Warsztaty badawcze 2020 - pd 1

Jakub Kosterna

3/3/2020

1. Wybór zbioru danych

Za cel pierwszej pracy domowej wziąłem sobie analizę zbioru **Speed Dating** (i tak samo moja decyzja była bardzo szybka - no hej, przede mną wspaniała podróż po świecie relacji damsko-męskich, czy to nie fascynujące?). Jak sama nazwa wskazuje dotyczy on danych zebranych na podstawie tak zwanych "szybkich randek". Informacje zostały zebrane podczas spotkań w latach 2002-2004 i oparte były na 4-minutowych "pierwszych randkach" z płcią przeciwną. Uczestnicy po każdej z nich byli pytani o zainteresowanie zobaczeniem potencjalnej drugiej połówki ponownie, a także mieli za zadanie ocenić ją pod kątem sześciu kryteriów:

1. Atrakcyjność
2. Szczerość
3. Inteligencja
4. Zabawa
5. Ambicja
6. Wspólne zainteresowania.

Zbiór znalazłem pod Id 146607 - co ciekawe o największym numerze spośród 100 zaproponowanych.

```
# install.packages("OpenML") # if not installed
library(OpenML)
# install.packages("dplyr") # if not installed
library(dplyr)
# install.packages("DataExplorer") # if not installed
library(DataExplorer)
task.ids <- getOMLStudy('OpenML100')$tasks$task.id
task <- getOMLTask(146607)
data <- as.data.frame(task)
```

W kolejnych krokach skorzystamy z pakietu **DataExplorer** - jest to przydatne narzędzie udostępniające wiele ciekawych funkcji do oglądu ramki.

2. Wstępna analiza i obróbka

Przyjrzyjmy się naszemu zbiorowi lepiej. Czym są i ile jest kolumn i wierszy?

```
ncol(data)
```

```
## [1] 123
```

```
nrow(data)
```

```
## [1] 8378
```

No nieźle, aż 123 wiersze! Weźmy tylko te najciekawsze.

Dokładne informacje o kolumnach znalazłem na: https://www.openml.org/d/40536

Pozostawimy sobie:

- gender: Gender of self
- age: Age of self
- age_o: Age of partner
- d_age: Difference in age
- attractive_o: Rating by partner (about me) at night of event on attractiveness
- attractive: Rate yourself - attractiveness
- attractive_partner: Rate your partner - attractiveness
- intelligence_o: Rating by partner (about me) at night of event on intelligence
- intelligence: Rate yourself - intelligence
- intelligence_partner: Rate your partner - intelligence
- decision: Decision at night of event.
- decision_o: Decision of partner at night of event.
- match: Match (yes/no)

Obróbmy dane i zobaczmy wynik wybierając dwadzieścia losowych wierszy.

```
data <- data %>% select(gender, age, age_o, d_age, attractive_o, attractive, attractive_partner,
                  intelligence_o, intelligence, intelligence_partner, decision, decision_o, match)
colnames(data) <- c("gender", "age", "age_o", "d_age", "attr_o", "attr", "attr_p",
                    "intel_o", "intel", "intel_p", "decision", "decision_o", "match")
set.seed(124)
knitr::kable(sample_n(data, 20))
```

gender	age	age_o	d_age	attr_o	attr	attr_p	intel_o	intel	intel_p	decision	decision_o	match
male	23	27	4	4	5	6	7	8	7	0	0	0
male	24	NA	24	NA	7	NA	NA	8	NA	0	0	0
female	22	27	5	8	7	6	8	9	9	0	1	0
male	27	NA	27	9	7	9	9	8	8	1	1	1
female	24	26	2	5	5	7	7	7	7	0	0	0
male	28	34	6	6	10	4	8	10	8	0	1	0
female	30	24	6	7	9	4	7	10	9	0	1	0
male	30	22	8	4	5	8	6	5	8	0	0	0
female	30	26	4	5	8	7	6	7	7	0	0	0
female	33	22	11	4	8	5	6	9	6	0	0	0
female	NA	27	27	10	NA	7	8	NA	9	1	1	1
female	28	23	5	6	8	4	6	8	7	0	1	0
male	23	23	0	5	7	6	7	10	7	1	0	0
male	29	27	2	7	7	8	8	7	7	1	0	0
female	32	23	9	4	4	7	7	6	7	0	0	0
female	25	30	5	NA	8	5	NA	10	9	1	0	0
female	25	29	4	10	8	7	10	8	9	1	1	1
female	23	26	3	6	3	5	6	3	9	1	0	0
female	27	24	3	7	6	4	8	7	8	1	1	1
female	27	27	0	8	5	3	9	7	7	0	1	0

O kurczę pieczone! Wygląda na to, że społeczność *speed dating* przynajmniej w tej grupie w latach 2002-2004 jest średnio zgodna. Na ową wylosowaną dwudziestkę piątkę tylko 4 matche i 9 nieodwzajemnionych polubień.

Przy okazji mamy styczność z pewnym brakiem danych, który przy surowym zbiorze może spowodować konsternację - nie wszyscy uczestnicy zabawy podali swój wiek, a jak pokazuje chociażby pierwszy wiersz, różnica wieku jest wtedy wyliczana jako informacja o wieku osoby już go posiadającego. Jak jest w przypadku dwóch niewiadomych wartości w tym temacie - nie mam pojęcia. Ale zmodyfikujmy data frame tak, żeby dla przynajmniej jednego niewiadomego wieku także i kolumnie *d_age* przypisywał NA.

```
data$d_age[is.na(data$age) | is.na(data$age_o)] <- NA
```

Zobaczmy efekt:

```
set.seed(124)
knitr::kable(sample_n(data, 10))
```

gender	age	age_o	d_age	attr_o	attr	attr_p	intel_o	intel	intel_p	decision	decision_o	match
male	23	27	4	4	5	6	7	8	7	0	0	0
male	24	NA	NA	NA	7	NA	NA	8	NA	0	0	0
female	22	27	5	8	7	6	8	9	9	0	1	0
male	27	NA	NA	9	7	9	9	8	8	1	1	1
female	24	26	2	5	5	7	7	7	7	0	0	0
male	28	34	6	6	10	4	8	10	8	0	1	0
female	30	24	6	7	9	4	7	10	9	0	1	0
male	30	22	8	4	5	8	6	5	8	0	0	0
female	30	26	4	5	8	7	6	7	7	0	0	0
female	33	22	11	4	8	5	6	9	6	0	0	0

No i elegancko, szafa gra!

3. Braki danych i ogólne wnioski

Zweryfikujmy zbiór pod kątem braków danych.

```
# install.packages("naniar") # if not installed
library(naniar)
```

```
## Warning: package 'naniar' was built under R version 3.6.3
```

```
# install.packages("visdat") # if not installed
library(visdat)
```

```
## Warning: package 'visdat' was built under R version 3.6.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(dplyr)
# install.packages("mice") # if not installed
library(mice)
```

```
## Warning: package 'mice' was built under R version 3.6.3
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

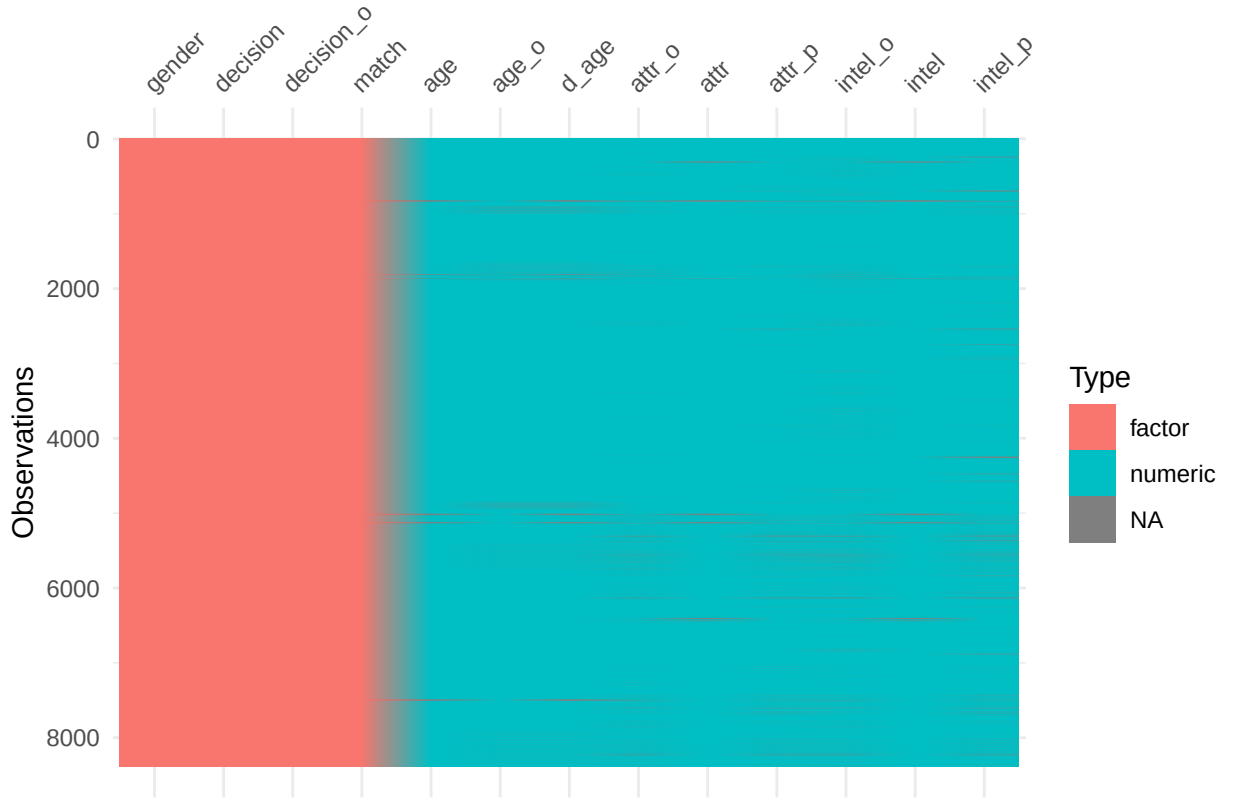
```
knitr::kable(summary(data[1:7]))
```

gender	age	age_o	d_age	attr_o	attr	attr_p
female:4184	Min. :18.00	Min. :18.00	Min. : 0.000	Min. : 0.00	Min. : 2.000	Min. : 0.00
male :4194	1st Qu.:24.00	1st Qu.:24.00	1st Qu.: 1.000	1st Qu.: 5.00	1st Qu.: 6.000	1st Qu.: 5.00
NA	Median :26.00	Median :26.00	Median : 3.000	Median : 6.00	Median : 7.000	Median : 6.00
NA	Mean :26.36	Mean :26.36	Mean : 3.658	Mean : 6.19	Mean : 7.085	Mean : 6.19
NA	3rd Qu.:28.00	3rd Qu.:28.00	3rd Qu.: 5.000	3rd Qu.: 8.00	3rd Qu.: 8.000	3rd Qu.: 8.00
NA	Max. :55.00	Max. :55.00	Max. :32.000	Max. :10.50	Max. :10.000	Max. :10.00
NA	NA's :95	NA's :104	NA's :198	NA's :212	NA's :105	NA's :202

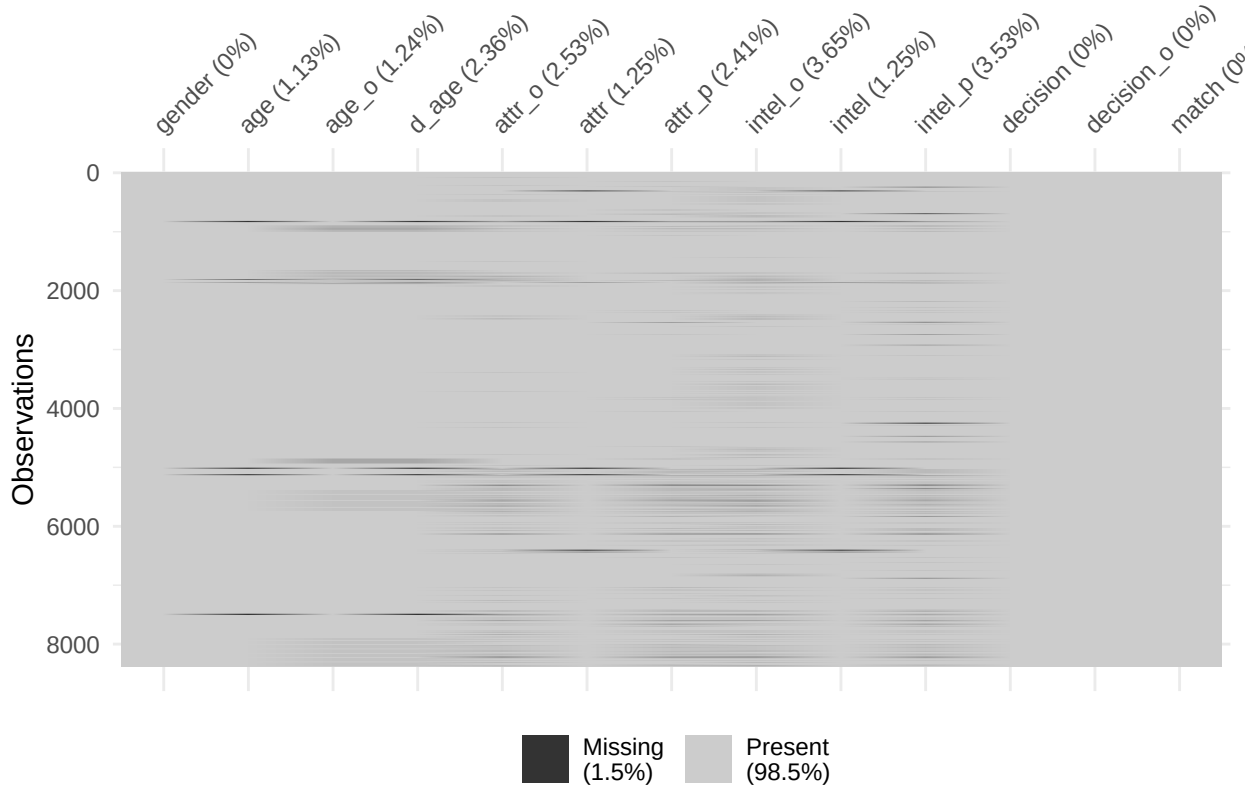
```
knitr::kable(summary(data[8:13]))
```

intel_o	intel	intel_p	decision	decision_o	match
Min. : 0.000	Min. : 2.000	Min. : 0.000	0:4860	0:4863	0:6998
1st Qu.: 6.000	1st Qu.: 7.000	1st Qu.: 6.000	1:3518	1:3515	1:1380
Median : 7.000	Median : 8.000	Median : 7.000	NA	NA	NA
Mean : 7.369	Mean : 7.704	Mean : 7.369	NA	NA	NA
3rd Qu.: 8.000	3rd Qu.: 9.000	3rd Qu.: 8.000	NA	NA	NA
Max. :10.000	Max. :10.000	Max. :10.000	NA	NA	NA
NA's :306	NA's :105	NA's :296	NA	NA	NA

