

Wstęp

Przedmiotem analizy będzie zbiór danych na temat gier planszowych. Zbiór pochodzi ze strony <https://www.kaggle.com/mrpantherson/board-game-data> Zbiór składa się z 20 kolumn oraz 4999 wierszy, każdy z nich dotyczy innej gry planszowej. Zbiór jest rankingiem tych gier na podstawie przyznanych im ocen. Spośród 20 kolumn, kilka postanowiłem odrzucić już na wstępie, ponieważ nie wnosiły żadnych istotnych pod kątem analizy informacji (np. link do strony internetowej.) Początkowy zbiór jest więc złożony z 13 kolumn

- **game_id**- unikatowy numer identyfikacyjny gry, zmienna numeryczna,
- **min_players**- minimalna liczba graczy, zmienna numeryczna,
- **max_players**- maksymalna liczba graczy, zmienna numeryczna,
- **avg_time**- przeciętny czas rozgrywki, zmienna numeryczna,
- **min_time**- minimalny czas rozgrywki, zmienna numeryczna,
- **max_time**- maksymalny czas rozgrywki, zmienna numeryczna,
- **year**- rok wydania gry, zmienna numeryczna,
- **avg_rating**- średnia ocena, zmienna numeryczna,
- **geek_rating**- średnia ocena specjalistów, zmienna numeryczna,
- **num_votes**- ilość oddanych głosów, zmienna numeryczna,
- **age**- minimalny wiek graczy, zmienna numeryczna,
- **owned**- ilość zakupionych egzemplarzy, zmienna numeryczna,
- **category**-kategoria gry, zmienna znakowa, często zawiera więcej niż jedną kategorię.

Całą analizę wykonam w programie *RStudio*.

Przygotowanie zbioru

W tej chwili mamy ramkę danych *gry* zawierającą 4999 rekordów. Typy wszystkich zmiennych w kolumnach zostały opisane we wstępnie, nie ma potrzeby zmiany typu żadnej z nich. Mamy za to kilka rekordów które mogą "przeszkadzać".

Zacznę od kolumny *min_players*. W kilku wierszach, wartość w tej kolumnie wynosi 0. Nie ma to sensu aby 0 było liczbą graczy, dlatego zacznę od usunięcia tych wierszy.

```
gry<-gry[gry$min_players>0, ]
```

W rezultacie tej komendy usunięto 14 wierszy z tabeli, mamy ich teraz 4985. Następnie przejdę do kolumny *max_players*. W tej kolumnie również znajdują się wartości 0, ale zamiast usuwać je tak jak powyżej, można pójść o krok dalej, i po prostu usunać wszystkie rekordy, w których maksymalna liczba graczy jest mniejsza od minimalnej.

```
gry<-gry[gry$min_players<=gry$max_players, ]
```

W ten sposób usuniętych zostało kolejne 16 wierszy. Analogicznie należy zająć się kolumnami *avg_time*, *min_time*, *max_time*. Trzeba też pamiętać o tym, że przeciętny czas rozgrywki musi być mniejszy bądź równy od maksymalnego oraz większy bądź równy od minimalnego czasu rozgrywki.

```
gry<-gry[gry$avg_time>0 & gry$min_time>0 & gry$max_time>0, ]
gry<-gry[gry$min_time<=gry$max_time, ]
gry<-gry[gry$avg_time>=gry$min_time & gry$avg_time<=gry$max_time, ]
```

"Odpadły" kolejne wiersze, teraz jest ich 4854.

Przeanalizuję jeszcze kolumnę *year*.

```
range(gry$year)
-3000 2018
```

Jak widać zakres wartości w tej kolumnie zaczyna się od -3000. Oczywiście teoretycznie jest to możliwe, jednak postanowiłem usunąć te rekordy z tabeli. W tej kolumnie znajdują się również wartości 0, jak i wartości mniejsze niż 1900. Mimo, że są one logicznie poprawne, postanowiłem zastosować jeszcze jeden filtr.

```
gry<-gry[gry$year>=1990, ]
```

Liczba obserwacji wynosi teraz 4360, co oznacza, że tabela zawierała około 500 gier powstały przed rokiem 1990. Zdecydowałam się pominąć te wiersze i skupić się na analizie gier, które powstały po roku 1990. Teraz mam już zbiór, zawierający 4360 obserwacji i 13 kolumn.

