

# Projekt Wnioskowanie Bayesowskie

Aleksander Mackiewicz-Kubiak

## ZADANIE

W pakiecie bayesrules znajdują się dane fake news zawierające informacje o 150 artykułach informacyjnych, niektórych prawdziwych, a niektóre fałszywych. Wykonaj analizę regresji logistycznej zmiennej type, przy użyciu przynajmniej trzech predyktorów.

## BIBLIOTEKI

```
library(bayesrules)
library(rstanarm)
library(bayesplot)
library(tidybayes)
library(tidyverse)
library(broom.mixed)
library(gridExtra)
library(loo)
```

## PREPROCESSING DANYCH

Dane dotyczą prawdziwości 150 artykułów. Wpierw ze zbioru danych usuń zbędne dane, takie jak pełne tytuły artykułów, ich treści czy dane o autorach, gdyż żadnej z tych rzeczy nie będzie można użyć jako zmiennej objaśniającej.

```
data(fake_news)
fake_news <- fake_news %>%
  select(-title, -text, -url, -authors)
```

Drugą ważną kwestią w zbiorze danych są zmienne podające te same wartości tylko w innych formatach (jedna zmienna z wartością liczbowych i druga z tą samą wartością wyrażoną w procentach). Pozbywam się takich kolumn:

```
fake_news <- fake_news %>%
  select(-title_caps_percent, -text_caps_percent, -title_excl_percent, -text_excl_percent, -text_syllab
```

Dosyć duża zbieżność zachodzi między parami zmiennych text\_words i text\_char, oraz title\_words i title\_char, gdyż ilość znaków jest bezpośrednio powiązana z ilością słów w artykule. Można by się tu doszukiwać teorii, że np. fałszywe artykuły używają krótszych i prostszych słów, ale jest to zbyt daleka teoria jak na tą analizę, dlatego po prostu pozbędę się zmiennych z nazwą char.

```
fake_news <- fake_news %>%
  select(-text_char, -title_char)
```

Ostatnią zmianą w zbiorze danych będzie ujednolicenie wszystkich danych na temat słów i ich przyporządkowań. Dla swojej analizy skorzystam tylko z tych ogólnych grup słów: pozytywnych i negatywnych. Dla jeszcze większego uproszczenia, stworzę z nich pojedynczą zmienną, która będzie wyrażać stosunek słów pozytywnych do negatywnych w danym artykule. Ponieważ wartość zmiennej negative może być równa zero, to dodaje do niej drobną wartość, by nie było problemu z dzieleniem przed 0.

```
fake_news <- fake_news %>%
  mutate(posnegratio = positive / (negative + 0.001))
```

Zmienną objaśnianą będzie zmienna type. Sprawdę jakie wartości przyjmuje ta zmienna:

```
head(fake_news$type)
```

```
## [1] fake real fake real fake real
## Levels: fake real
```

Jest to zmienna binarna o wartościach real i fake. Zatem do modelowania jej posłuży mi regresja logistyczna

## REGRESJA LOGISTYCZNA

Do modelowania zmiennej binarnej, czyli zmiennej przyjmującej wartości  $\{0, 1\}$  służy regresja logistyczna. Parametrem  $\theta$  przy takim podejściu jest prawdopodobieństwo, że modelowana zmienna Y przyjmuje wartość 1. W moim przypadku, tą zmienną jest zmienna type, której wartości to fake lub true. Dla moich modeli przypisze 0 do wartości fake, a 1 do wartości real.

Przy modelu regresji logistycznej wyliczane parametry nie będą dotyczyć samego parametru  $\theta$ , a wartości szans, czyli wartości danej wzorem  $\frac{\theta}{1-\theta}$ . Sam model będzie przyporządkowywał parametry  $\beta_0, \beta_1, \dots, \beta_n$  zmiennym  $X_1, X_2, \dots, X_n$  dla logarytmu szans, czyli dla  $\ln(\frac{\theta}{1-\theta})$ :  $\ln(\frac{\theta}{1-\theta}) = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n$ . Stąd wzór do wyliczenia samego parametru  $\theta$  to  $\theta = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n}}$ . Każdy  $X_n$  oznacza jedną ze zmiennych uwzględnionych w modelu,  $\beta_n$  to współczynniki dla tych zmiennych, a sama  $\beta_0$  daje wartość szans, gdy wszystkie zmienne przyjmą wartość 0. By interpretować wyniki  $\beta_n$  dla tworzonych modeli, należy wpierw liczyć ich eksponent, czyli  $e^{\beta_n}$ . Tak wyliczoną wartość odczytuje się tak: o ile % zmienią się szanse na Y=1, jeśli wartość wybranej zmiennej  $X_n$  zwiększy się o 1, oraz żadna inna zmieni nie zmieni swojej wartości.

Według zasad regresji bayesowskiej, wpierw potrzebujemy stworzyć model apriori, z pewnymi założeniami bazującymi na empirycznych danych lub intuicji. Ich wybór nie będzie miał wielkiego wpływu na końcowy wygląd modelu, a nawet można przy pomocy gotowych funkcji w R automatycznie te założenia poprawiać. Następnie, na podstawie zestawu danych, stworzę model aposteriori, który będzie modelem predykejnym wybranej zmiennej objaśnianej, czyli w moim przypadku zmiennej type.

Ponieważ mam w danych 150 wartości zmiennej type, mogę wyliczyć wartość parametru  $\theta$  dla apriori:

```
table(fake_news$type)
```

```
##
## fake real
##   60   90
```

```
ratio=90/(60+90); cat("ratio =", ratio)
```

```
## ratio = 0.6
```

Następnie z tego wyniku wyliczam szansę:

```
cat("szansa =", ratio/(1-ratio))
```

```
## szansa = 1.5
```

I całość logarytmuję, by dostać wartość  $\beta_0$  dla rozkładu apriori.

```
log(ratio/(1-ratio))
```

```
## [1] 0.4054651
```

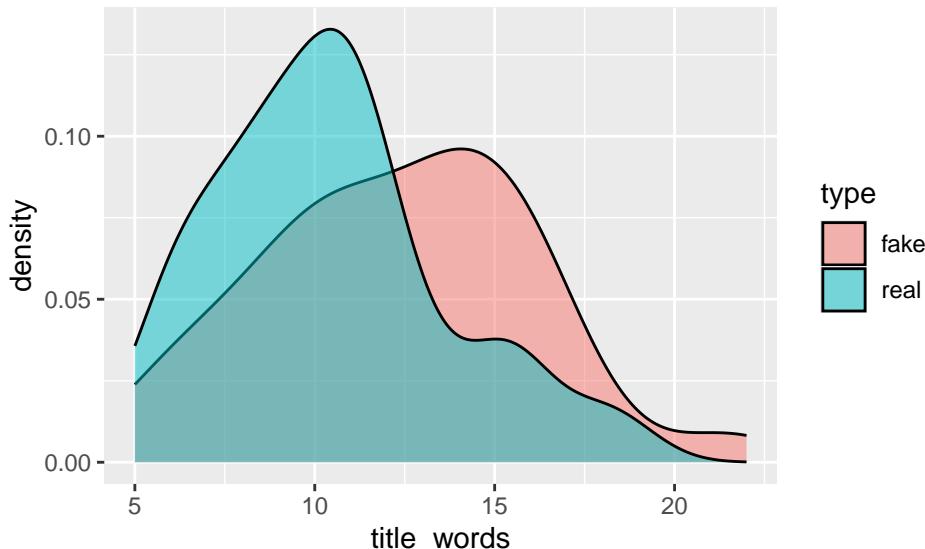
Dla estetyki obliczeń, w każdym z tworzonych modeli będę zaokrągał tę wartość do 0.41. Wartość odchylenia dla rozkładu apriori nie można już w taki sposób oszacować, a jedyne dopasować na podstawie domysłów, jak mocno wrażliwe i zmienne są dane z rozkładu apriori. Ja przyjmę wartość  $sd=0.63$ .

## BADANIE ZALEŻNOŚCI

Mając zdefiniowany typ modelu i regresji, jaką będę stosować, przygotowaną zmienną objaśnianą i potencjalne zmienne objaśniające, nie pozostało nic innego jak po kolei przypatrywać się zmiennym.