```
vis_dat(data)
```




```
vis_miss(data)
```



Jak widzimy braki danych występują w kolumnach dotyczących wieku i oceny atrakcyjności oraz inteligencji (swojej lub partnera). Są jednak kompletne informacje w temacie płci, a także decyzji co do chęci na następne spotkanie.

Najciekawsze wnioski?

1. Dane dotyczą przybliżonej liczby mężczyzn i kobiet. Są tu głównie 20-kilkulatkowie
2. Najtrudniej uczestnikom ocenić było inteligencję szybkiej-partnerki (szybkiego-partnera), najłatwiej zaś swoją atrakcyjność (patrząc na liczbę wartości NA)
3. Przeciętna różnica wieku randkowiczów to około 3-4 lata
4. Uczestnicy raczej dowartościowani - średnio ocenili swoją atrakcyjność o jeden punkt wyżej niż atrakcyjność drugiego uczestnika randki, a swoją inteligencję o niecałe 0,5 punkta lepiej

Zweryfikujmy jeszcze liczbę nieodwzajemnionych "polubień" i wszystkich matchy.

```
likes <- as.integer(data$decision) + as.integer(data$decision_o) - 2
knitr::kable(prop.table(table(likes)))
```

likes	Freq
0	0.3252566
1	0.5100263
2	0.1647171

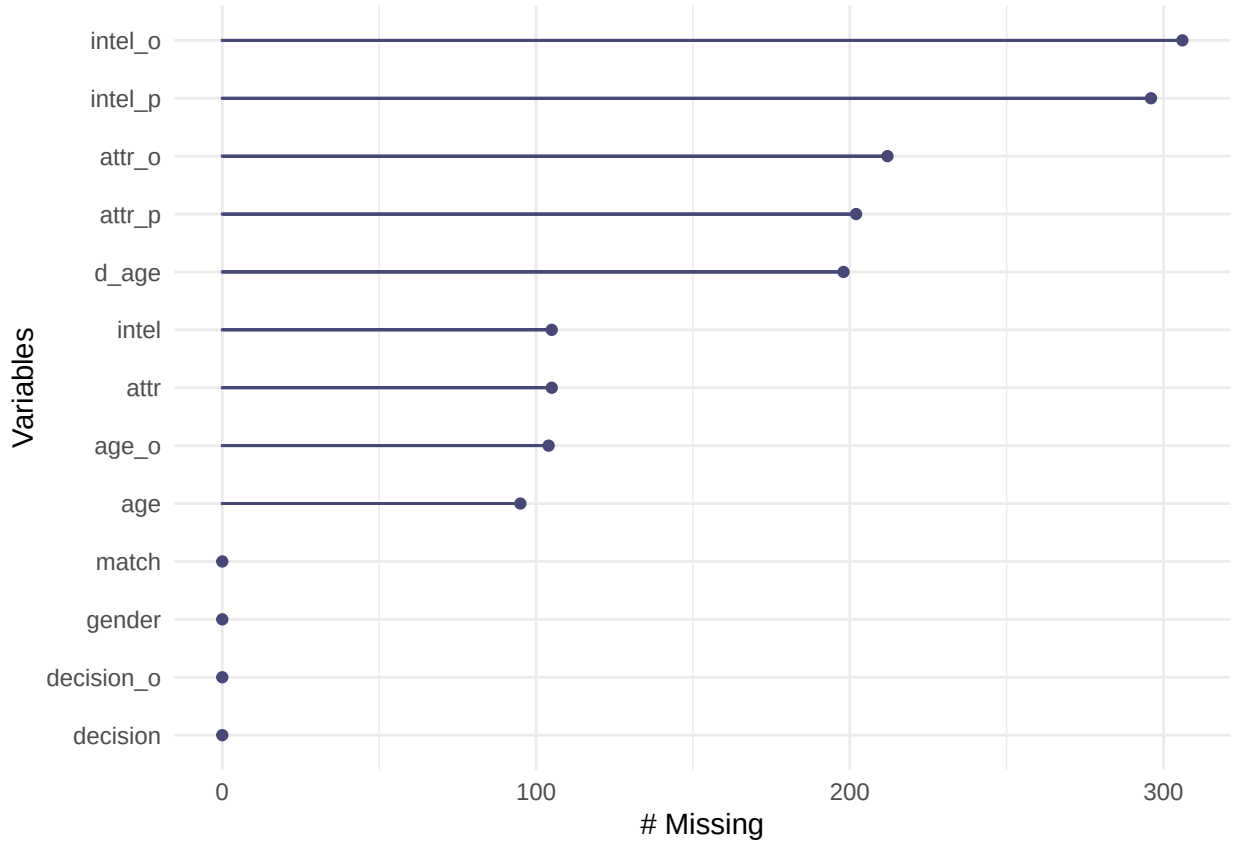
No i mamy jeszcze jeden wniosek...

5. Tylko co szósta para jednogłośnie ogłosiła chęć kolejnego spotkania. Co ciekawe **aż połowa werdyktów to nieodwzajemnione polubienia**, a tylko około 1/3 randkowiczów zgodnie stwierdziła, że nie ma co dalej marnować czasu.

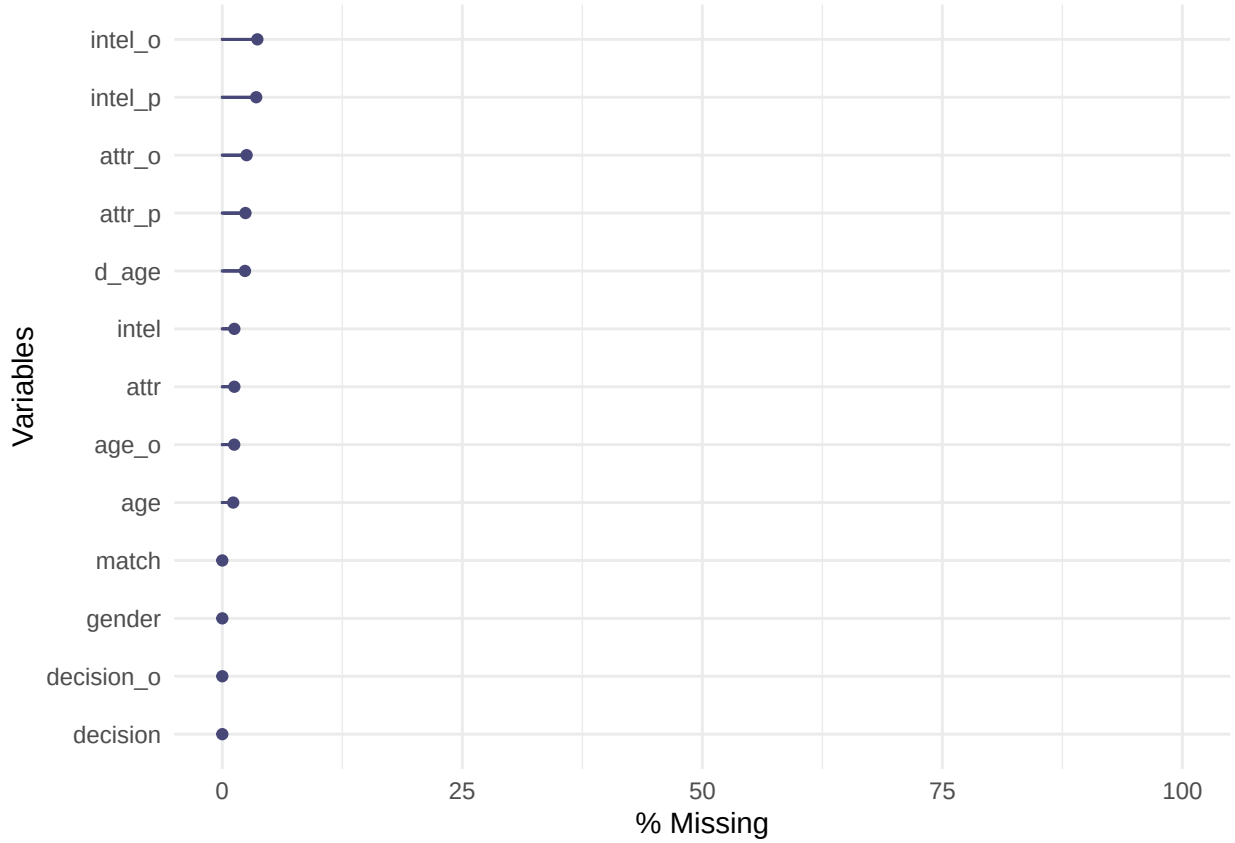
Co ciekawe okazuje się tu także [choć bez zaskoczeń], że wnioskowanie z ledwo 20-wierszowej losowo wygenerowanej podramki może być bardzo mylące - tak można by pomyśleć, że dalej umawia się nie co 6., ale co 20. para, zaś polubień jednostronnych jest nie połowa, a aż 3 na 4. Nie dajmy się zwieść!

Wiemy już na czym stoimy w ogólnym stopniu. Skorzystajmy z narzędzi nauczonych na laboratoriach 2 w celu zdobycia jeszcze większej ilości przydatnych informacji o brakach danych w naszym Speed-datingowym zbiorze.