Chciałabym przyjrzeć się jeszcze kolumnie *category*. W kolumnie tej znajdują się kategorie, do których została zaliczona dana gra, np. *Economic*, *Wargame*, *Card Game*. Poszczególne kategorie są w tej kolumnie wypisane i oddzielone przecinkami. Aby móc wykorzystać kategorie gier do analizy, postanowiłam rozdzielić poszczególne kategorie do osobnych kolumn, dodatkową przeszkodą był fakt, że ilość przepisanych kategorii nie jest taka sama. Wykorzystałam więc funkcję rozdzielenia napisów, i stworzenia nowych kolumn, w której wpisane są kategorie pojedynczo.

```

for (i in 1: nrow(gry)){
x<-strsplit(as.character(gry$category[i]), ", ")
gry[i, "newcategory"]<-length(x[[1]])
for (j in 1:length(x[[1]])){
gry[i,paste("Category", j,sep="_")]<-x[[1]][j]
}
}

gry$Category_1<-as.factor(gsub(" ", "", gry$Category_1))
gry$Category_2<-as.factor(gsub(" ", "", gry$Category_2))
...
gry$Category_1 <- as.factor(ifelse(is.na(gry$Category_1)==T, "0",
as.character(gry$Category_1)))
gry$Category_2 <- as.factor(ifelse(is.na(gry$Category_2)==T, "0",
as.character(gry$Category_2)))
...

```

W zbiorze pojawiło się 12 nowych kolumn. *newcategory* zawiera liczbę przypisanych kategorii. Oprócz tej kolumny mamy jeszcze 11 nowych kolumn *Category_1*, *Category_2*..., w których wpisane są poszczególne kategorie. Kolumn jest 11, ponieważ tyle maksymalnie kategorii miała jedna gra. Dla gier które mają mniej kategorii, od pewnego momentu kolumny zawierają wartość *NA*. Taka postać tej tabeli nie jest jeszcze dobra pod kątem analizy danych, dlatego wykonałam jeszcze jedną pętle.

```

categories<-unique(c(as.character(unique(gry["Category_1"])$Category_1),
as.character(unique(gry["Category_2"])$Category_2), as.character(
unique(gry["Category_3"])$Category_3),as.character(unique(gry["Category_4"])
$Category_4), as.character(unique(gry["Category_5"])$Category_5),
as.character(unique(gry["Category_6"])$Category_6), as.character(unique(
gry["Category_7"])$Category_7), as.character(unique(gry["Category_8"])
$Category_8), as.character(unique(gry["Category_9"])$Category_9),
as.character(unique(gry["Category_10"])$Category_10), as.character(
unique(gry["Category_11"])$Category_11)))

for (i in 1:length(categories)){
for (j in 1: nrow(gry)){
if (gry[j,"Category_1"]==categories[i] | gry[j,"Category_2"]==categories[i] |
gry[j,"Category_3"]==categories[i] |
gry[j,"Category_4"]==categories[i] | gry[j,"Category_5"]==categories[i] |
gry[j,"Category_6"]==categories[i] |
gry[j,"Category_7"]==categories[i] | gry[j,"Category_8"]==categories[i] |
gry[j,"Category_9"]==categories[i] |
gry[j,"Category_10"]==categories[i] | gry[j,"Category_11"]==categories[i])
gry[j,paste(categories[i],",",",")]<-1
else
gry[j,paste(categories[i],",",",")]<-0
}
}

```

Teraz w tabeli jest dodatkowe 85 kolumn, każda z tych kolumn nazywa się jak jedna kategoria i zawiera wartości 0 lub 1 , w zależności od tego czy gra została zakwalifi-

kowana do danej kategorii.

