)==kol),any(names(df)==cz))

   n<-nrow(df)
   x<-df[cz==names(df)][1:n,]
   y<-df[kol==names(df)][1:n,]
   if(is.numeric(y)==FALSE)
      stop("Nie agregujemy wektora liczb")
   if(is.factor(x)==FALSE)
      stop("Czynnik nie jest typu factor")


   s<-tapply(y,x,fun,...)
```

```
    z<-cbind(df,s[x])
    names(z)[ncol(z)]<-paste(cz,"_",kol,"_",deparse(substitute(fun)),sep="")
    z
}
```