```
gg_miss_var(data)
```



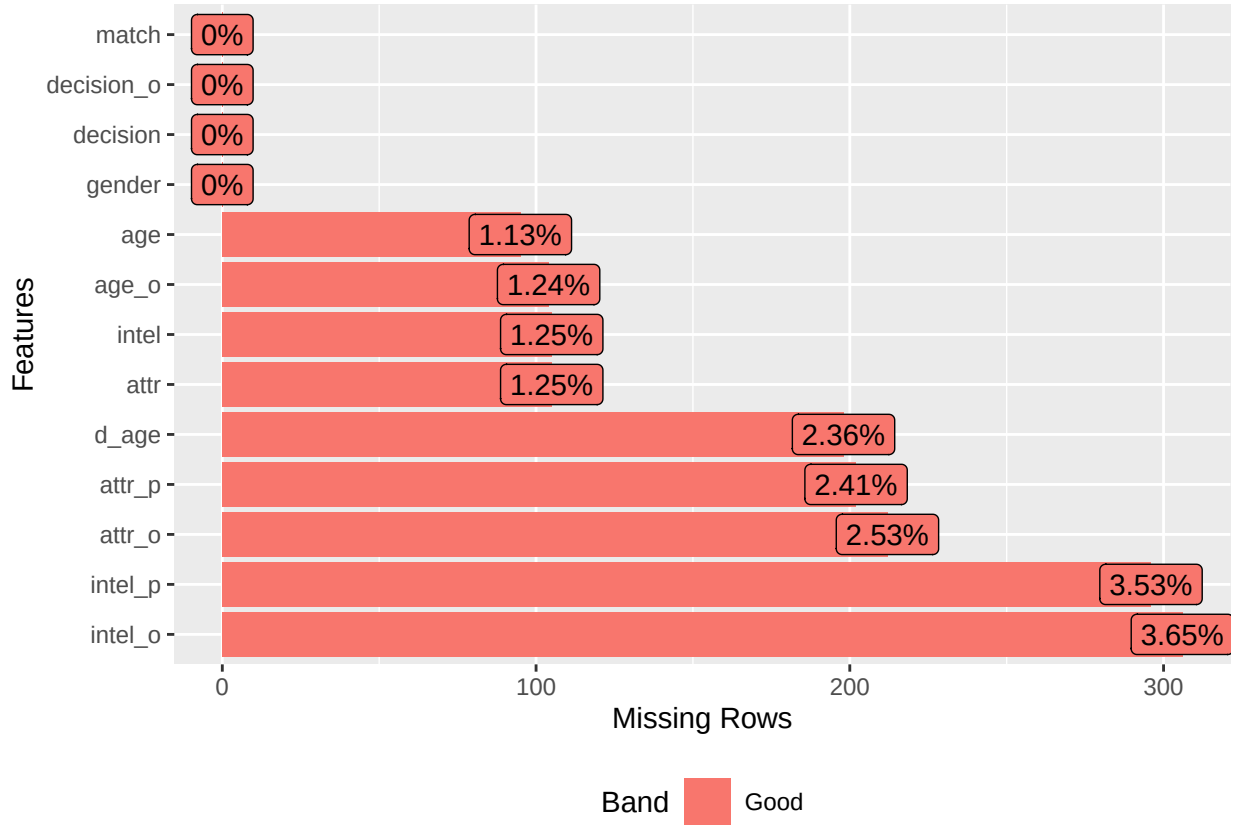

```
gg_miss_var(data,
            show_pct = TRUE) +
  ylim(0, 100)
```



Tak oto otrzymaliśmy ładną wizualizację ilości braków danych dla kolejnych kolumn. Jak widać w każdym wypadku jest to maksymalnie kilka procent - możemy więc stąd wstępnie pomyśleć, że olanie wierszy je zawierających lub ich modyfikacja nie powinna wpłynąć bardzo na istotę zbioru.

Ładną prezentację możemy także otrzymać dzięki pakietowi **DataExplorer**.

```
plot_missing(data)
```



Znając konkretne liczby możemy się spodziewać, że dla tak rzadkich braków nie powinniśmy się otrzymać dużych rożnic w analizie zależności od tego, co z nimi zrobimy.

Jak widać mimo rzadkich braków, jedynie w pełni pozostają kolumny dotyczące płci, decyzji i informacji o sparowaniu. Takie dane to w gruncie rzeczy nic ciekawego, dlatego w kolejnych krokach nie będę sprawdzał efektów dla usunięcia kolumn zawierających jakiekolwiek brakujące informacje.

4. Wizualizacje i wnioski suchych danych z NA (usunięte wiersze i kolumny)

Usuńmy brakujące wartości.

```
data_no_na <- na.omit(data)
print(paste("Usunietych wierszy:", nrow(data) - nrow(data_no_na)))
```

```
## [1] "Usunietych wierszy: 681"
```

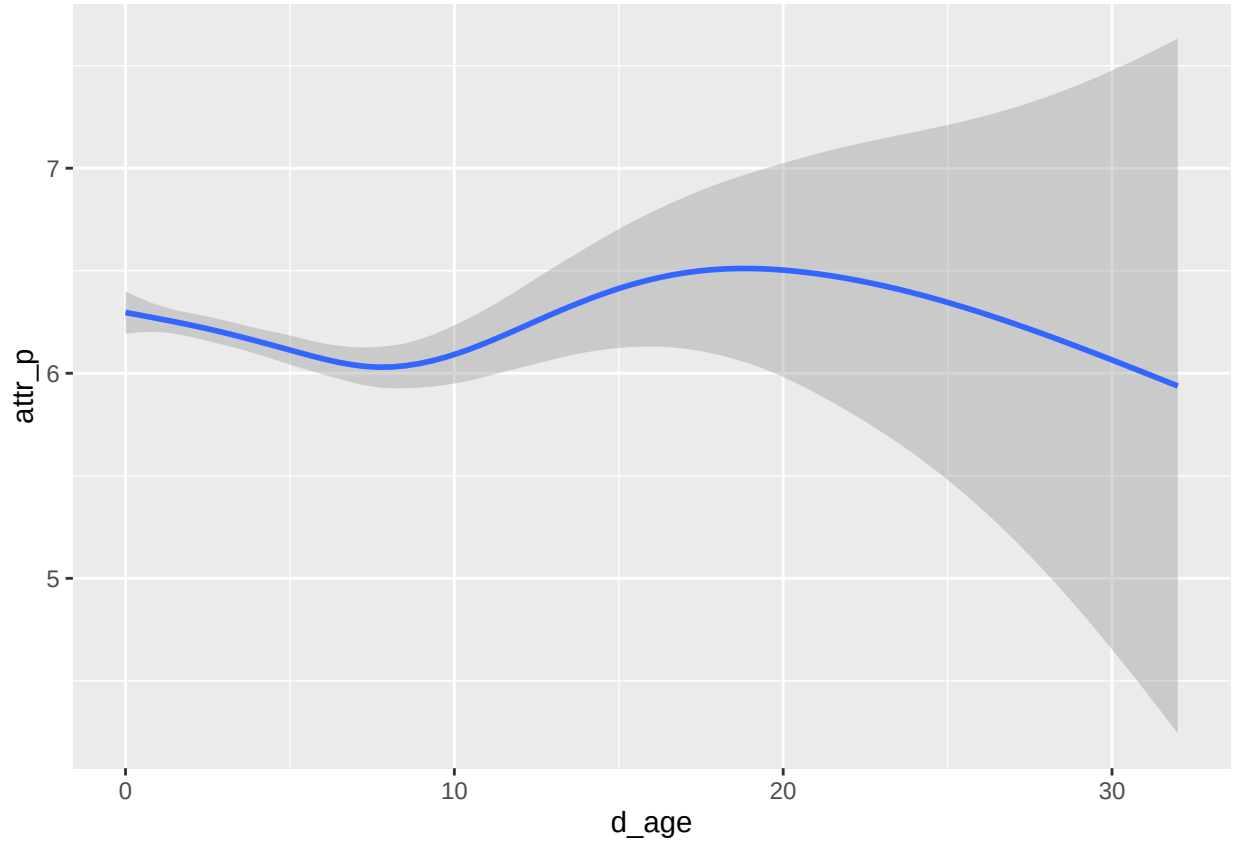
```
print(paste("Procent usunitych wierszy:", round((nrow(data) - nrow(data_no_na)) / nrow(data) * 100, 2),
```

```
## [1] "Procent usunitych wierszy: 8.13 %"
```

4.1. Ocena potencjalnej drugiej połówki a różnica wieku

```
ggplot(data_no_na,
       aes(x = d_age,
           y = attr_p)) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



O proszę! Jak można się spodziewać, zbliżony wiek sprzyja lepszej ocenie. Ciekawie robi się od około 8 lat w góre, gdzie do różnicy wieku ~18 lat ocena atrakcyjności wzrasta, a potem maleje. Jeżeli przybliżenie poprzez *geom_smooth* na niezbyt dużym zbiorze danych można brać na poważnie.

4.2. Inteligencja i atrakcyjność partnera a decyzja

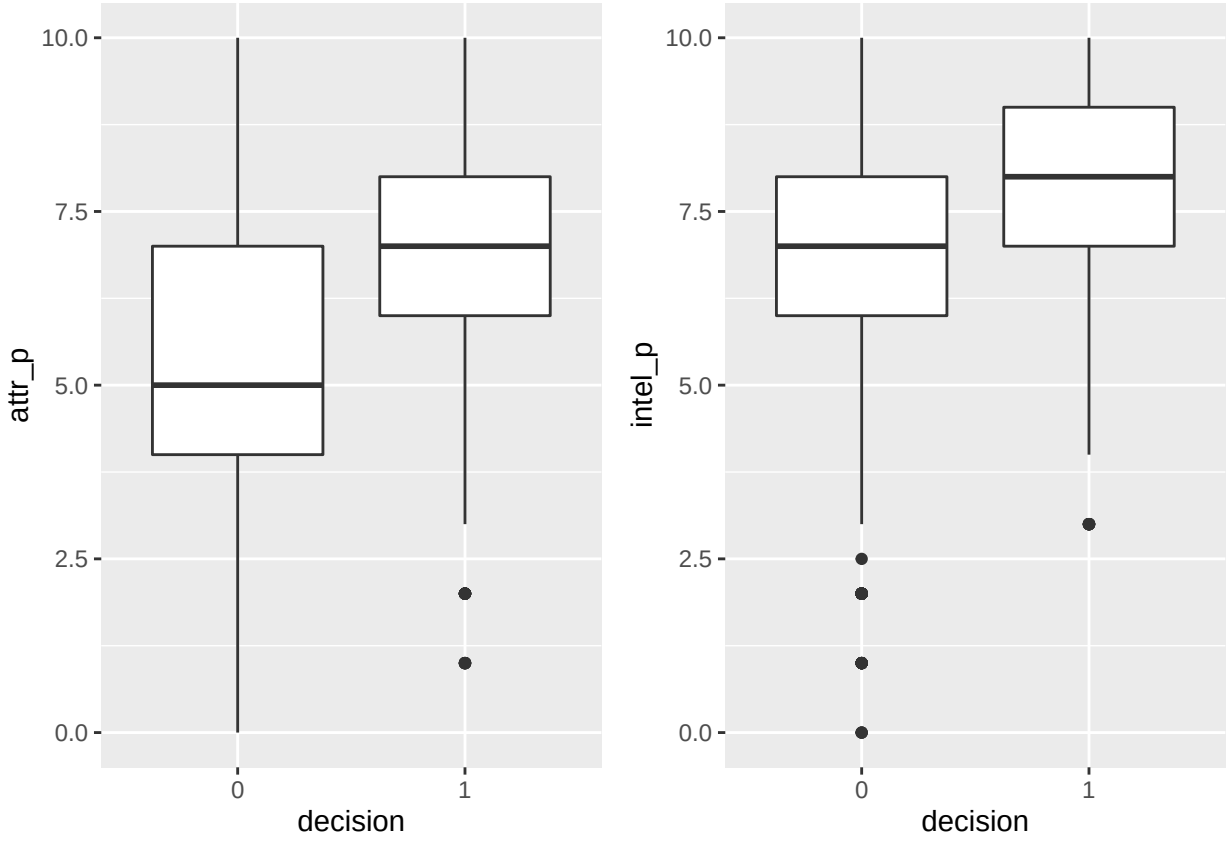
```
g1 <- ggplot(data_no_na, aes(x = decision, y = attr_p)) +
  geom_boxplot()
g2 <- ggplot(data_no_na, aes(x = decision, y = intel_p)) +
  geom_boxplot()
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.6.3
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(g1, g2, ncol=2)
```