85 kategorii jest dużą liczbą, dlatego postanowiłem stworzyć nowe kategorie, łącząc tematycznie część kategorii. Zrobiłem następującą pętle.

```
for (i in 1:nrow(gry)){
  if (grepl("Economic",gry$category[i]) || grepl("Civilization",
  gry$category[i]) || grepl("Negotiation",gry$category[i])){
    gry$theme[i]<-"Economic"
  }
  else if (grepl("Wargame",gry$category[i])){
    gry$theme[i]<-"War"
  }
  else if (grepl("Ancient",gry$category[i])|| grepl("Mythology",
  gry$category[i])|| grepl("Prehistoric",gry$category[i]) || 
  grepl("Post-Napoleonic",gry$category[i]) || grepl("Renaissance",
  gry$category[i]) || grepl("Medieval",gry$category[i]) || 
  grepl("KoreanWar",gry$category[i])|| grepl(
  "AmericanRevolutionaryWar",gry$category[i]) || grepl("Napoleonic",
  gry$category[i]) || grepl("CivilWar",gry$category[i]) || grepl(
  "WorldWarI", gry$category[i])){
    gry$theme[i]<-"Historical"
  }
  else if (grepl("Aviation/Flight",gry$category[i]) || grepl(
  "Trains",gry$category[i]) || grepl("Transportation",gry$category[i])){
    gry$theme[i]<-"Transport"
  }
  else if (grepl("Math",gry$category[i]) || grepl("Memory",
  gry$category[i]) || grepl("Trivia",gry$category[i]) || grepl("Puzzle",
  gry$category[i])|| grepl("WordGame",gry$category[i]) || grepl(
  "Number",gry$category[i])){
    gry$theme[i]<-"Logical"
  }
  else if (grepl("Spies/SecretAgents",gry$category[i]) || grepl(
  "Mafia",gry$category[i]) || grepl("Murder/Mystery",gry$category[i])){
  {
    gry$theme[i]<-"Crimes"
  }
  else if (grepl("Book",gry$category[i]) || grepl("Novel-based",
  gry$category[i]) || grepl("VideoGameTheme",gry$category[i]) || 
  grepl("ComicBook/Strip",gry$category[i]) || grepl("Movies/TV/
  Radiotheme", gry$category[i])){
    gry$theme[i]<-"BasedOnTheme"
  }
  else if (grepl("Environmental",gry$category[i]) || grepl("Animals",
  gry$category[i]) || grepl("Farming", gry$category[i]) || 
  grepl("Travel", gry$category[i]) || grepl("Medical", gry$category[i))){
    gry$theme[i]<-"Nature"
  }
  else if (grepl("AbstractStrategy",gry$category[i]) || grepl(
  "Industry/Manufacturing",gry$category[i]) || grepl("CityBuilding",
  gry$category[i]) || grepl("Political", gry$category[i))){

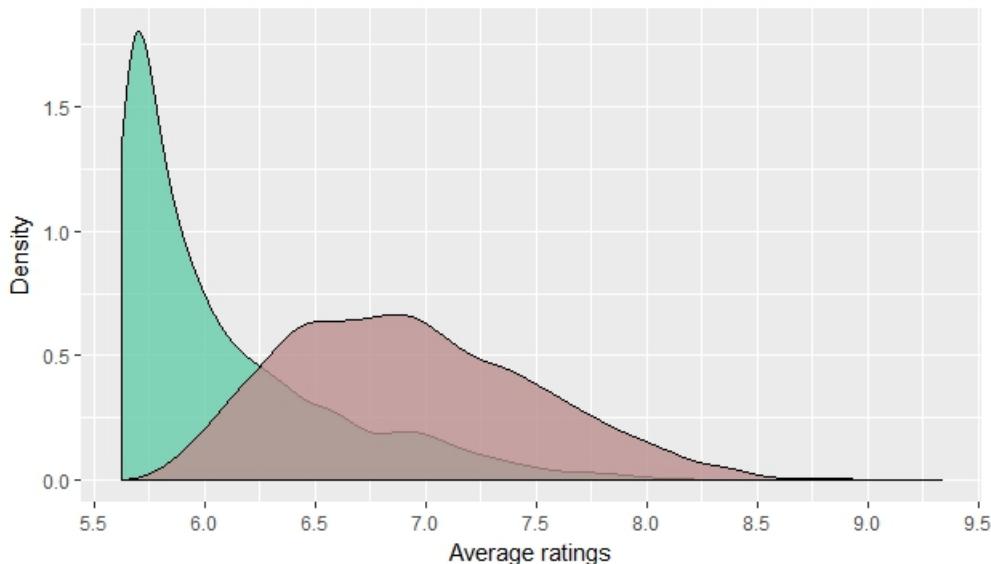
```

```
gry$theme[i]<-"Strategy"
}
else if (grepl("Adventure",gry$category[i]) || grepl("Exploration",
gry$category[i]) || grepl("SpaceExploration",gry$category[i])
|| grepl("Political", gry$category[i])) {
gry$theme[i]<-"Adventure"
}
else {
gry$theme[i]<- "other"
}
}
```