Pierwszą zmienną, której się przyjrze to zmienna title\_words.

```
ggplot(fake_news, aes(x = title_words, fill = type)) +  
  geom_density(alpha = 0.5)
```



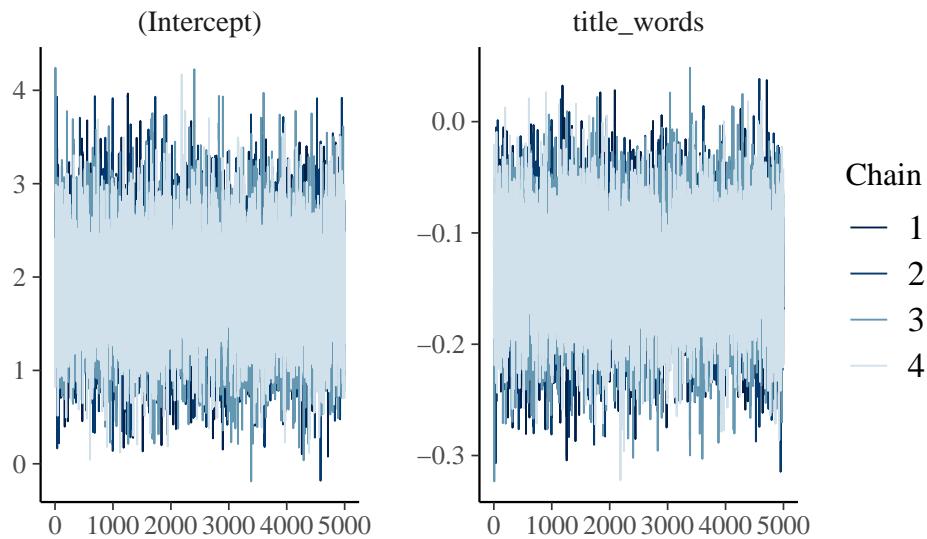
Na wykresach widać różnice wartości title\_words dla różnych typów artykułów, chociaż są widoczne pewne pokrycia. Zatem stwierdzam, że jest zależność między tymi zmiennymi, i buduje pierwszy model:

```
model1 <- stan_glm(type ~ title_words,
                     data = fake_news, family = binomial,
                     prior_intercept = normal(0.41, 0.63),
                     prior = normal(0.1, 0.5, autoscale = T),
                     chains = 4, iter = 5000*2, seed = 12345)
```

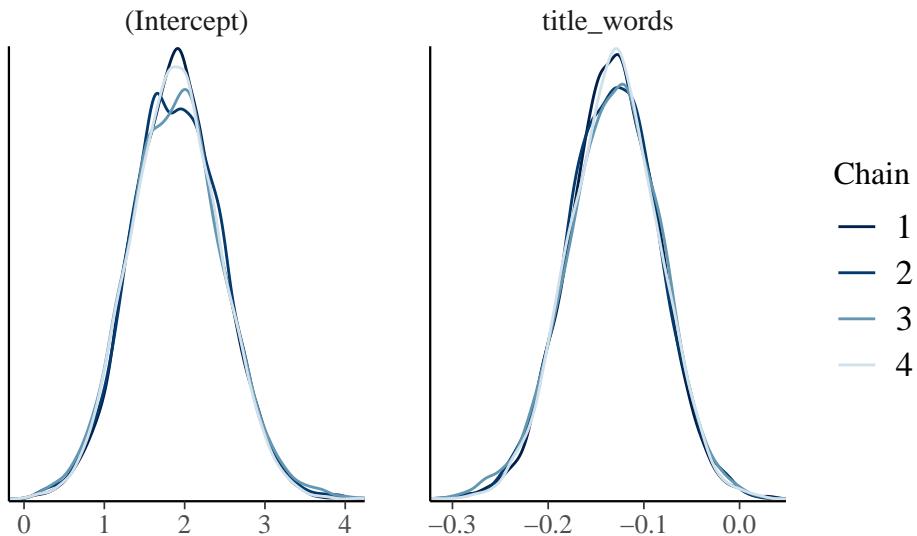
```
tidy(model1, effects = "fixed")
```

```
## # A tibble: 2 x 3
##   term      estimate std.error
##   <chr>     <dbl>     <dbl>
## 1 (Intercept)  1.89     0.570
## 2 title_words -0.131    0.0477
```

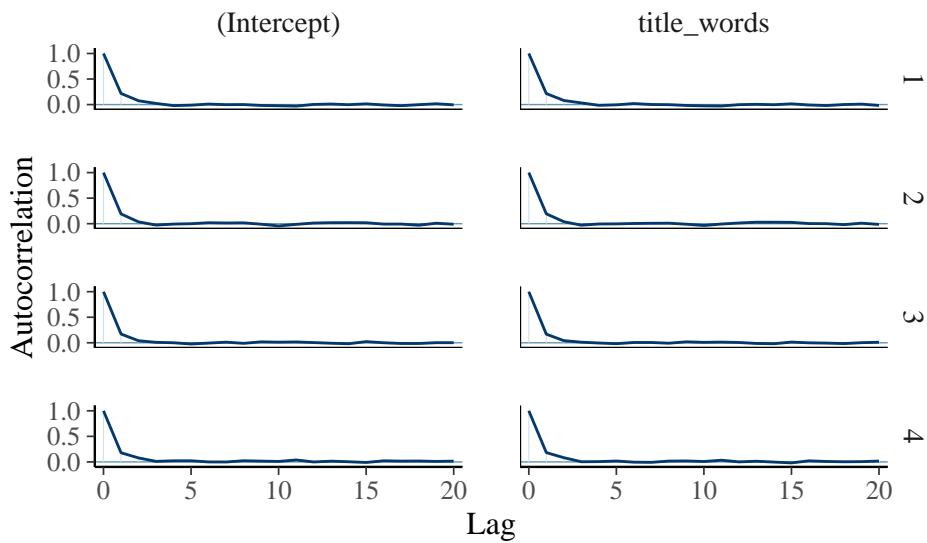
```
mcmc_trace(model1)
```



```
mcmc_dens_overlay(model1)
```



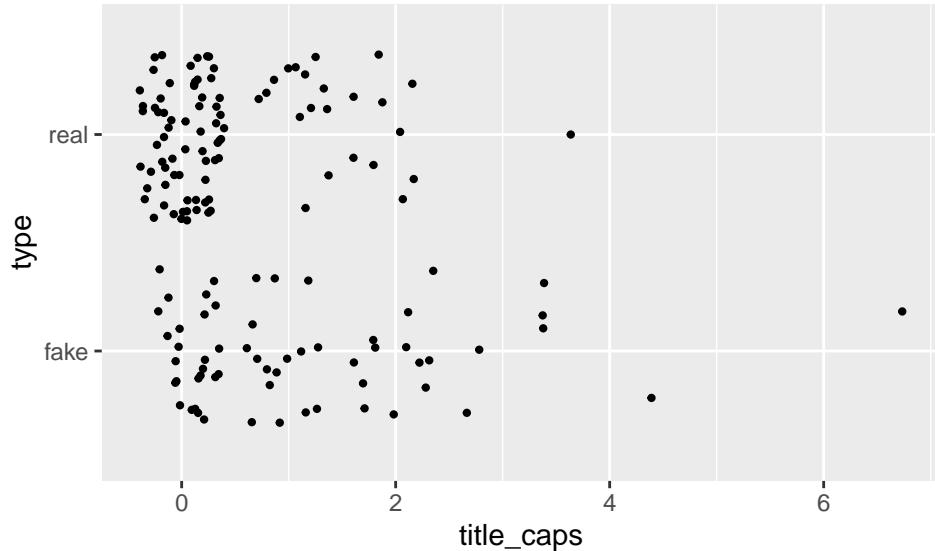
```
mcmc_acf(model1)
```



Pierwszy model wygląda dobrze, symulacja przebiegła bez problemów, autokorelacja dla każdego łańcucha się zeruje. Widzimy pewne wartości parametrów  $\beta$ , których interpretacją zajmę się dopiero przy modelu końcowym. Natomiast wartości tutaj wskazane wydają się być możliwe i w miarze normalne. Zwłaszcza fakt, że  $\beta_1$  jest ujemna jest dobrym znakiem, bo oznacza to, że im więcej słów w tytule, tym mniejsza szansa że artykuł jest prawdziwy. Pokrywa się to z wnioskami, które można wyciągnąć z poprzedniego wykresu.