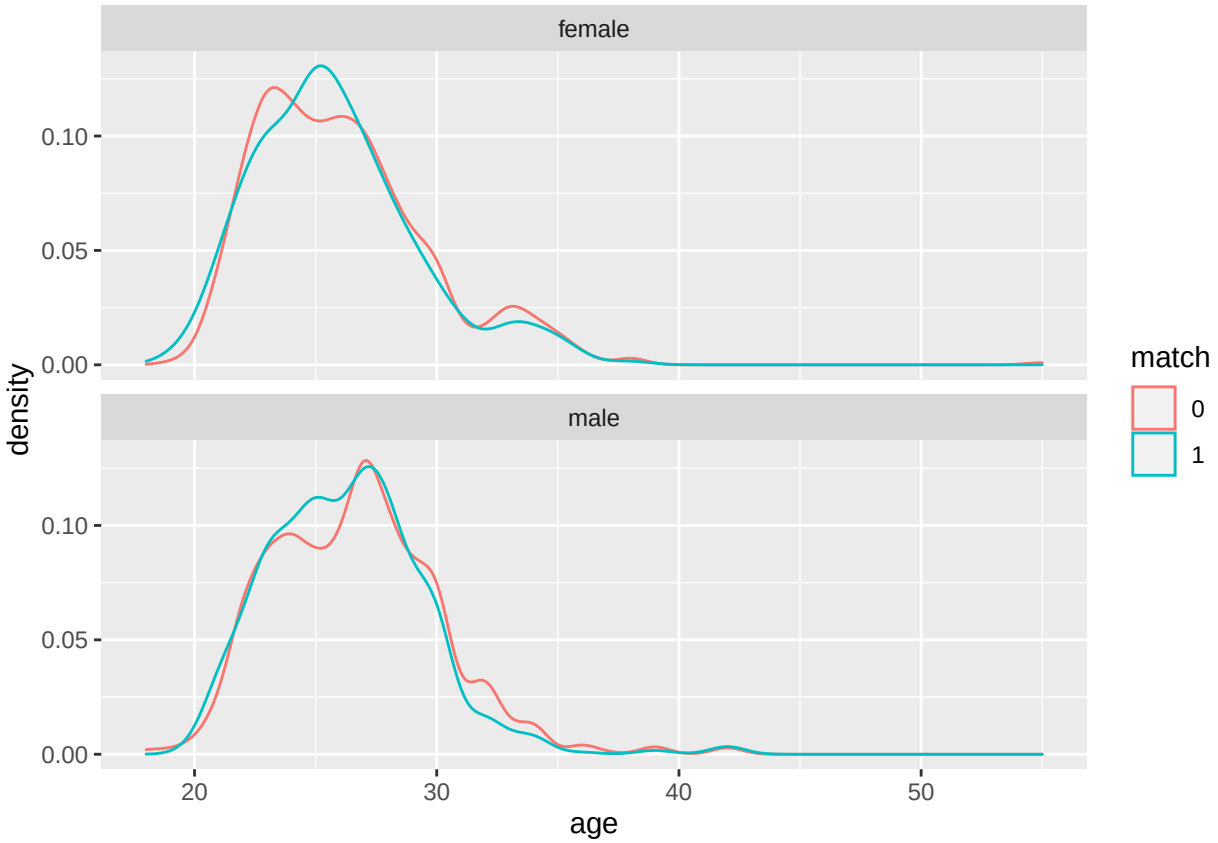


Mediana ocen speed-partnerów uznanych za godnych następnego spotkania jest niemalże równa trzeciemu kwantylowi niewybranych. Dotyczy to zarówno inteligencji jak i atrakcyjności. Co ciekawe, uczestnicy zdecydowanie lepiej ocenili inteligencję niż atrakcyjność swoich randkowiczów - wykres skrzynkowy inteligencji niewybranych partnerów jest niemalże identyczny do boxplota wybranych biorąc pod uwagę atrakcyjność.

Okazuje się jednak, że o wiele bardziej istotna jest atrakcyjność niż inteligencja w kwestii chęci przyjścia na następną randkę - różnica mediany dla atrakcyjności ogólnej wyniosła około dwa punkty, zaś dla inteligencji - niecały jeden.

4.3. Różnica wieku i płeć a matche

```
ggplot(data_no_na, aes(x = age, color = match)) +
  geom_density() +
  facet_wrap(~gender, ncol = 1)
```



Zarówno kobiety jak i mężczyźni najczęściej na match mogą liczyć w wieku lat 25. Najczęściej odpychane są 23-latki i 33-latki, a także 32-33-latkowie. Czemu 30-paro-latkowie mają takie problemy z dogadaniem się? Tego nie wiem, acz intryguje.

5. Wizualizacje i wnioski pustych danych zastąpionych średnimi

Zastąpmy średnimi.

```
imp <- mice(data, method = "mean", m = 1, maxit = 1)
```

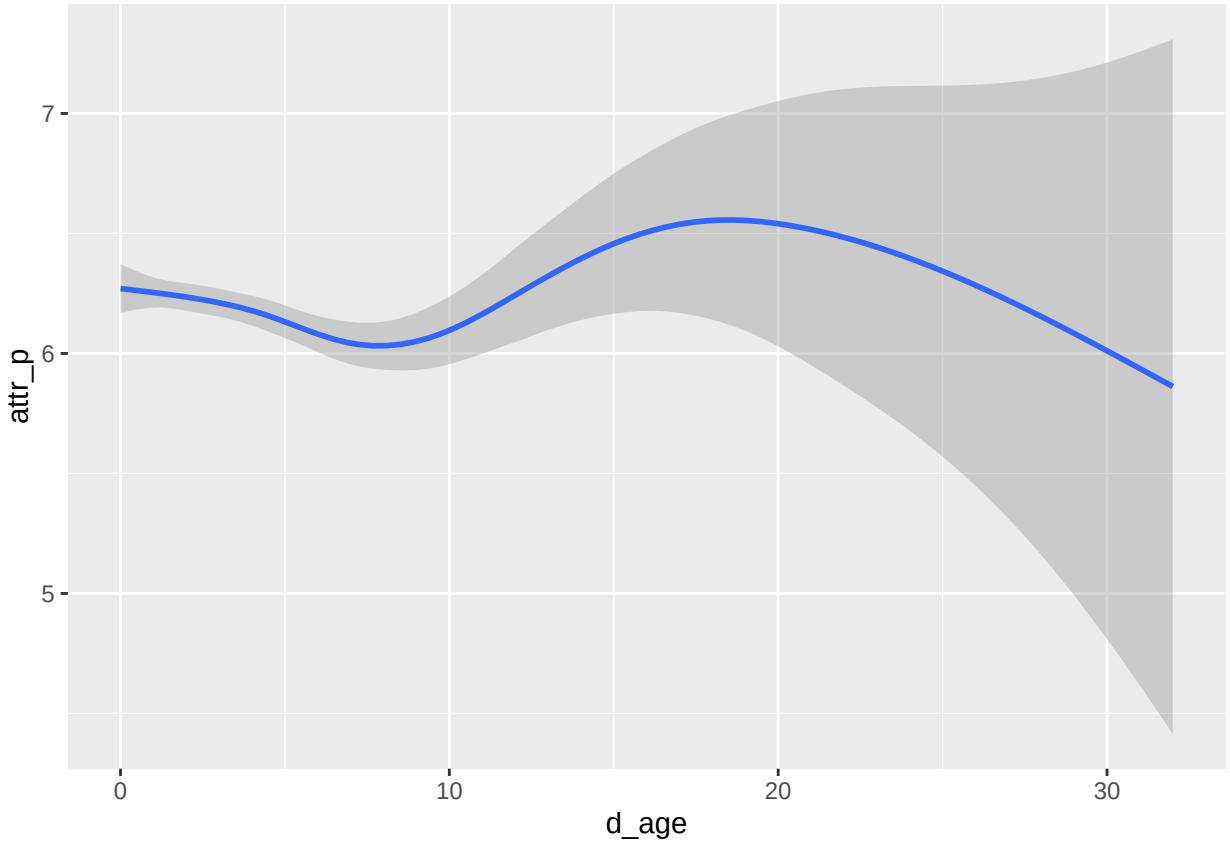
```
##
##  iter imp variable
##   1   1  age  age_o  d_age  attr_o  attr  attr_p  intel_o  intel  intel_p
```

Wypełnijmy naszą nową ramkę danych.

```
data_mean <- complete(imp)
```

5.1. Ocena potencjalnej drugiej połówki a różnica wieku

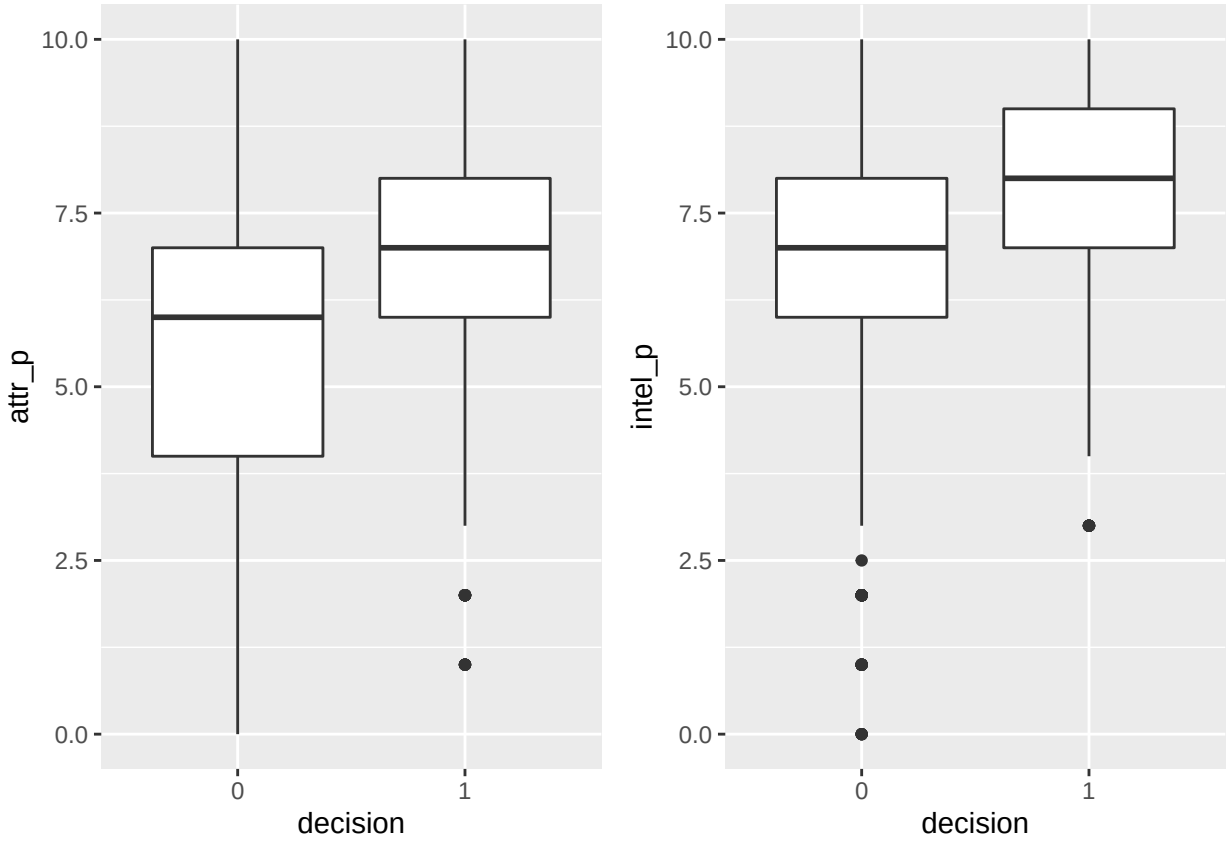
```
ggplot(data_mean,
       aes(x = d_age,
           y = attr_p)) +
  geom_smooth()
```



Bez widocznych zmian - zastąpienie średnimi nie wpłynęło na efekt.