Teraz mam już gotowy zbiór do analizy, zawierający 4360 obserwacji oraz 112 kolumn.

Analiza statystyczna

W tym rozdziale chciałabym dokonać analizy statystycznej zbioru danych, spróbuję przeanalizować m. in. które cechy gier wpływają na oceny graczy i specjalistów. Najpierw porównam oceny specjalistów i graczy za pomocą wykresów gęstości.



```
summary(gry$geek_rating)
Min. 1st Qu. Median Mean 3rd Qu. Max.
5.626 5.711 5.903 6.088 6.312 8.489

summary(gry$avg_rating)
Min. 1st Qu. Median Mean 3rd Qu. Max.
5.783 6.496 6.884 6.937 7.328 9.332
```

Jak widać na wykresie, specjaliści znacznie częściej wystawiali oceny 5-6, im wyższa ocena, tym rzadziej się ona pojawia. Oceny graczy natomiast najczęściej były w przedziale 6-7, oceny niższe i wyższe występowały rzadziej. Jak spojrzymy na statystyki to okazuje się, że średnia tych ocen różni się o prawie 1, tak samo jak mediana. Oznacza to że oceny graczy są widocznie wyższe od ocen specjalistów. Co ciekawe, używając komendy

```
nrow(gry[gry$avg_rating < gry$geek_rating, ])
```

okazuje się, że w żadnym przypadku ocena specjalistów nie jest wyższa od średniej oceny graczy.

Przy okazji wykresów gęstości postanowiłem jeszcze dopasować rozkłady prawdopodobieństwa do ocen. Z wykresów można się już domyślić, że będą to rozkład normalny i wykładniczy, ale wykorzystam do tego pakiet *fitdistrplus* i funkcję *fitdist*.

```

fitdist(gry$avg_rating, "norm")
Fitting of the distribution ' norm ' by maximum likelihood
Parameters:
estimate Std. Error
mean 6.9374170 0.008586993
sd 0.5670017 0.006071836

fitdist(gry$geek_rating, "exp")
Fitting of the distribution ' exp ' by maximum likelihood
Parameters:
estimate Std. Error
rate 0.1642589 0.002487538

```

Co ciekawe, funkcja *fitdist* wymaga listy parametrów początkowych, jednak tutaj w obu przypadkach dopasowanie było na tyle "oczywiste", że program poradził sobie bez tych parametrów.