Drugą badaną zmienną będzie zmienna title\_caps.

```
ggplot(fake_news, aes(x = title_caps, y = type)) +  
  geom_jitter(size = 0.8)
```



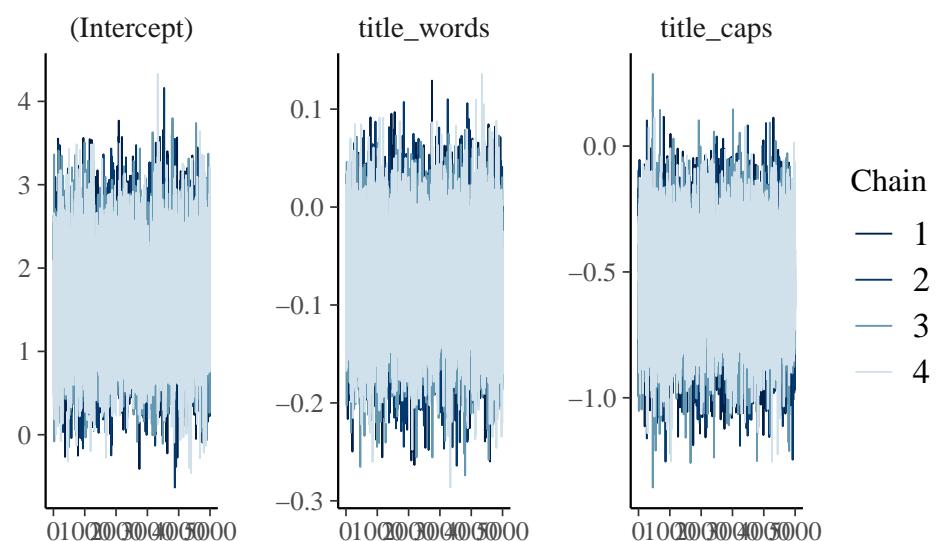
Widac o wiele większą gęstość w 0 dla rzeczywistych artykułów, co sugeruje, że ta zmienna może być przydatna dla modelowania zmiennej type, zatem ją też dodaje do modelu:

```
model2 <- stan_glm(type ~ title_words+title_caps,
                      data = fake_news, family = binomial,
                      prior_intercept = normal(0.41, 0.63),
                      prior = normal(0.1, 0.5, autoscale = T),
                      chains = 4, iter = 5000*2, seed = 12345)
```

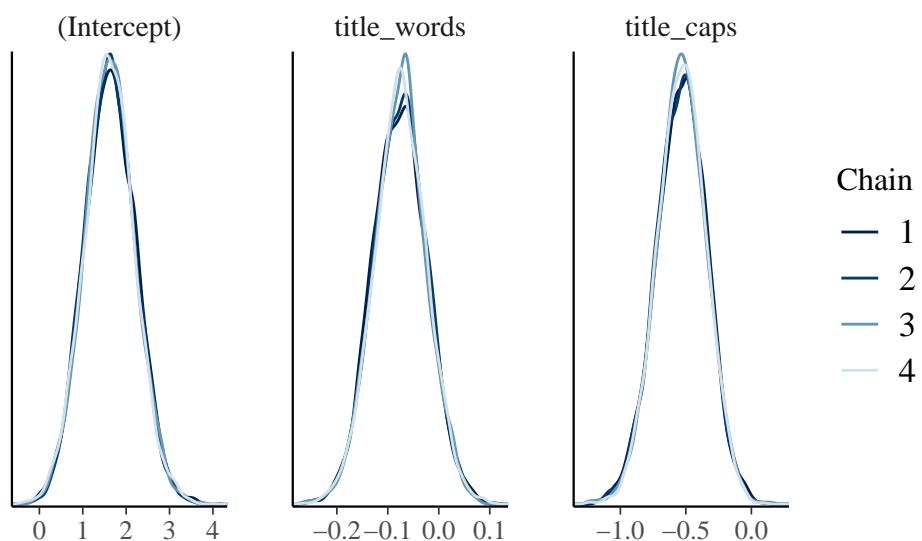
```
tidy(model2, effects = "fixed")
```

```
## # A tibble: 3 x 3
##   term      estimate std.error
##   <chr>      <dbl>     <dbl>
## 1 (Intercept) 1.62      0.580
## 2 title_words -0.0751    0.0516
## 3 title_caps  -0.530     0.188
```

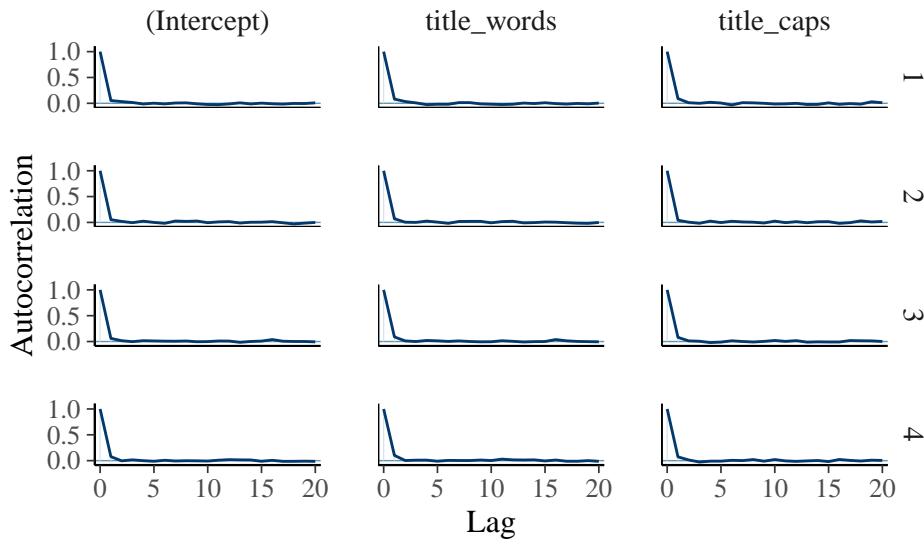
```
mcmc_trace(model2)
```



```
mcmc_dens_overlay(model2)
```



```
mcmc_acf(model2)
```

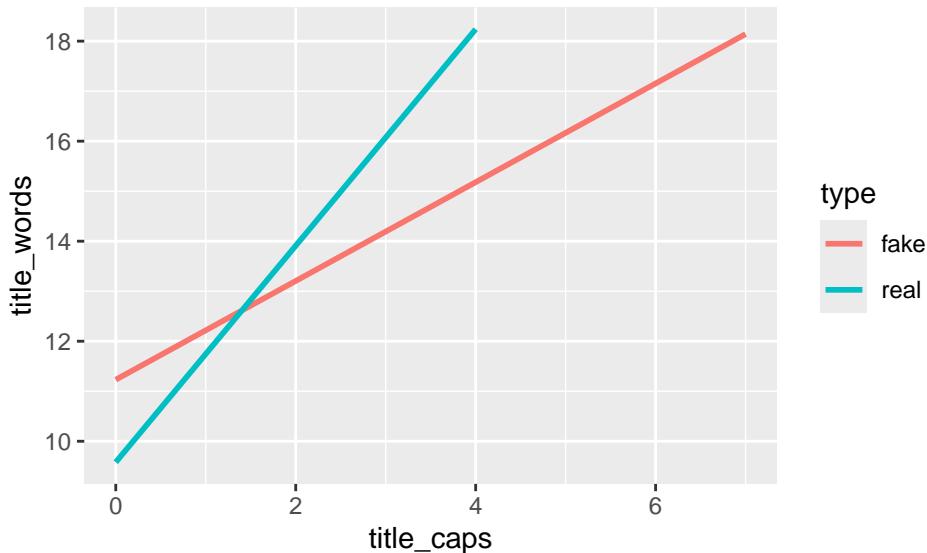


Ponownie model nie wygląda, by było z nim coś nie tak. Wszystkie wykresy, jak i wartości  $\beta$  wydają się być w normie.

Kolejnym krokiem będzie zbadanie interakcji między zmiennymi `title_words` i `title_caps`, gdyż obie zmienne dotyczą tytułów artykułów, i bardzo możliwe że ilość słów pisanych dużymi literami w artykule będzie zależeć od tego ile tych słów tam będzie (im więcej słów tym chociażby większa szansa na pojawienie się takich pisanych wielkimi literami). Wpierw wykres zależności między zmiennymi:

```
ggplot(fake_news, aes(y = title_words, x = title_caps, color = type)) +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



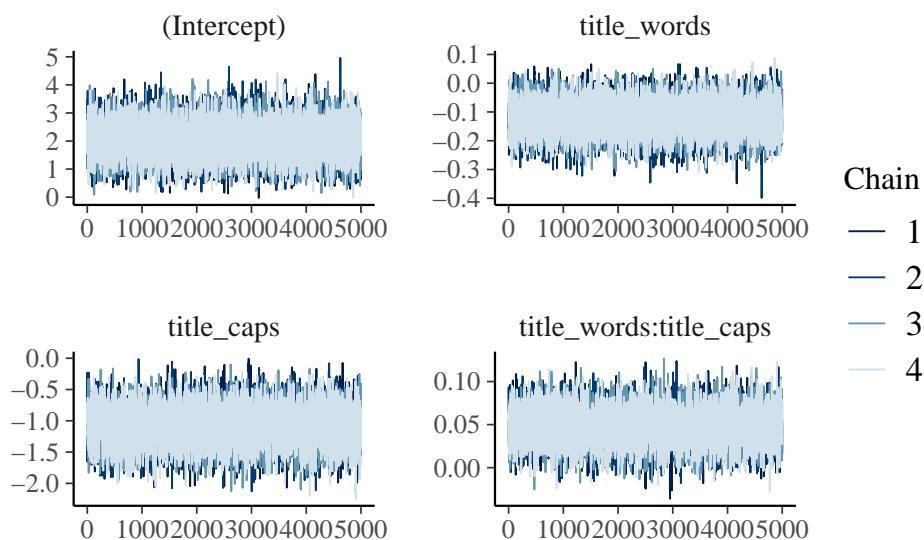
Dla różnych typów artykułów ich zależności nie są równoległe, zatem zachodzi interakcja między zmiennymi. Różnica nachylenia nie jest jednak bardzo duża, zatem interakcja może nie być wystarczająco istotna. Niemniej jednak spróbujmy dodać ją do modelu:

```
model3 <- stan_glm(type ~ title_words+title_caps+title_words:title_caps,
                     data = fake_news, family = binomial,
                     prior_intercept = normal(0.41, 0.63),
                     prior = normal(0.1, 0.5, autoscale = T),
                     chains = 4, iter = 5000*2, seed = 12345)
```

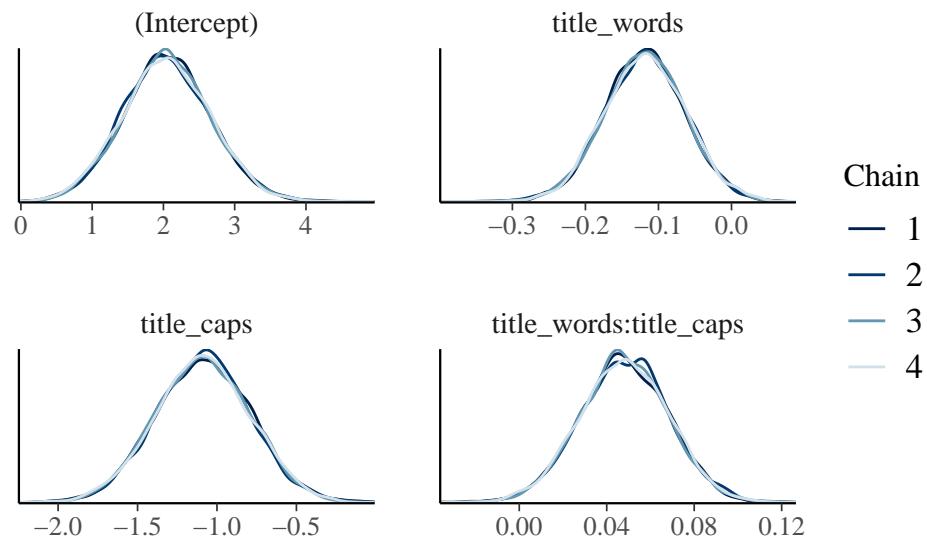
```
tidy(model3, effects = "fixed")
```

```
## # A tibble: 4 x 3
##   term            estimate std.error
##   <chr>          <dbl>     <dbl>
## 1 (Intercept)    2.04      0.611
## 2 title_words   -0.118     0.0553
## 3 title_caps     -1.09     0.300
## 4 title_words:title_caps  0.0480   0.0201
```

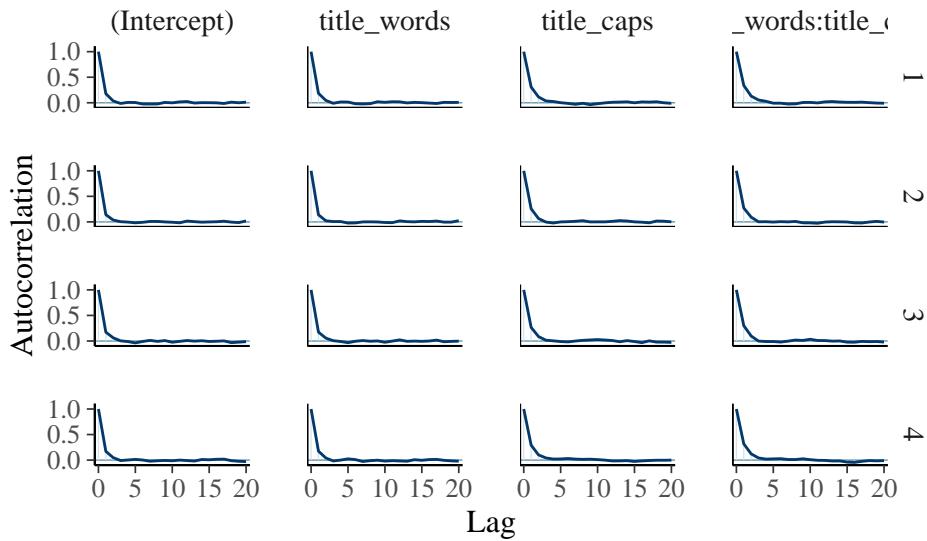
```
mcmc_trace(model3)
```



```
mcmc_dens_overlay(model3)
```



```
mcmc_acf(model3)
```



Trzeci model również prezentuje się dobrze. Sprawdzę teraz jaki wpływ ma dodana interakcja do modelu:

```
posterior_interval(model3, prob = 0.9)
```

```
##                                     5%          95%
## (Intercept)      1.03409415  3.08318799
## title_words     -0.20966823 -0.02792907
## title_caps      -1.57870525 -0.60202164
## title_words:title_caps  0.01461108  0.08118836
```

Współczynnik  $\beta$  przy interakcji według tego modelu jest w przedziale od 0.014 do 0.08, co intuicyjnie wydaje się być bardzo małą wartością. Jednak ponieważ jest to model regresji logistycznej, to wartości  $\beta$  należy interpretować razem z eksponentem:

```
exp(0.01461108); exp(0.08118836)
```

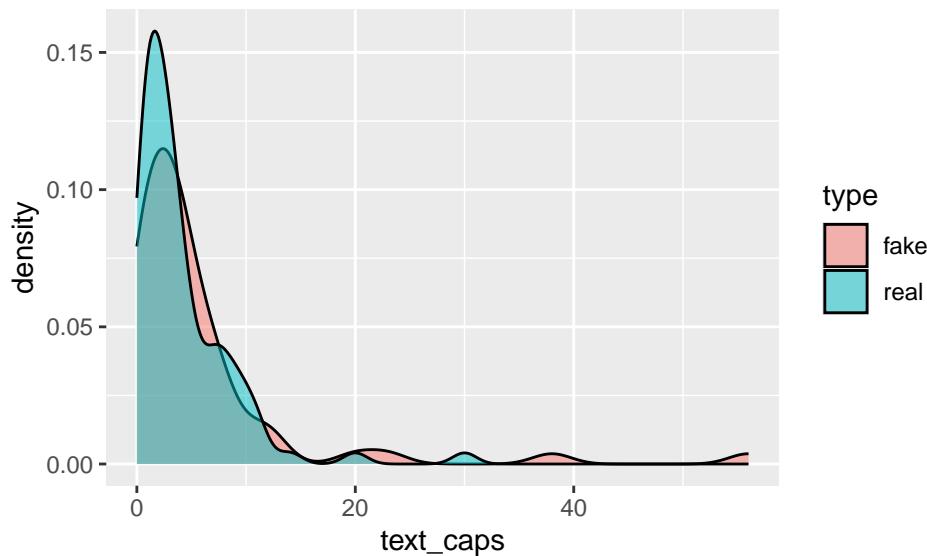
```
## [1] 1.014718
```

```
## [1] 1.084575
```

Dla takich małych wartości  $\beta$ , operacja  $e^\beta$  praktycznie ich nie zmienia. Oznacza to, że wzrost interakcji między title\_words i title\_caps o 1 wartość sprawi, że szanse że artykuł będzie fałszywy wzrosną między 1.4%-8.5%. O ile nie są to bardzo duże wartości, to jednak nie są one do pominięcia, zatem interakcja pozostanie w modelu.

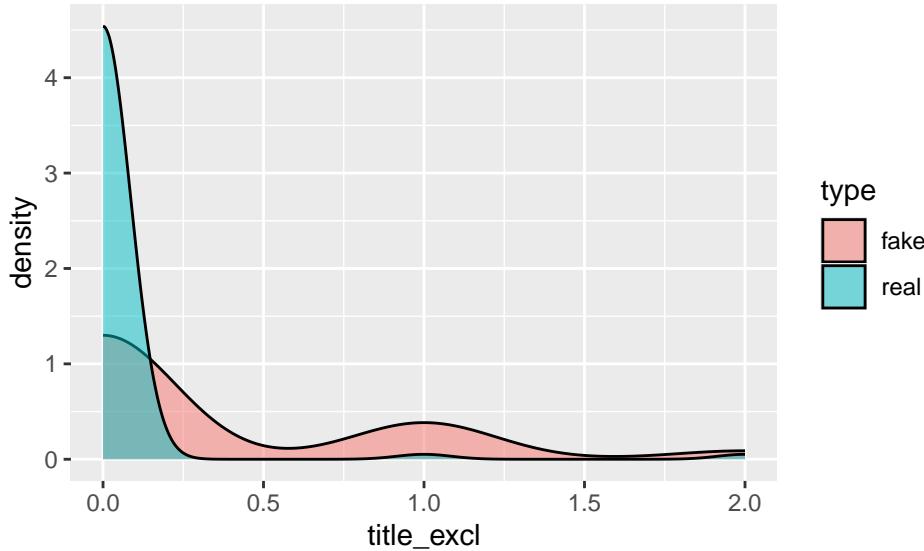
Kolejna zmienna to text\_caps:

```
ggplot(fake_news, aes(x = text_caps, fill = type)) +  
  geom_density(alpha = 0.5)
```



Wykresy są dosyć podobne, gęstości dosyć mocno się pokrywają. Patrząc na poprzednie wykresy dla innych zmiennych, i to ile zmiennych już mam w modelu, stwierdzam, że dodanie tej zmiennej nie będzie istotne. Następna zmienna to title\_excl:

```
ggplot(fake_news, aes(x = title_excl, fill = type)) +  
  geom_density(alpha = 0.5)
```



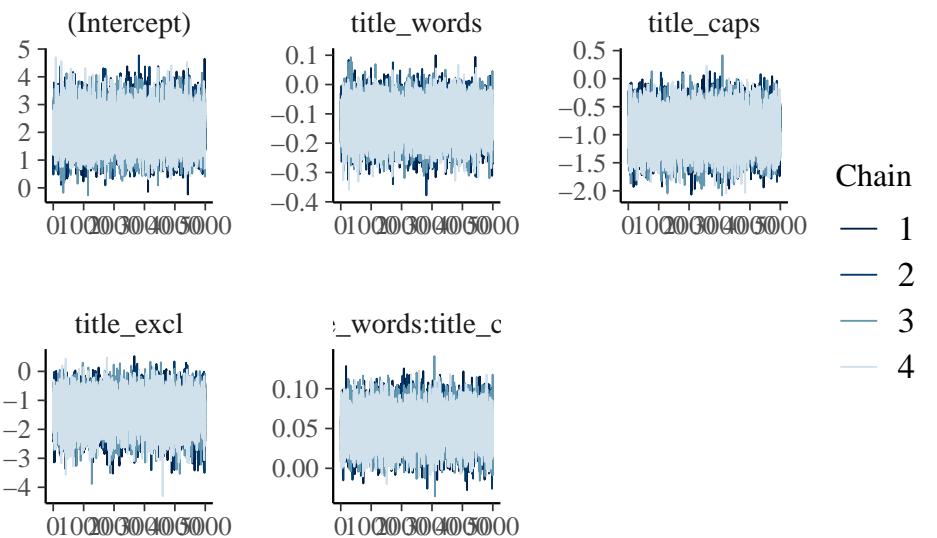
Ponownie, tak jak w przypadku title\_caps, widać bardzo duży wzrost gęstości dla prawdziwych artykułów przy title\_excl mających wartości bliskie zero. Dodatkowo, im większa wartość title\_excl, tym większa pewność, że artykuł jest nieprawdziwy. Zmienna wygląda na idealną do objaśniania zmiennej type, zatem dodaję ją do modelu:

```
model4 <- stan_glm(type ~ title_words+title_caps+title_words:title_caps+title_excl,
                     data = fake_news, family = binomial,
                     prior_intercept = normal(0.41, 0.63),
                     prior = normal(0.1, 0.5, autoscale = T),
                     chains = 4, iter = 5000*2, seed = 12345)
```

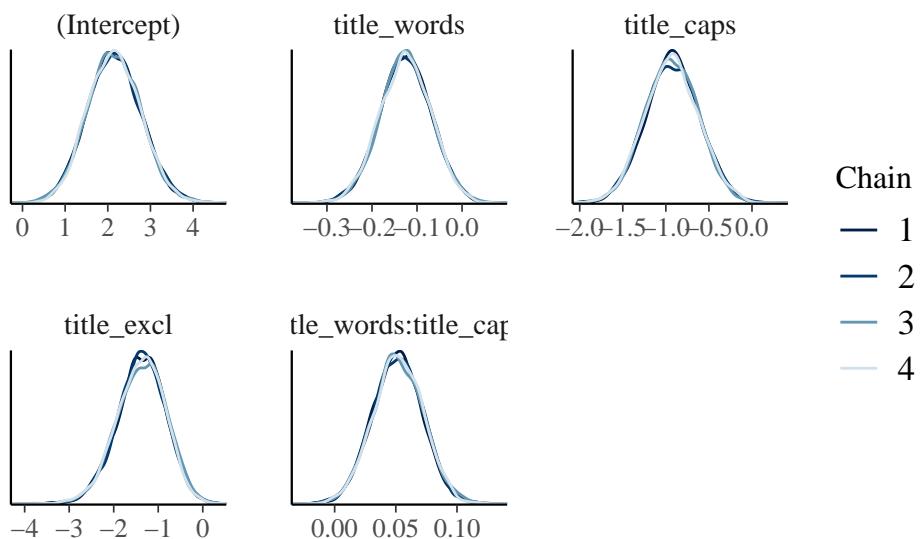
```
tidy(model4, effects = "fixed")
```

```
## # A tibble: 5 x 3
##   term                  estimate std.error
##   <chr>                <dbl>     <dbl>
## 1 (Intercept)           2.14      0.635
## 2 title_words          -0.126     0.0570
## 3 title_caps            -0.928     0.315
## 4 title_excl           -1.36      0.546
## 5 title_words:title_caps 0.0517    0.0205
```

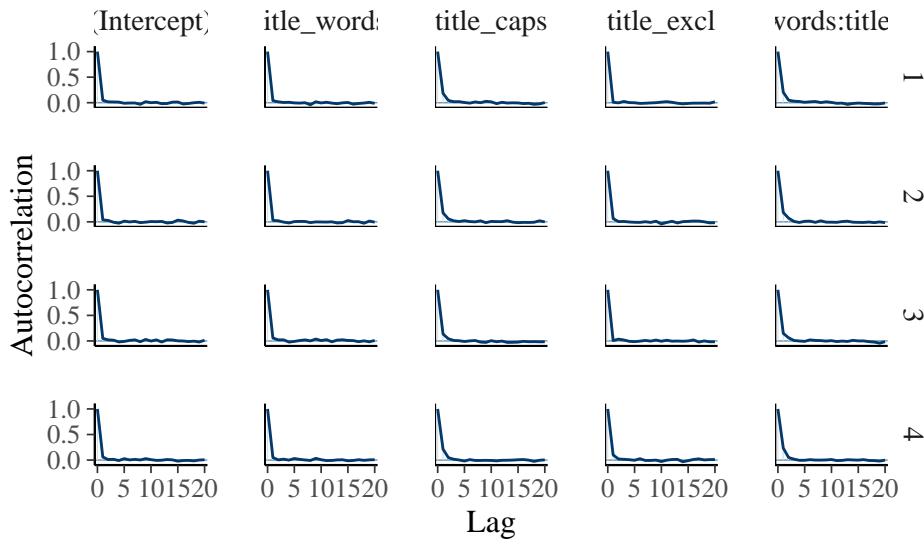
```
mcmc_trace(model4)
```