5.2. Inteligencja i atrakcyjność partnera a decyzja

```
g1 <- ggplot(data_mean, aes(x = decision, y = attr_p)) +
  geom_boxplot()
g2 <- ggplot(data_mean, aes(x = decision, y = intel_p)) +
  geom_boxplot()
grid.arrange(g1, g2, ncol=2)
```

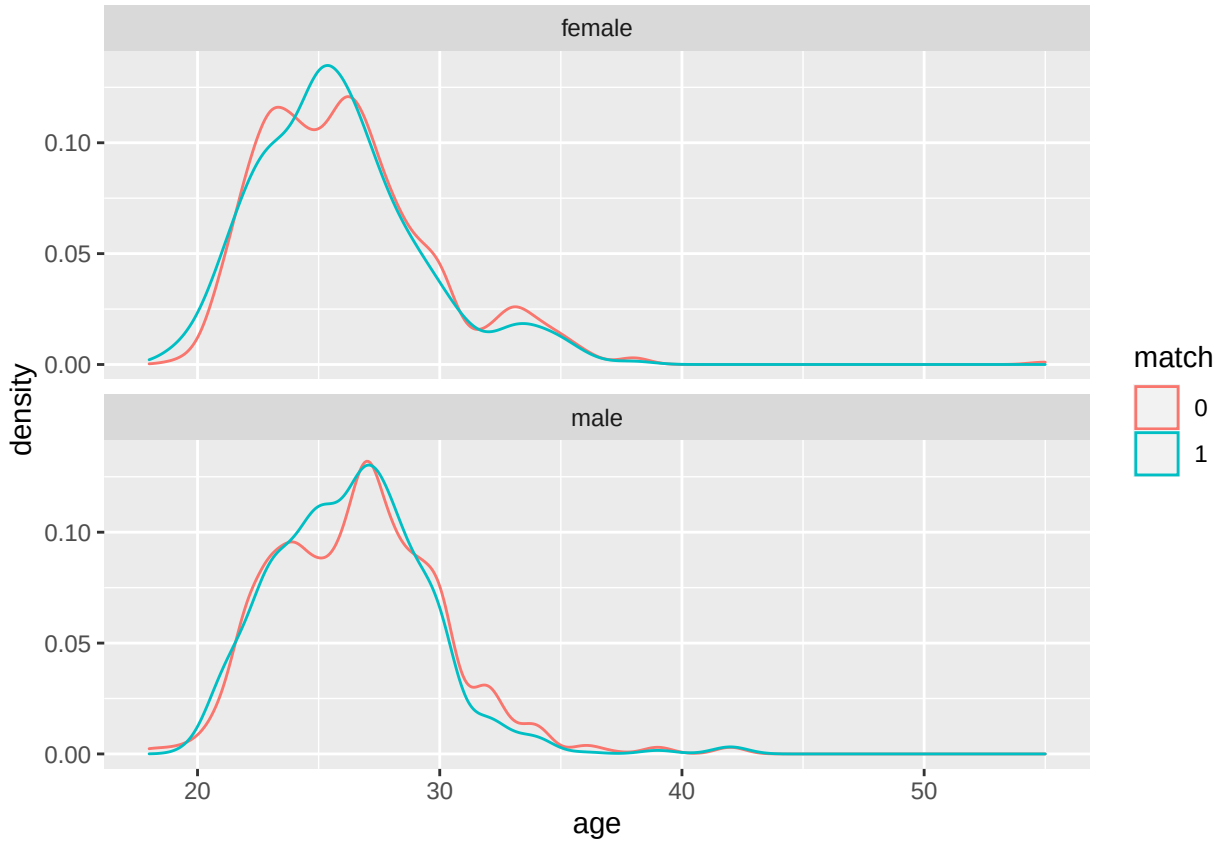


Mediana oceny atrakcyjności partnera się zmieniła - wzrosła z 5 aż do 6!

Uśrednienie zaburzyło nam efekt i różnice między postrzeganą inteligencją a ogólną atrakcyjnością wydają się być teraz mniejsze - błędnie.

5.3. Różnica wieku i płeć a matche

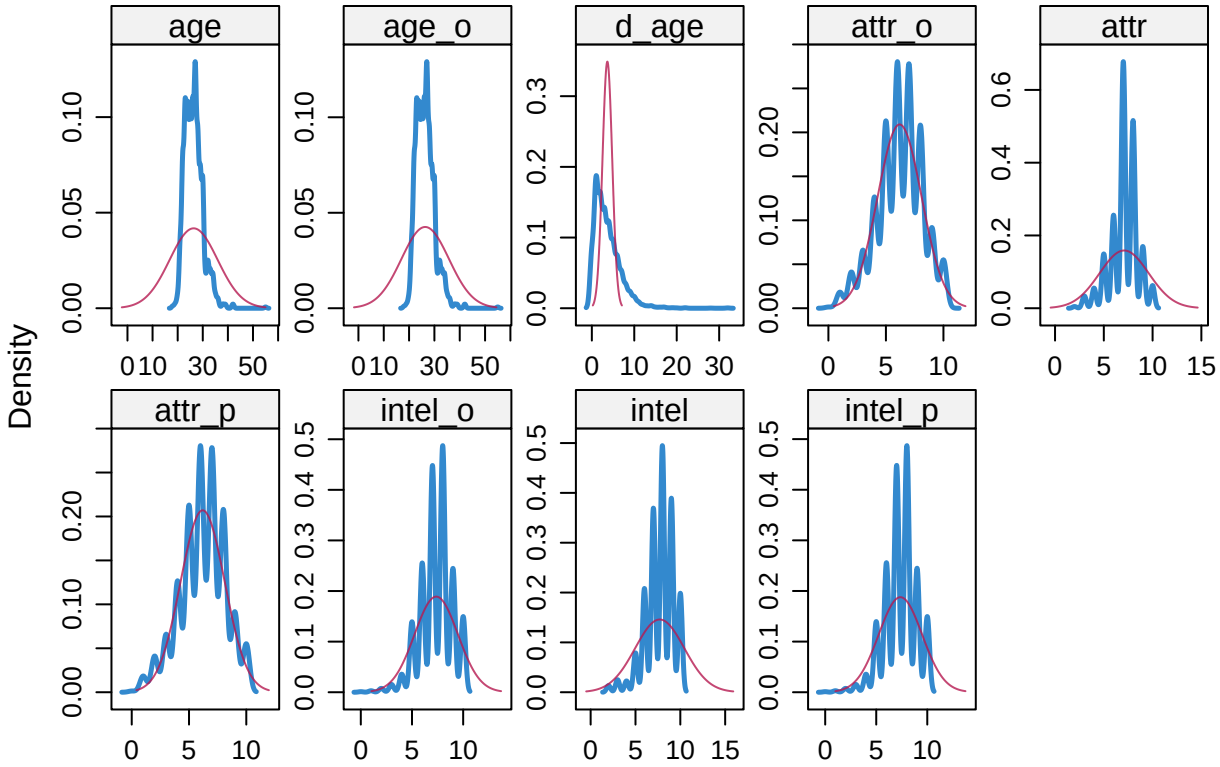
```
ggplot(data_mean, aes(x = age, color = match)) +
  geom_density() +
  facet_wrap(~gender, ncol = 1)
```



Nie widać istotnych różnic.

5.4. Zastępowanie średnimi ogólnie

```
densityplot(imp)
```



Ze względu na małą liczbę braków danych zastąpienie ich średnimi nie wpłynęło istotnie na wynik naszej analizy i wniosków z niej płynących. Można więc strzelać, że parę procent braków na każdą kolumnę to w praktyce żadna strata.

6. Wizualizacje i wnioski pustych danych zastąpionych losowymi

Zastąpmy brakujące wartości losowymi z naszej próbki.

```
imp2 <- mice(data, method = "sample", m = 1, maxit = 1)
```

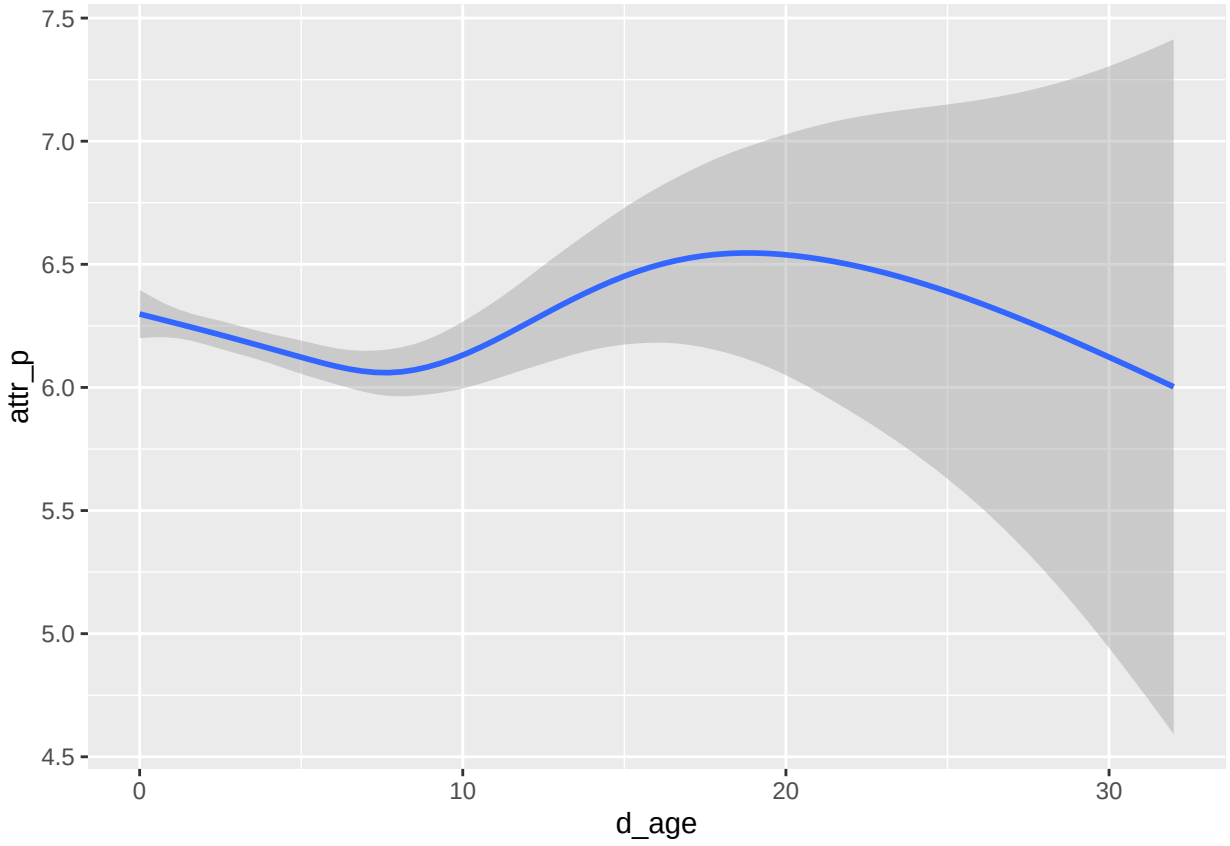
```
##
##  iter imp variable
##   1   1  age  age_o  d_age  attr_o  attr  attr_p  intel_o  intel  intel_p
```