Po analizie całego zbioru, spróbuję zobaczyć jak statystyki będą zmieniać się przy filtrowaniu tego zbioru.

```

srednia_geek<-c()
srednia_players<-c()
kategoria<-c()
for ( i in 1:85){
srednia_geek[i]<-as.numeric(summary(gry[gry[26+i]==1,]$geek_rating) [4])
srednia_players[i]<-as.numeric(summary(gry[gry[26+i]==1,]$avg_rating) [4])
kategoria[i]<-colnames(gry) [i+26]
sr<-cbind(kategoria, srednia_geek, srednia_players)
}

```

	kategoria	srednia_geek	srednia_players
1	Environmental	6.44209225	7.15814525
2	CardGame	6.04372498017446	6.77507222839017
3	ModernWarfare	6.01490125	7.470269375
4	Civilization	6.35215878205128	7.13098262820513
5	Adventure	6.24202322957198	7.02991988326848
6	Dice	6.09872867924528	6.86753726415094
7	Economic	6.32195696969697	7.1253876969697
8	Ancient	6.14023649805447	6.93562147859922
9	Animals	6.03609638655462	6.68243008403361
10	CityBuilding	6.31061724137931	6.94779099137931
11	Fantasy	6.18297622792937	7.00880799357945
12	AmericanWest	6.12192253731343	6.88647925373134

Showing 1 to 12 of 85 entries

Po wykonaniu tej pętli otrzymuję ramkę danych, w której znajdują się średnie oceny dla poszczególnych kategorii. Okazuje się, że średnie ocen specjalistów wahają się od 5.75 do 6.46, natomiast średnie ocen graczy to zakres 6.48–7.8, czyli najniższa średnia graczy jest wyższa od najwyższej średniej ocen specjalistów.

Predykcja oceny

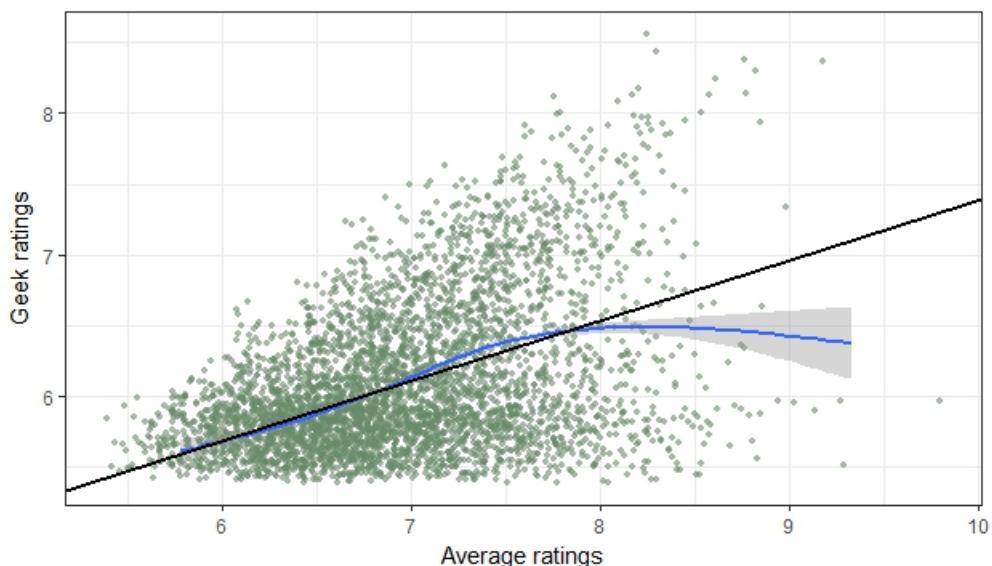
Po krótkiej analizie statystycznej, postanowiłem wykorzystać dostępne funkcje i dokonać predykcji wystawionych ocen. W tabeli znajdują się dwie kolumny z oceną (*ratingiem*) gier. Pierwsza z nich *avg_rating* jest średnią oceną gry wystawioną przez różnych graczy, oprócz niej mamy kolumnę *geek_rating* oznaczającą ocenę specjalistów. Spróbuję przewidzieć ocenę specjalistów na podstawie ocen graczy oraz innych cech gier planszowych. Wykorzystam kilka metod predykcji dostępnych w języku R.