```
mcmc_dens_overlay(model4)
```

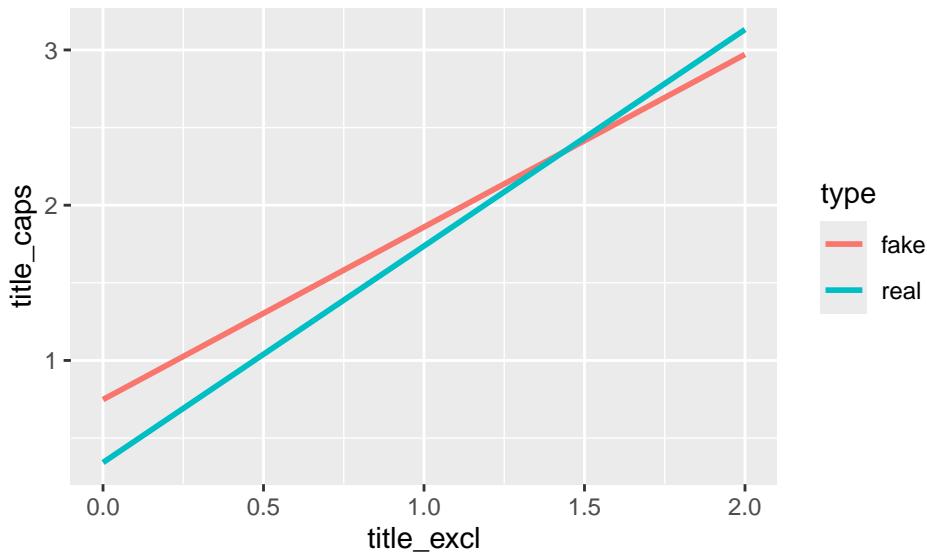


```
mcmc_acf(model4)
```



Czwarty model jak każdy poprzedni wygląda odpowiednio, mimo że zawiera więcej zmiennych. Ponownie, można domniemywać czy wartości zmiennej `title_excl` nie będą przypadkiem zależeć od pozostałych zmiennych dotyczących tytułów, i czy nie będzie między nimi istotnej interakcji. Ponownie sprawdzę to na wykresach:

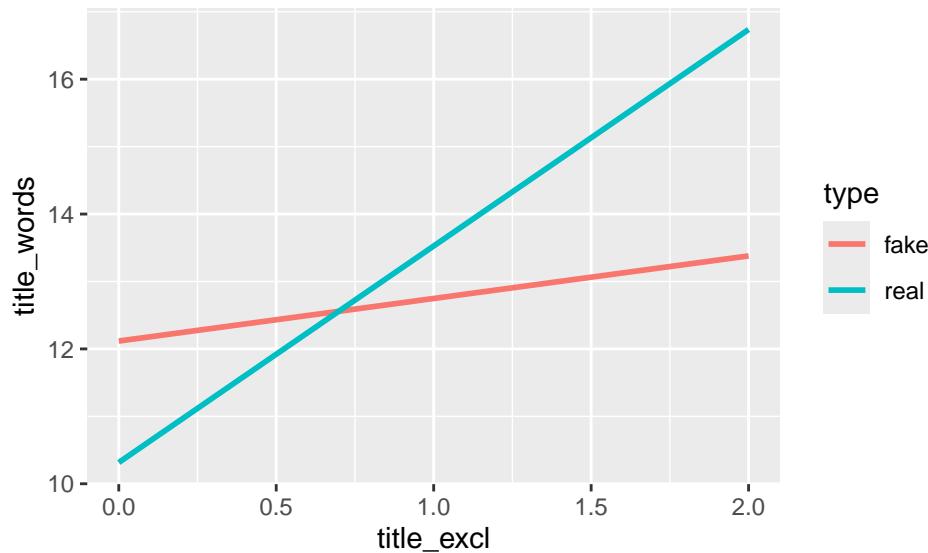
```
ggplot(fake_news, aes(y = title_caps, x = title_excl, color = type)) +
  geom_smooth(method = "lm", se = FALSE)
## `geom_smooth()` using formula = 'y ~ x'
```



O ile linie dla różnych typów artykułów nie są równoległe, to różnica nachyleń jest bardzo mała. Stąd, patrząc na to jak poprzednia interakcja mająca o wiele większe nachylenie miała bardzo mały wpływ na wartości w modelu, interakcje między `title_excl` i `title_caps` traktuję jako niestotną i nie dodaje jej do modelu.

```
ggplot(fake_news, aes(y = title_words, x = title_excl, color = type)) +  
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

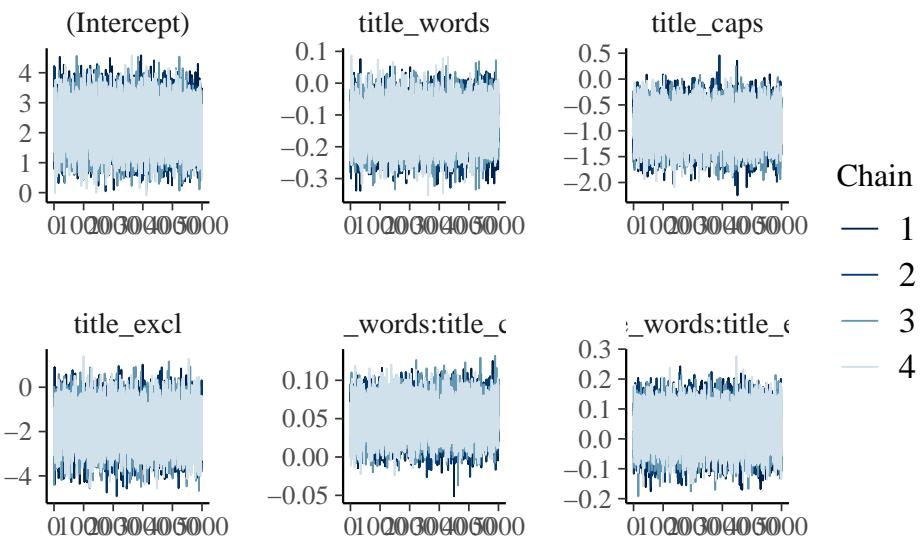


Dla zmiennych title\_excl i title\_words interakcja jest o wiele bardziej widoczna. Zatem ponownie dodam ją do modelu, i sprawdze czy i jak duży wpływ ma na zmienną objaśnianą:

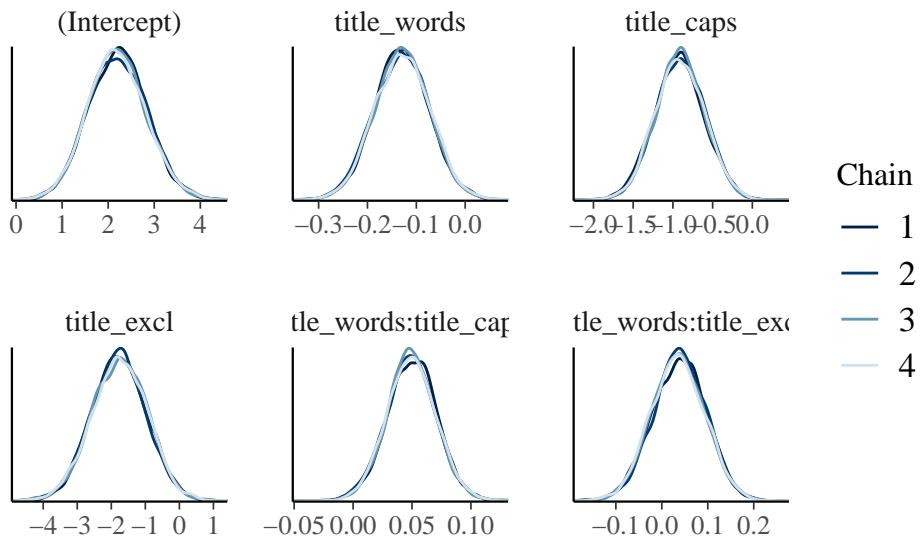
```
model5 <- stan_glm(type ~ title_words+title_caps+title_words:title_caps+title_excl+title_words:title_excl,  
                     data = fake_news, family = binomial,  
                     prior_intercept = normal(0.41, 0.63),  
                     prior = normal(0.1, 0.5, autoscale = T),  
                     chains = 4, iter = 5000*2, seed = 12345)  
  
tidy(model5, effects = "fixed")
```

```
## # A tibble: 6 x 3  
##   term                  estimate std.error  
##   <chr>                 <dbl>     <dbl>  
## 1 (Intercept)            2.19      0.640  
## 2 title_words            -0.130     0.0574  
## 3 title_caps             -0.912     0.312  
## 4 title_excl             -1.77      0.816  
## 5 title_words:title_caps 0.0495    0.0206  
## 6 title_words:title_excl 0.0370    0.0580
```

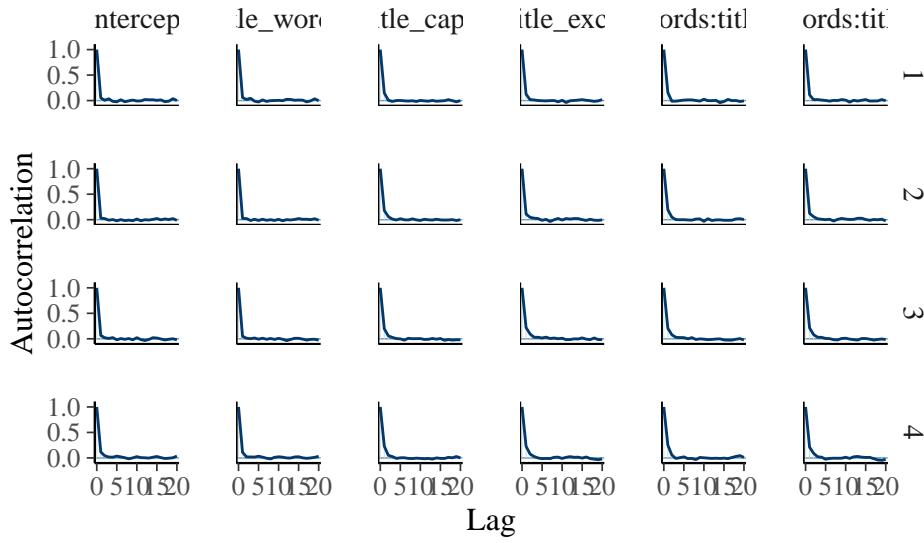
```
mcmc_trace(model5)
```



```
mcmc_dens_overlay(model5)
```



```
mcmc_acf(model5)
```



Model jest dobrze zbudowany. Sprawdzę wartości  $\beta$ :