Wypełnijmy naszą nową ramkę danych.

```
data_sample <- complete(imp2)
```

6.1. Ocena potencjalnej drugiej połówki a różnica wieku

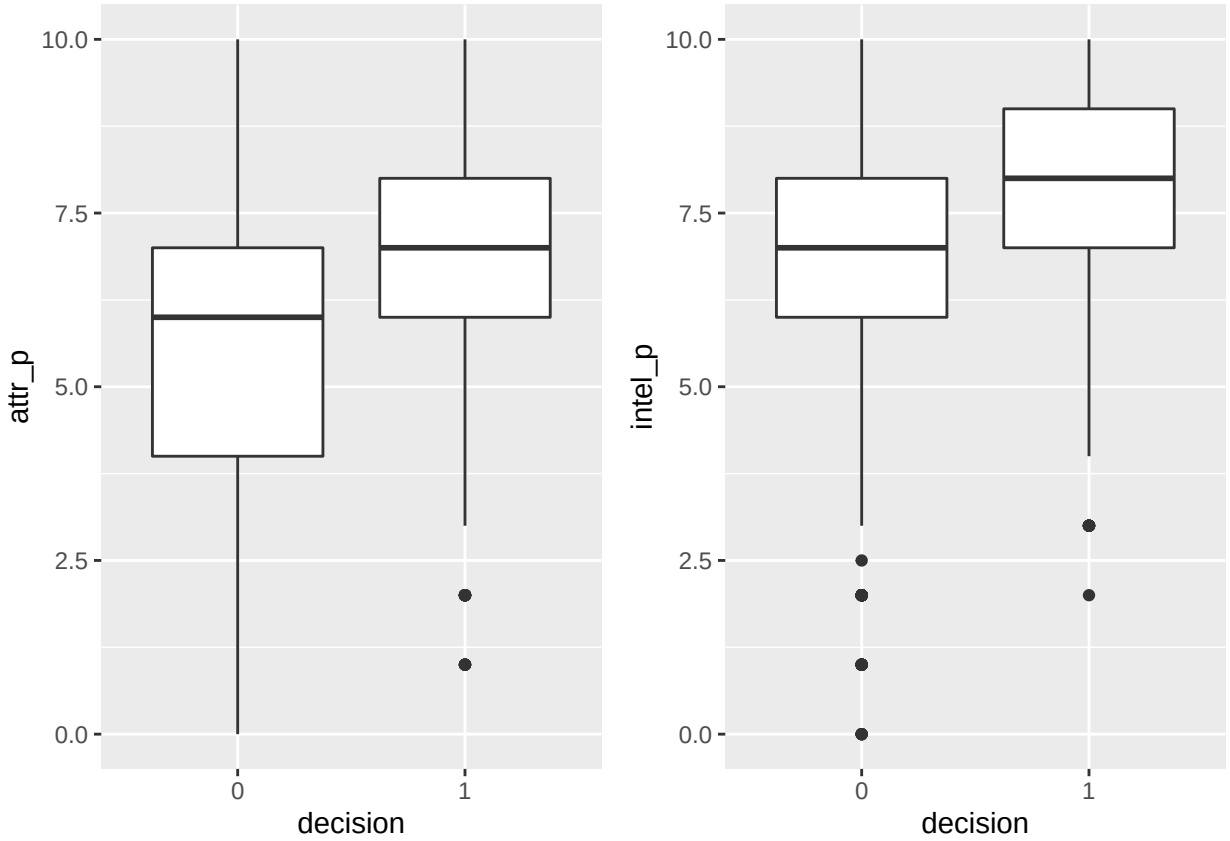
```
ggplot(data_sample,
       aes(x = d_age,
           y = attr_p)) +
  geom_smooth()
```



Bez widocznych zmian - zastąpienie losowymi nie wpłynęło na efekt.

6.2. Inteligencja i atrakcyjność partnera a decyzja

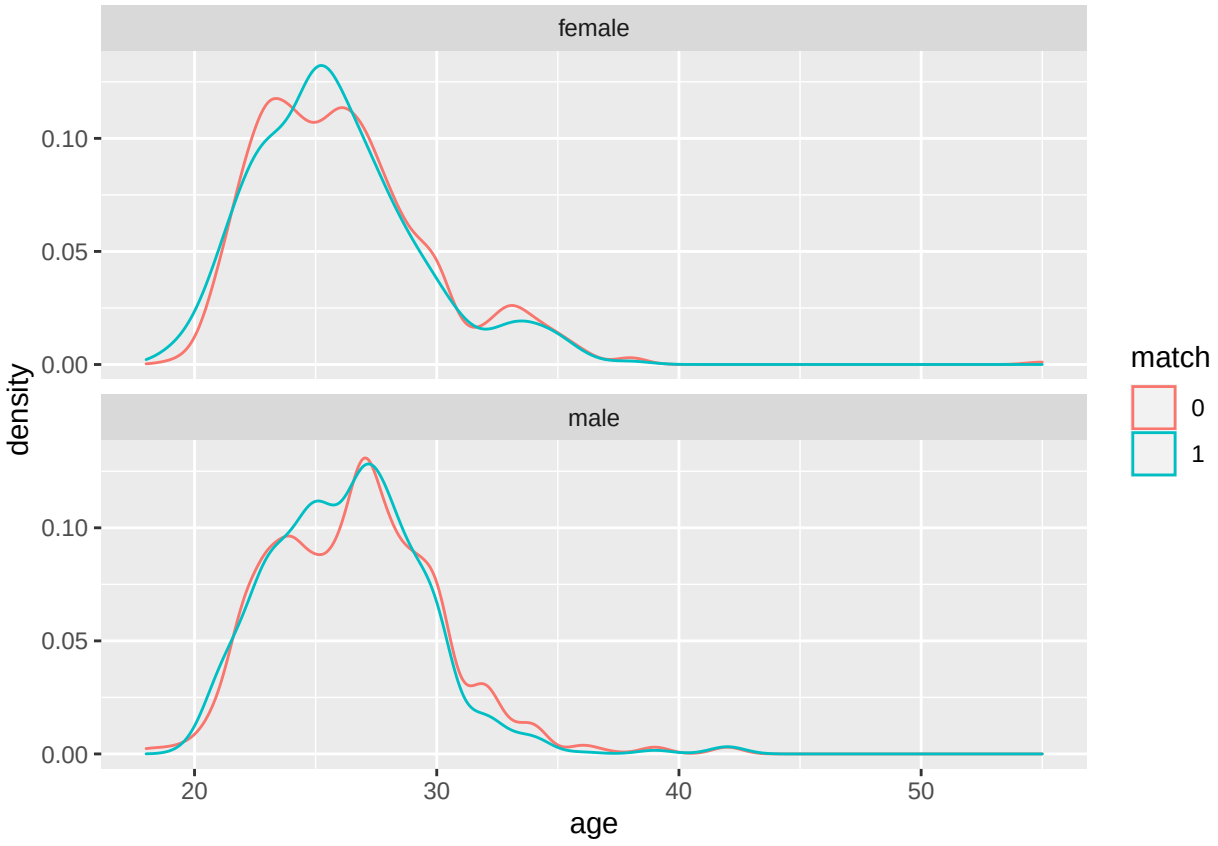
```
g1 <- ggplot(data_sample, aes(x = decision, y = attr_p)) +
  geom_boxplot()
g2 <- ggplot(data_sample, aes(x = decision, y = intel_p)) +
  geom_boxplot()
grid.arrange(g1, g2, ncol=2)
```



Sytuacja jak dla wzięcia średniej zamiast pustych danych - mediana postrzeganej atrakcyjności wzrosła.

6.3. Różnica wieku i płeć a matche

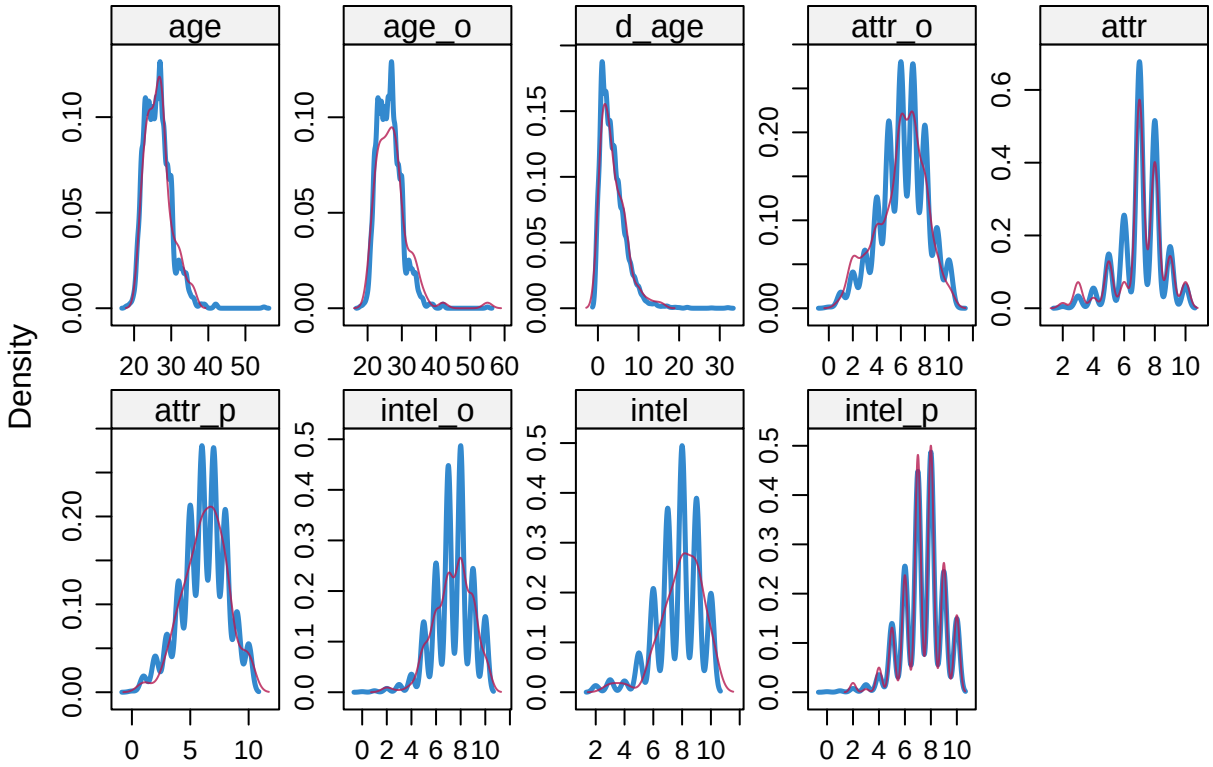
```
ggplot(data_sample, aes(x = age, color = match)) +
  geom_density() +
  facet_wrap(~gender, ncol = 1)
```



Nie widać istotnych różnic.

6.4. Zastępowanie losowymi ogólnie

```
densityplot(imp2)
```



Zastąpienie brakujących danych losowymi wartościami poskutkowało jeszcze bardziej zbliżonymi wykresami gęstości niż zastępowanie ich średnimi. Niby z matematycznego punktu widzenia ma to sens (patrz: Prawa Wielkich Liczb) i zbiorcze wyniki dla całej populacji często dla takiej modyfikacji zapewne są bardziej zbliżone do rzeczywistości niż uśredniane, takie dane można wyrzucić do kosza patrząc na jednostki - konkretnym Jankom i Aniom przypisujemy informacje zupełnie bez ładu i składu, tożto nonsens.

7. Podsumowanie oglądu danych według różnych sposobów radzenia sobie z brakami

Biorąc pod uwagę małą liczbę braków, niezależnie od usunięcia wierszy zawierających wartości NA, zastępowanie brakujących wartości średnimi czy zamienianie ich na losowe z odpowiadajej jej kolumny, nie wpłynęło na ogólną analizę danych jako na zbiorczy twór - przynajmniej na pierwszy rzut oka. Można wysnuć wniosek, że zastępowanie liczbami losowymi globalnie daje efekt bardziej zbliżony do rzeczywistości, lecz zamienianie na średnie mniej burzy obraz jednostek.