Regresja liniowa

```
model1<-lm(geek_rating ~ avg_rating, data=gry)
coef(model1)

(Intercept) avg_rating
3.1406256 0.4248444
```

Dokonałem regresji liniowej, gdzie ocena specjalistów jest zmienną objaśnianą, natomiast średnia ocena jest zmienną objaśniającą. Widzimy od razu współczynnik kierunkowy i wyraz wolny funkcji liniowej. Jak wygląda to na wykresie?



Na wykresie dodatkowo wykorzystałem funkcję *stat_smooth* która narysowała jak wygląda zależność nieliniowa między zmiennymi.

Aby określić trafność predykcji, skorzystam z funkcji *predict* z pakietu *stats* oraz z funkcji zdefiniowanej w następujący sposób.

```
predlims <- function(preds,sigma) {
  prediction.sd <- sqrt(preds$se.fit^2+sigma^2)
  upper <- preds$fit+2*prediction.sd
  lower <- preds$fit-2*prediction.sd
  lims <- cbind(lower=lower,upper=upper)
  return(lims)
}
```

```
preds.model1 <- predict(mob dell, se.fit=TRUE)
```

Obiekt *preds.model1* jest listą składającą się z 4 elementów.

Name	Type	Value
preds.model1	list [4]	List of length 4
fit	double [4360]	6.82 6.85 6.69 6.67 7.00 6.76 ...
se.fit	double [4360]	0.0210 0.0217 0.0176 0.0170 0.0256 0.0195 ...
df	integer [1]	4358
residual.scale	double [1]	0.4314401

```
predlims.model1 <- predlims(preds.model1, sigma=summary(model1)$sigma)
```

predlims.model1 jest tabelą składającą się z dwóch kolumn, *lower* oraz *upper*, oznaczających dopasowanie z modelem pomniejszone i powiększone o przewidywane odchylenie standardowe.

```
mean(gry$geek_rating <= predlims.model1[, "upper"] & gry$geek_rating >= predlims.model1[, "lower"])
```

Dzięki temu poleceniu obliczam, kiedy rzeczywiste oceny specjalistów mieszczą się w przedziale *[lower, upper]*, a więc kiedy model wskazuje dobry przedział. W tym przypadku, trafność tego dopasowania wynosi 94.44954%.

Do przedwidzenia ocen specjalistów użyłam średniej oceny graczy. Ale chciałabym też spróbować przewidzieć oceny specjalistów na podstawie innych cech gier.

```
for ( i in c(2:8,10:12)) {
model<-lm(geek_rating gry[,i], data=gry)
preds.model <- predict(model, se.fit=TRUE)
predlims.model <- predlims(preds.model, sigma=summary(model)$sigma)
wynik[i]<-mean(gry$geek_rating <= predlims.model[, "upper"] & gry$geek_rating >= predlims.model[, "lower"])
}
wynik<-wynik[c(2:8,10:12)]
```

Wektor *wynik* jest wektorem zawierającym trafność dopasowania dla różnych cech.

	Cechy	Wynik
1	min_players	0.94151376146789
2	max_players	0.94105504587156
3	avg_time	0.941284403669725
4	min_time	0.94151376146789
5	max_time	0.941284403669725
6	year	0.942201834862385
7	avg_rating	0.944495412844037
8	num_votes	0.946788990825688
9	age	0.943348623853211
10	owned	0.946788990825688

Okazuje się że wszystkie wartości są w okolicy 94%. Różnice między poszczególnymi cechami są niewielkie.

Teraz chciałabym spróbować przewidzieć oceny specjalistów biorąc pod uwagę większą ilość cech.

```
kombinacje<-list()
for ( i in 1:10) {
kombinacje[[i]]<-matrix(nrow=choose(10,i), ncol=i)
kombinacje[[i]]<-as.data.frame(combn(c(2,3,4,5,6,7,8,10,11,12),i))
}

wynik<-c()
for ( i in 1:45){
model<-lm(geek_rating gry[,kombinacje[[2]][[i]][1]]+
gry[,kombinacje[[2]][[i]][2]], data=gry)
preds.model <- predict(model, se.fit=TRUE)
predlims.model <- predlims(preds.model, sigma=summary(model)$sigma)
wynik[i]<-mean(gry$geek_rating <= predlims.model[, "upper"]
& gry$geek_rating >= predlims.model[, "lower"])
}

kombinacje[[2]][which(wynik==max(wynik)) ]
colnames(gry) [c(8,12)]
```