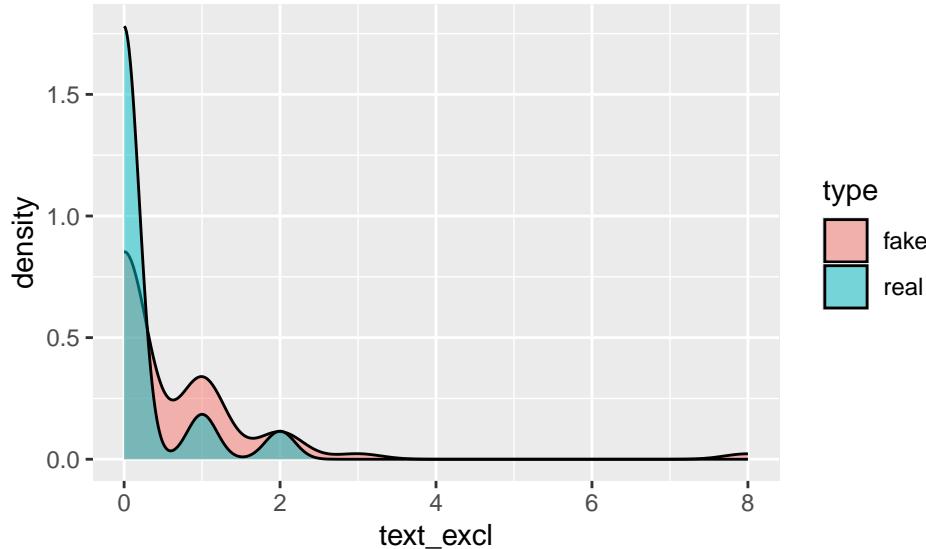
```
posterior_interval(model5, prob = 0.9)
```

```
##                                     5%          95%
## (Intercept)      1.17623171  3.25468784
## title_words     -0.22322375 -0.03719089
## title_caps       -1.42325736 -0.39552264
## title_excl      -3.11169930 -0.45633647
## title_words:title_caps  0.01498227  0.08383036
## title_words:title_excl -0.05864473  0.12969879
```

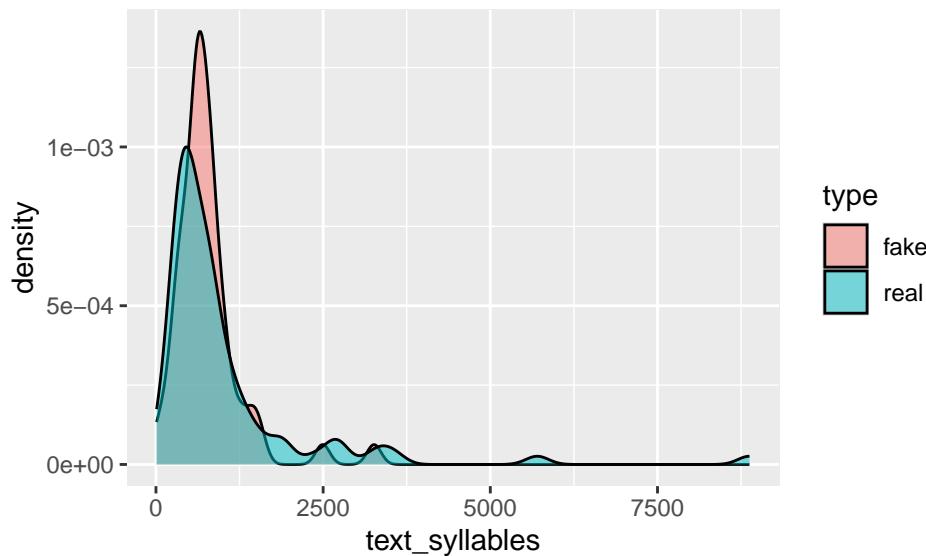
Wartości współczynnika  $\beta$  dla interakcji `title_words:title_excl` są o wiele bardziej skrajne niż dla interakcji `title_words:title_caps`, co może sugerować, że ta interakcja będzie miała większy wpływ. Jednak problemem tutaj jest fakt, że w przedziale możliwych wartości  $\beta$  jest liczba 0, która oznacza, że interakcja nie będzie miała żadnego wpływu na szansę otrzymania 1 w zmiennej objaśnianej. Jest to pierwszy taki przypadek dla moich współczynników, i dodając do tego fakt, że te skrajne wartości  $\beta$  są dosyć bliskie zera, to nie będę dodawał tej interakcji do modelu.

Kolejne trzy zmienne do sprawdzenia to `text_excl`, `text_words` i `text_syllables`:

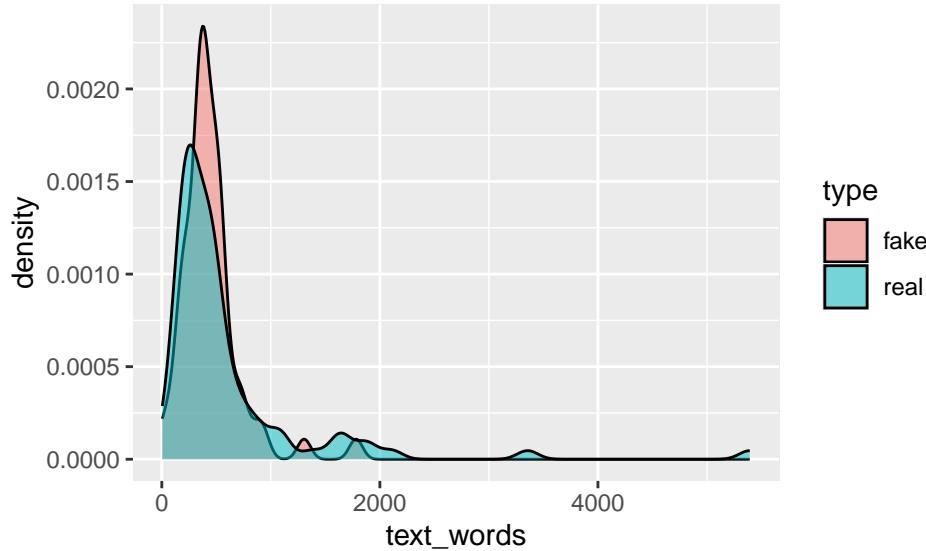
```
ggplot(fake_news, aes(x = text_excl, fill = type)) +
  geom_density(alpha = 0.5)
```



```
ggplot(fake_news, aes(x = text_syllables, fill = type)) +  
  geom_density(alpha = 0.5)
```



```
ggplot(fake_news, aes(x = text_words, fill = type)) +  
  geom_density(alpha = 0.5)
```



W każdym z przypadków wykresy dla wartości fake i real dosyć mocno nachodzą na siebie, zatem te zmienne nie będą dobrze objaśniały zmiennej type, zwłaszcza że w modelach mam już wybrane kilka innych, o wiele widoczniej skorelowanych zmiennych. Więc nie dodaje do modelu żadnej nowej zmiennej.

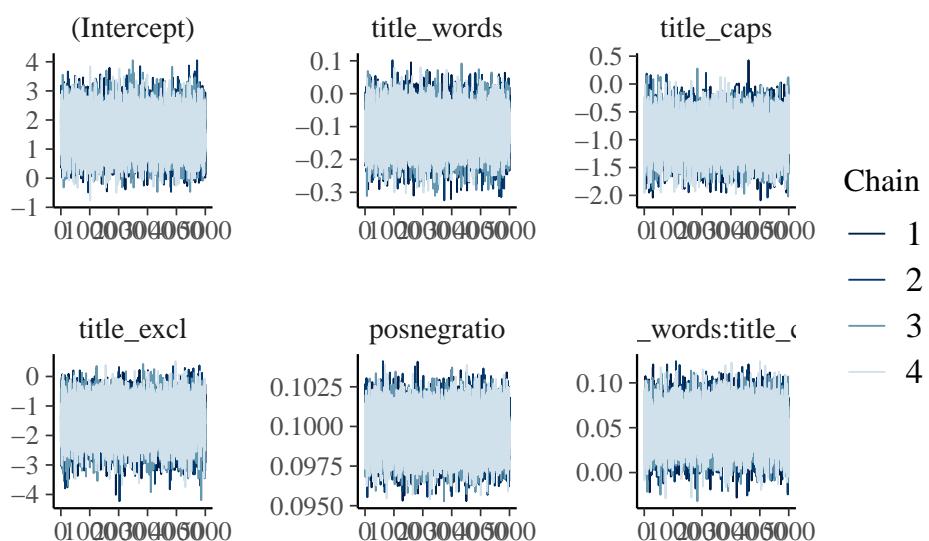
Ostatnią zmienną, którą samemu dodałem do datasetu jest wcześniej wspomniane posnegratio. Zakładam, że będzie ona ważną zmienną objaśniającą, zatem od razu dodaję ją do modelu:

```
model6 <- stan_glm(type ~ title_words+title_caps+title_words:title_caps+title_excl+posnegratio,
                      data = fake_news, family = binomial,
                      prior_intercept = normal(0.41, 0.63),
                      prior = normal(0.1, 0.5, autoscale = T),
                      chains = 4, iter = 5000*2, seed = 12345)
```