8. Algorytm uczenia maszynowego

8.1. Wprowadzenie

Zbiorami danych, na których wytrenujemy nasz algorytm będą te utworzone w rodziałach 4., 5. i 6.:

1. *data_no_na* - z usuniętymi wierszami posiadającymi wartości NA
2. *data_mean* - z wartościami brakującymi zastąpionymi średnimi
3. *data_sample* - z NA zamienionymi na losowe z odpowiadających kolumn

Cechami niech będą kolumny: *gender*, *age*, *age_o*, *d_age*, *attr_0*, *attr*, *attr_p*, *intel_o*, *intel* i *intel_p* a klasą - *match*.

Można by także dokonać ciekawej klasyfikacji trenując nasz algorytm pod prognozowanie *decision* lub *decision_o* - zdecydowałem się jednak na *match*, ponieważ to on odpowiada za finalny efekt i najważniejszy wynik randki, a przy tym na swój sposób ma "pod sobą" dwa wspomniane.

Mamy tutaj styczność z typową **klasyfikacją binarną**.

```
cechy <- colnames(data_no_na)[1:10]
klasa <- colnames(data_no_na)[13]

p <- length(cechy)
n_no <- nrow(data_no_na) # liczba wierszy w wariancie usuwania tych z NA
n <- nrow(data_mean) # liczba wierwszy w wariancie zastępowania brakujących
```

8.2. Podział na zbiór testowy i treningowy

```
set.seed(123)

id_train_n <- sample(1:n, 4/5 * n)
id_train_n_no <- sample(1:n_no, 4/5 * n_no)

df_train_no_na <- data_no_na[id_train_n_no, c(cechy, klasa)]
df_train_sample <- data_sample[id_train_n, c(cechy, klasa)]
df_train_mean <- data_mean[id_train_n, c(cechy, klasa)]

id_test_n <- (1:n)[-id_train_n]
id_test_n_no <- (1:n_no)[id_train_n_no]

df_test_no_na <- data_no_na[id_test_n_no, c(cechy, klasa)]
df_test_sample <- data_sample[id_test_n_no, c(cechy, klasa)]
df_test_mean <- data_mean[id_test_n, c(cechy, klasa)]
```

Dzięki takim samym wylosowanym indeksowym treningowym, będziemy mogli porównać wyniki dla *data_sample* i *data_mean* dla tych samych wierszach. Operacja będzie niestety niemożliwa dla *data_no_na* ze względu na inną wielkość analizowanych wartości.

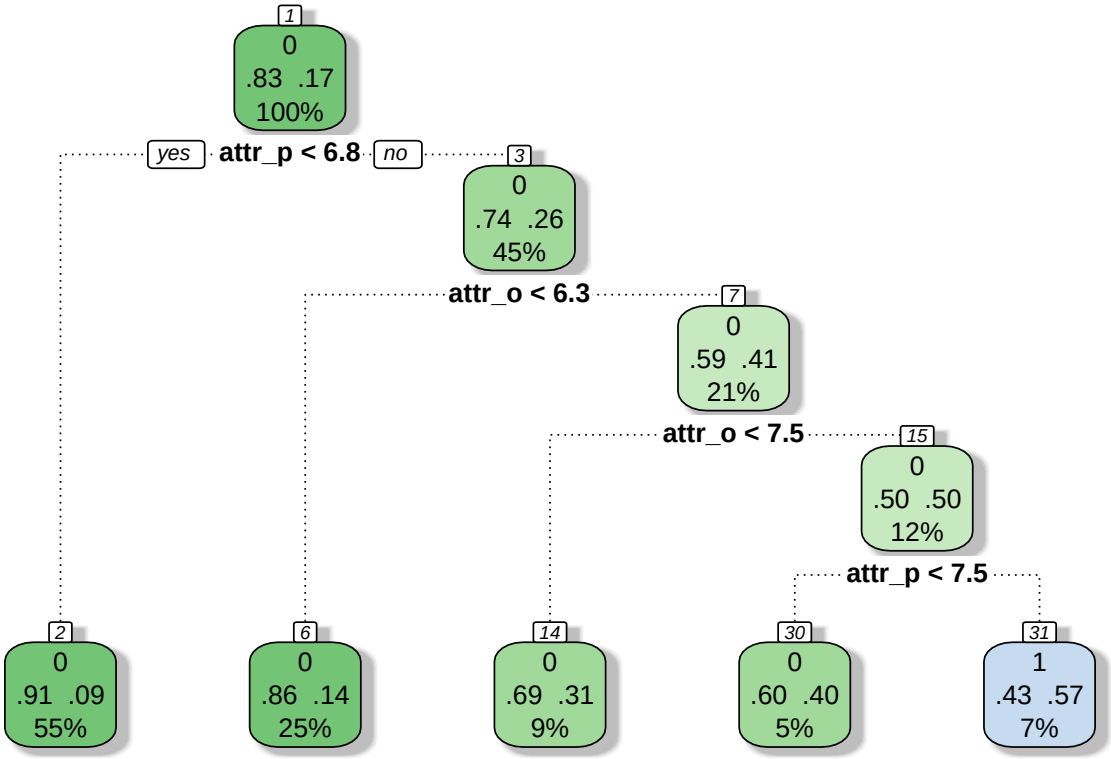
8.3. Klasyfikacja z pomocą drzewa klasyfikacyjnego

No nie powiem, urzekły mnie te wesołe binarne cosie z labów trzecich.

Przetwórzmy najpierw dla ramki z usuniętymi wierszami zawierającymi NA.

```
# install.packages("rpart") # if not installed
library(rpart)
tree_classifier_no_na <- rpart(match~., data = df_train_no_na)
par(mar = c(1,1,1,1))
par(xpd = TRUE)

# install.packages("rattle") # if not installed
library(rattle)
fancyRpartPlot(tree_classifier_no_na, caption = NULL)
```



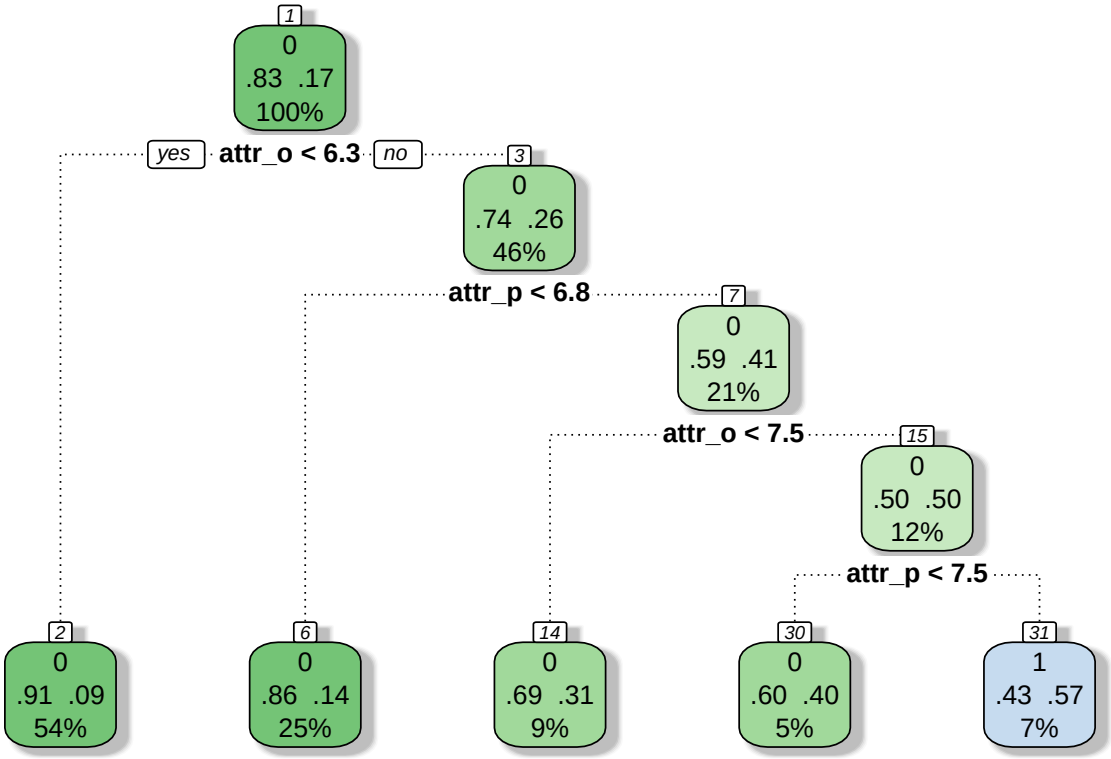

```
predict(tree_classifier_no_na, newdata = df_test_no_na[,cechy], type="class")[1:20]
```

```
##   50 6375 3752 1351 7573 7620 4076 1851 3033 5835 8179 1650 3941  696 2145
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## 5370 8214 1021 1045 7378
##    0    0    1    0    0
## Levels: 0 1
```

```
predict(tree_classifier_no_na, newdata = df_test_no_na[,cechy], type="prob")[1:20]
```

```
##  [1] 0.9104700 0.9104700 0.9104700 0.9104700 0.9104700 0.9104700 0.9104700
##  [8] 0.8610013 0.9104700 0.6040268 0.9104700 0.9104700 0.9104700 0.9104700
## [15] 0.9104700 0.8610013 0.6040268 0.4302885 0.9104700 0.8610013
```

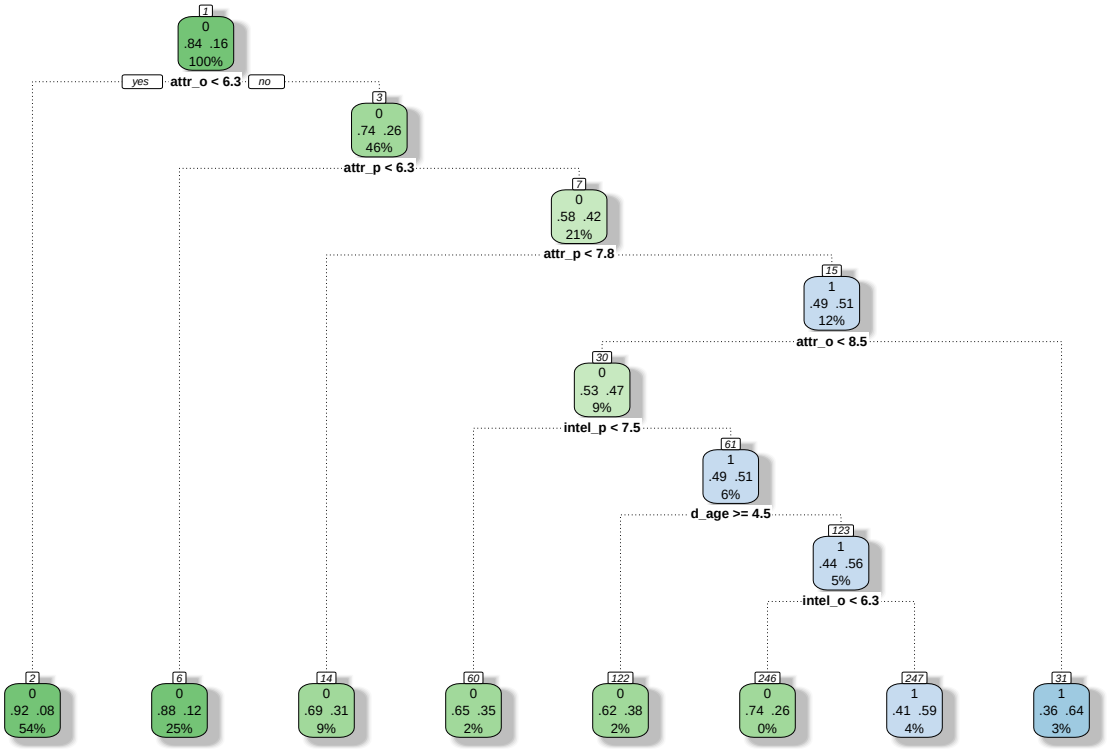
```
tree_classifier_parameter_change_no_na <- rpart(match~., data=df_test_no_na,
  parms = list(split = 'information'),
  minsplit = 10,
  cp = 0.01)
fancyRpartPlot(tree_classifier_parameter_change_no_na, caption = NULL)
```