Obiekt *kombinacje* jest listą 10 ramek danych. 10 dlatego że biorę pod uwagę 10 cech gier. W powyższym przypadku rozważam pary cech, stąd pętla wykonywana jest 45 razy (dwumian Newtona, ilość dwuelementowych podzbiorów zbioru 10 elementowego). W efekcie dostałam wektor *wynik*, w którym są trafności dopasowań. Okazuje się, że znalazło się dopasowanie które daje większą trafność niż przy braniu pod uwagę jednej cechy. Jest to 0.9603211, model względem średniej oceny graczy *avg_rating* oraz ilości zakupionych egzemplarzy *owned*.

Teraz chciałabym wykonać analogiczne rozumowanie dla 4 obserwacji jednocześnie.

```
wynik<-c()
for ( i in 1:210){
model<-lm(geek_rating gry[,kombinacje[[4]][[i]][1]]+
gry[,kombinacje[[4]][[i]][2]]+gry[,kombinacje[[4]][[i]][3]]+
gry[,kombinacje[[4]][[i]][4]], data=gry)
preds.model <- predict(model, se.fit=TRUE)
predlims.model <- predlims(preds.model, sigma=summary(model)$sigma)
wynik[i]<-mean(gry$geek_rating <= predlims.model[, "upper"]
& gry$geek_rating >= predlims.model[, "lower"])
}

kombinacje[[4]][which(wynik==max(wynik)) ]
colnames(gry) [c(2,8,10,11)]
```

W rezultacie mamy maksymalny wynik dopasowania 0.9612385, dla zestawu: *min_players*, *avg_rating*, *num_votes*, *age*. Po raz kolejny jedną z cech jest średnia ocena graczy, ale dodając 2 cechy, trafność wzrosła nam tylko o około 0,001, co nie jest dużym poprawieniem.

Teraz chciałabym jeszcze sprawdzić jak wygląda trafność dopasowania gdy weźmiemy wszystkie 10 cech. W tym przypadku nie będzie potrzebna pętla.

```

model<-lm(geek_rating gry[,kombinacje[[1]][[1]][1]]+...
...+gry[,kombinacje[[1]][[1]][10]], data=gry)
preds.model <- predict(model, se.fit=TRUE)
predlims.model <- predlims(preds.model, sigma=summary(model)$sigma)
wynik[i]<-mean(gry$geek_rating <= predlims.model[, "upper"]
& gry$geek_rating >= predlims.model[, "lower"])

```

Okazało się, że w przypadku wzięcia 10 cech, trafność wcale nie zwiększyła się. Wyszło 0.9594037, czyli więcej niż przy wzięciu pod uwagę jednej cechy, ale mniej niż np. dla dwóch i czterech.

Algorytm knn

Oprócz modelu liniowego postanowiłem spróbować dokonać predykcji za pomocą algorytmu k najbliższych sąsiadów. Postanowiłem wykorzystać algorytm aby przewidzieć z których lat pochodzi gra na podstawie przyznanych jej ocen, zarówno przez specjalistów jak i wszystkich graczy.

Zaczęłam od dodania do tabeli kolumny *Years* która zawiera 3 "factory": *1990s, 2000s, 2010s*.

```

gry$Years<-as.factor(ifelse( gry$year<=1990, '1990s', ifelse(
gry$year<=2000, '2000s', ifelse( gry$year<=2010,NA))))