O proszę! Wychodzi na to, że algorytm stwierdził, że to jednak atrakcyjność ma kluczową wartość - o wiele większą niż inteligencja. Sprawdźmy jeszcze efekt dla dwóch pozostałych utworzonych ramek.

```
tree_classifier_sample <- rpart(match~., data = df_train_sample)
par(mar = c(1,1,1,1))
par(xpd = TRUE)

fancyRpartPlot(tree_classifier_sample, caption = NULL)
```



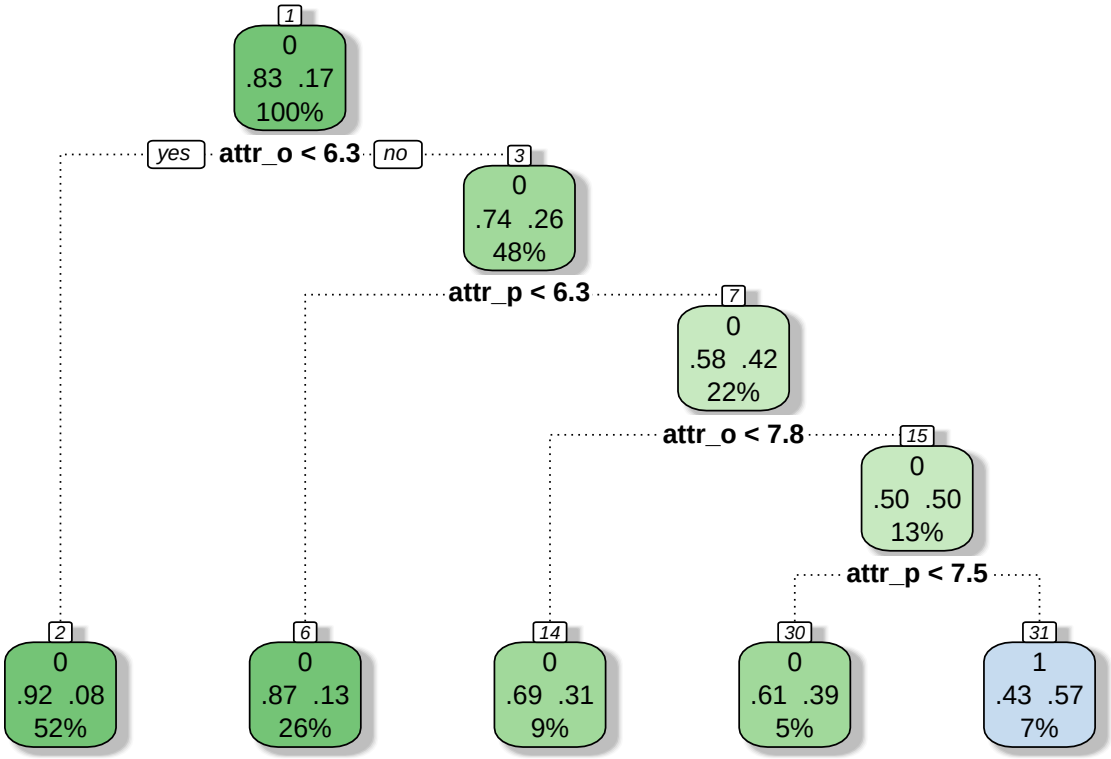

```
predict(tree_classifier_sample, newdata = df_test_sample[,cechy], type="class")[1:20]
```

```
##   51 5871 3525 1245 6976 7016 3835 1707 2830 5367 7525 1542 3704  652 1979
##    0    1    0    0    0    0    0    1    0    0    0    0    0    0    0
## 4980 7551  916  940 6814
##    0    0    0    0    0
## Levels: 0 1
```

```
predict(tree_classifier_sample, newdata = df_test_sample[,cechy], type="prob")[1:20]
```

```
##  [1] 0.8779481 0.4084507 0.9178502 0.9178502 0.8779481 0.8779481 0.8779481
##  [8] 0.3623188 0.9178502 0.8779481 0.9178502 0.9178502 0.8779481 0.9178502
## [15] 0.6540881 0.6540881 0.8779481 0.9178502 0.8779481 0.9178502
```

```
tree_classifier_parameter_change_sample <- rpart(match~., data=df_test_sample,
  parms = list(split = 'information'),
  minsplit = 10,
  cp = 0.01)
fancyRpartPlot(tree_classifier_parameter_change_sample, caption = NULL)
```

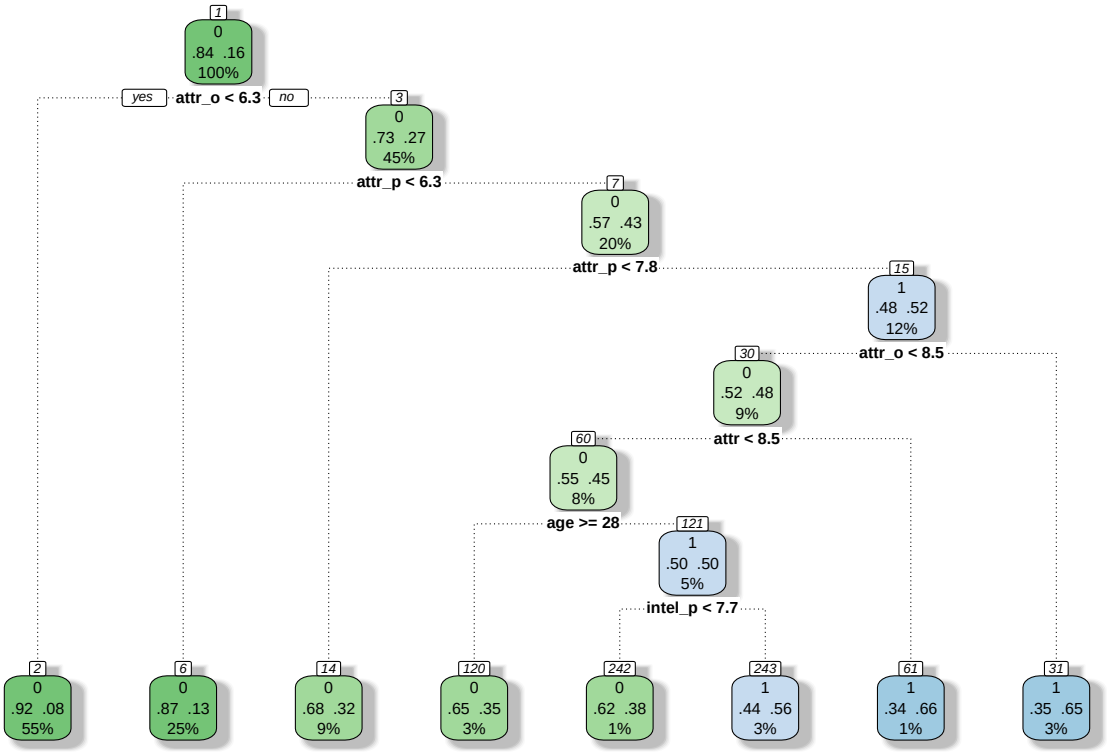


Dla wylosowanych wartości "wymagania" na match w liczbach wzrosły, ale w praktyce tylko pozornie - wciąż na porozumienie według drzewek mogą liczyć tylko ci o ocenie własnej atrakcyjności i swojego partnera równej przynajmniej 8.

A jak będzie wyglądał wynik dla uśrednionych liczb?

```
tree_classifier_mean <- rpart(match~., data = df_train_mean)
par(mar = c(1,1,1,1))
par(xpd = TRUE)

fancyRpartPlot(tree_classifier_mean, caption = NULL)
```



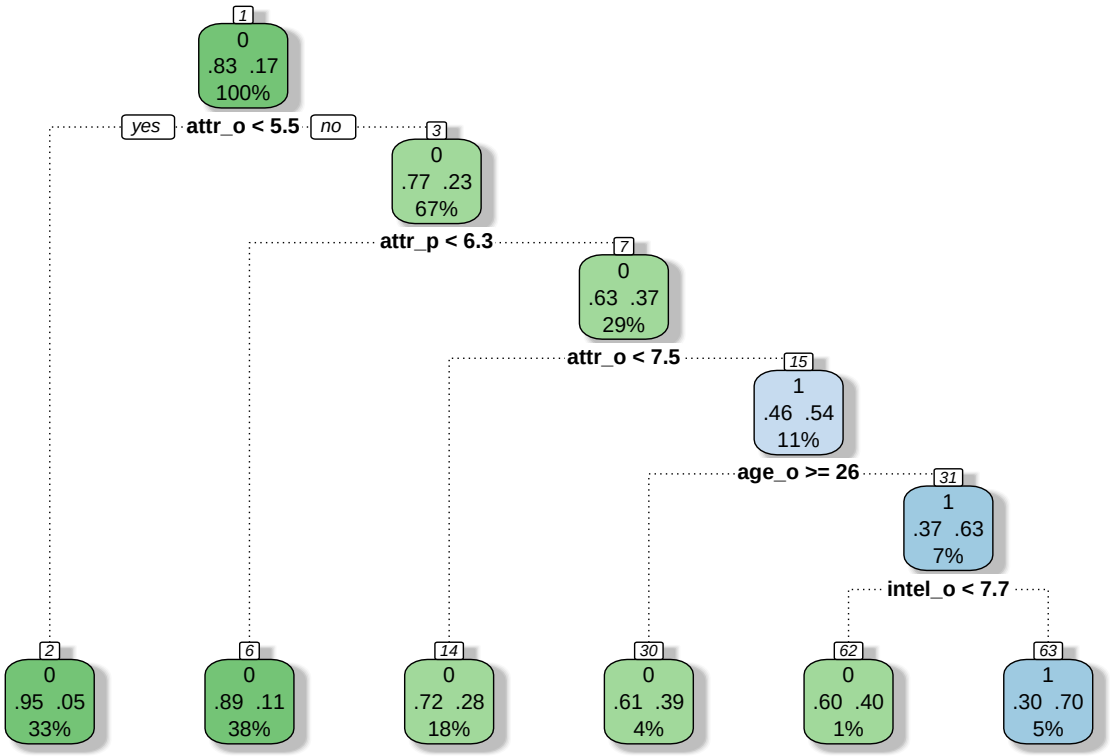

```
predict(tree_classifier_mean, newdata = df_test_mean[,cechy], type="class")[1:20]
```

```
##  8 10 12 14 19 24 29 33 34 37 40 43 46 47 48 50 69 70 74 80
##  0  0  0  0  0  1  1  0  0  0  1  0  0  0  0  0  0  0  0  0
## Levels: 0 1
```

```
predict(tree_classifier_mean, newdata = df_test_mean[,cechy], type="prob")[1:20]
```

```
##  [1] 0.9193241 0.9193241 0.6185567 0.6830508 0.6830508 0.4434783 0.4434783
##  [8] 0.8747742 0.6185567 0.6830508 0.4434783 0.8747742 0.9193241 0.9193241
## [15] 0.9193241 0.9193241 0.8747742 0.6185567 0.6185567 0.6830508
```

```
tree_classifier_parameter_change_mean <- rpart(match~., data=df_test_mean,
  parms = list(split = 'information'),
  minsplit = 10,
  cp = 0.01)
fancyRpartPlot(tree_classifier_parameter_change_mean, caption = NULL)
```



Niesamowite! Nagle dodatkowym "warunkiem"... stał się wiek partnera przynajmniej 26 lat.

Podsumowując metoda drzewa klasyfikacyjnego wydaje się być mocno taka sobie - z zaproponowanych 10 cech wzięła pod uwagę tylko dwie, a w przypadku uzupełnienia braków danych średnimi wartościami - trzy z nich. Wynik jest bardzo ogólny i mało satysfakcjonujący