```

Następnie przeszłam do utworzenia zbiorów uczących i testowych. Zbiór *gry_reduced* to zbiór w którym zostawiłam tylko 3 kolumny, *avg_rating*, *geek_rating*, *Years*. Następnie podzieliłam go na 3 zbiory, według kolumny *Years* i w sposób losowy wybrałam 100 obserwacji z każdego zbioru. Następnie podzieliłam te zbiory na pół, jedna z nich trafiła do zbioru treningowego, druga do zbioru testowego.

```

gry_reduced_90<-gry_reduced[gry_reduced$Years=="1990s",]
gry_reduced_90<-gry_reduced_90[sample(1:nrow(gry_reduced_90),100),]
gry_reduced_00<-gry_reduced[gry_reduced$Years=="2000s",]
gry_reduced_00<-gry_reduced_00[sample(1:nrow(gry_reduced_00),100),]
gry_reduced_10<-gry_reduced[gry_reduced$Years=="2010s",]
gry_reduced_10<-gry_reduced_10[sample(1:nrow(gry_reduced_10),100),]
train <- as.matrix(rbind(gry_reduced_90[1:50,1:2],
gry_reduced_00[1:50,1:2], gry_reduced_10[1:50,1:2]))
test <- as.matrix(rbind(gry_reduced_90[51:100,1:2],
gry_reduced_00[51:100,1:2], gry_reduced_10[51:100,1:2]))

```

Następnie tworzę klasyfikację zmiennych, jest to po prostu wektor z powtórzonymi po 50 razy *1990s, 2000s, 2010s*. I dla zbioru testowego i treningowego.

```

train_cl<-factor(rep(c("1990s", "2000s", "2010s"), each=50))
test_cl<-factor(rep(c("1990s", "2000s", "2010s"), each=50))

```

Teraz mogę już przejść do uruchomienia algorytmu. Pakiet *class* zawiera funkcję *knn*, której argumentami są zbiór treningowy, zbiór testowy, klasyfikacja oraz ilość sąsiadów.

Spróbuje najpierw uruchomić algorytm dla $k = 10$.

```
knn<-knn(train,test,train_cl, k=10)
table(knn,test_cl)
```

Tabela wywołana w linijce drugiej, pokażę nam ile z predykcji było poprawnych. Wygląda ona następująco.

knn	test_cl		
	1990s	2000s	2010s
1990s	13	15	18
2000s	23	21	11
2010s	14	14	21

Jak czytać taką tabelę? Wartości na przekątnej oznaczają poprawne dopasowania. Co oznacza że jedynie $13+21+21$ wartości zostało przewidzianych poprawnie. Daje to poprawność algorytmu dla $k=10$ zaledwie 36.7%. Dlatego spróbuje teraz wykonać algorytm dla innej liczby sąsiadów.

```
p<-c()
for ( k in 1:49) {
  knn<-knn(train,test,train_cl, k)
  p[k]<-mean(knn==test_cl)
}
range(p)
which(p==max(p))
```

Okazuje się, że trafność dopasowań mieści się w przedziale [0.34, 0.4733], wartość największa osiągana jest gdy k równa się 18, 19 lub 25. Jednak nawet największa trafność jest mniejsza niż 50%. Oznacza to, że predykcja roku powstania na podstawie przeciętnej oceny oraz oceny specjalistów nie jest zbyt dobra.

Wnioski

Celem mojego projektu, była analiza ocen gier planszowych. W pierwszej części dokonałam krótkiej analizy statystycznej, aby potem przejść do predykcji. Z analizy statystycznej dowiedziałam się, że oceny specjalistów są widocznie niższe od przeciętnych ocen. Podczas próby predykcji z użyciem modelu liniowego, okazało się, że trafność przewidywanych ocen specjalistów na podstawie ocen przeciętnych jest na poziomie ponad 94%, co jest całkiem dobrym wynikiem. W mojej pracy starałam się znaleźć jak najlepsze dopasowanie, z użyciem kilku cech. Udało się podwyższyć trafność do 96%, z użyciem dwóch cech. Spróbowałam również dokonać predykcji korzystając z algorytmu *KNN*, próbowałam przewidzieć dekadę powstania grt na podstawie dwóch wartości - ocen specjalistów i graczy, jednak ta predykcja okazała się być mało trafna.