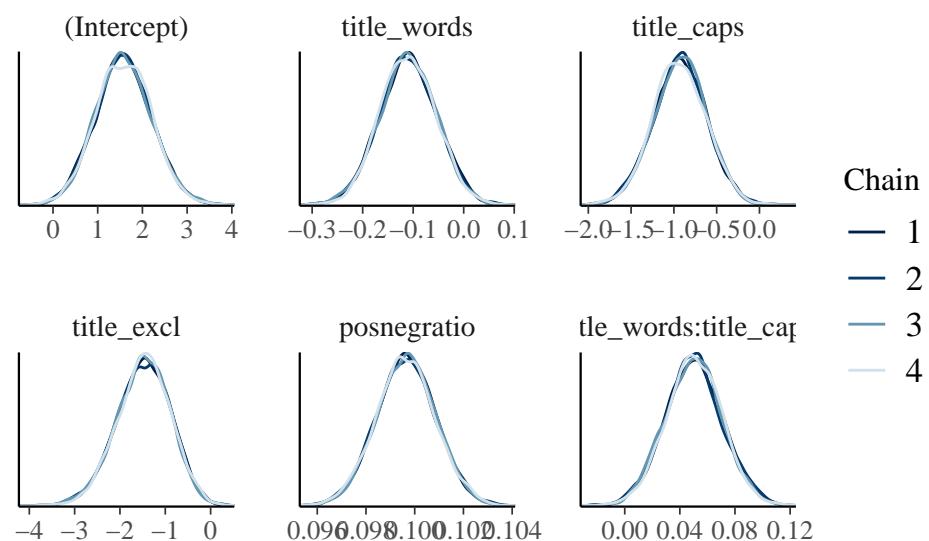
```
tidy(model6, effects = "fixed")
```

```
## # A tibble: 6 x 3
##   term                  estimate std.error
##   <chr>                 <dbl>     <dbl>
## 1 (Intercept)            1.57      0.624
## 2 title_words           -0.110     0.0565
## 3 title_caps             -0.927     0.311
## 4 title_excl            -1.47      0.595
## 5 posnegratio            0.0996    0.00121
## 6 title_words:title_caps 0.0501    0.0204
```

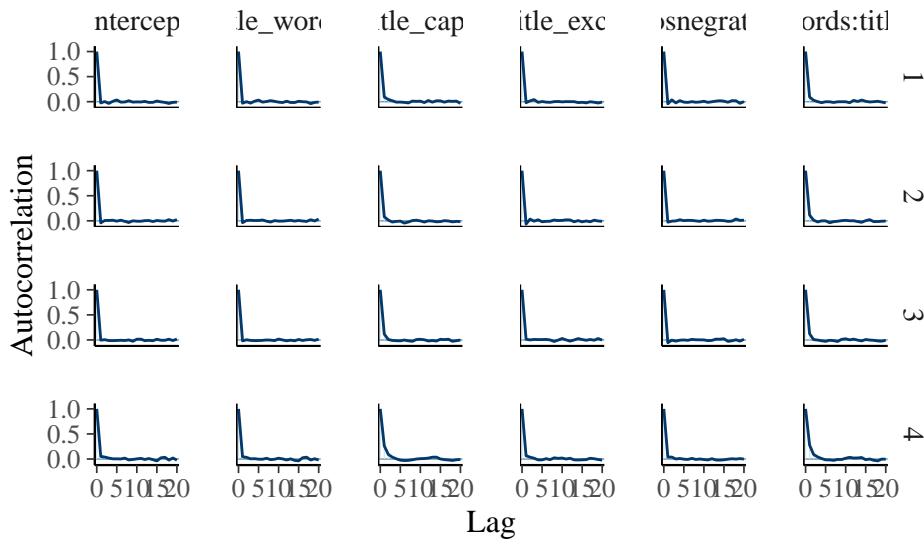
```
mcmc_trace(model6)
```



```
mcmc_dens_overlay(model6)
```



```
mcmc_acf(model6)
```



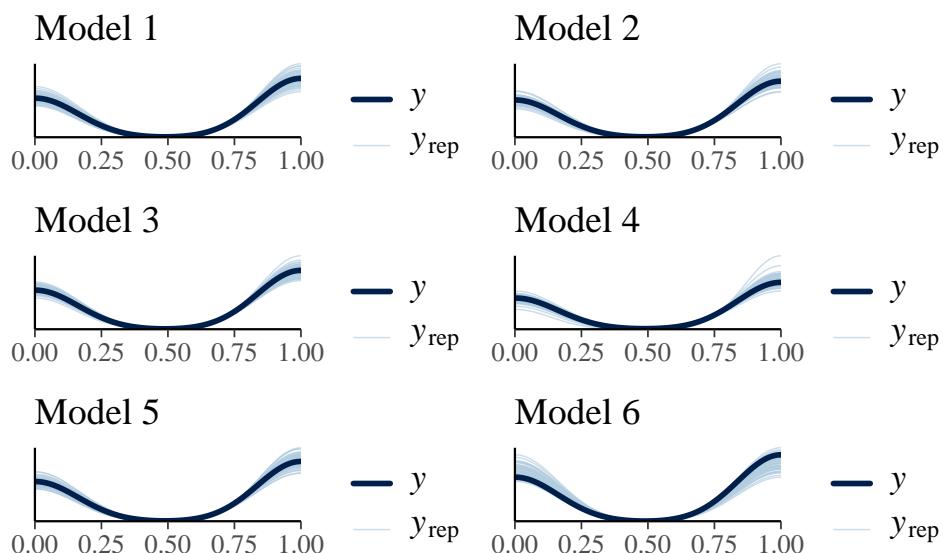
Model wygląda na dobrze zbudowany.

## PORÓWNANIE MODELI

Sprawdzę teraz, czy faktycznie ostatni model zbudowany ze wszystkich zmiennych które wycenilem na wartości dodania do modelu będzie najlepszy. Porównam wszystkie modele do siebie, wpierw wyświetlając dopasowania każdego modelu do prawdziwych wartości:

```
g1<-pp_check(model1)+ggtitle("Model 1")
g2<-pp_check(model2)+ggtitle("Model 2")
g3<-pp_check(model3)+ggtitle("Model 3")
g4<-pp_check(model4)+ggtitle("Model 4")
g5<-pp_check(model5)+ggtitle("Model 5")
g6<-pp_check(model6)+ggtitle("Model 6")

grid.arrange(g1, g2, g3, g4, g5, g6)
```



Co ciekawe każdy model wykazuje dosyć dobre dopasowanie do rzeczywistych danych, i nie ma między dopasowaniami widocznych różnic. Zatem do głównej oceny jakości dopasowania modelu posłuży mi funkcja loo porównująca jak model wyznacza wartości:

```
loo_model1 <- loo(model1)
loo_model2 <- loo(model2)
loo_model3 <- loo(model3)
loo_model4 <- loo(model4)
loo_model5 <- loo(model5)
loo_model6 <- loo(model6)
loo_compare(loo_model1, loo_model2, loo_model3, loo_model4, loo_model5, loo_model6)

##          elpd_diff se_diff
## model6    0.0      0.0
## model4   -0.5     1.8
## model5   -0.7     1.8
## model3  -2.7     3.6
## model2 -3.9     3.3
## model1 -7.6     4.6
```

Interesuje mnie by miara elpd\_diff, która ocenia dopasowanie modelu do danych, była jak największa. Według tej miary, faktycznie najlepszym modelem jest model nr 6, jednak patrząc na różnice między wartościami, oraz wartości se\_diff, to jakość wszystkich modeli jest porównywalnie dobra.

## KOŃCOWY MODEL

Zatem na podstawie powyższych porównań najlepszym modelem jest model 6. Przyjrzę się mu teraz dokładniej, zaczynając od jego rozkładu apriori:

```
model <- stan_glm(type ~ title_words+title_caps+title_words:title_caps+title_excl+posnegratio,
                     data = fake_news, family = binomial,
                     prior_intercept = normal(0.41, 63),
                     prior = normal(0.1, 0.5, autoscale = T),
                     chains = 4, iter = 5000*2, seed = 12345,
                     prior_PD = TRUE)

prior_summary(model)$prior

## $dist
## [1] "normal"
##
## $location
## [1] 0.1 0.1 0.1 0.1 0.1
##
## $scale
## [1] 0.5 0.5 0.5 0.5 0.5
##
## $adjusted_scale
## [1] 0.141132189 0.466738417 1.179537944 0.001225141 0.029521019
##
## $df
## NULL
```

Współczynniki  $\beta$  dla poprawionego rozkładu apriori z 0.5 zostały zmienione na odpowiednio 0.1411, 0.4667, 1.1795, 0.0012 i 0.0295.  $\beta_0$  wyliczona przeze mnie wcześniej wynosi 0.41.

Następnie możemy zmienić model z apriori na aposteriori i odczytać wartości  $\beta$  dla końcowego modelu predykcyjnego.

```
model <- update(model, prior_PD = FALSE)

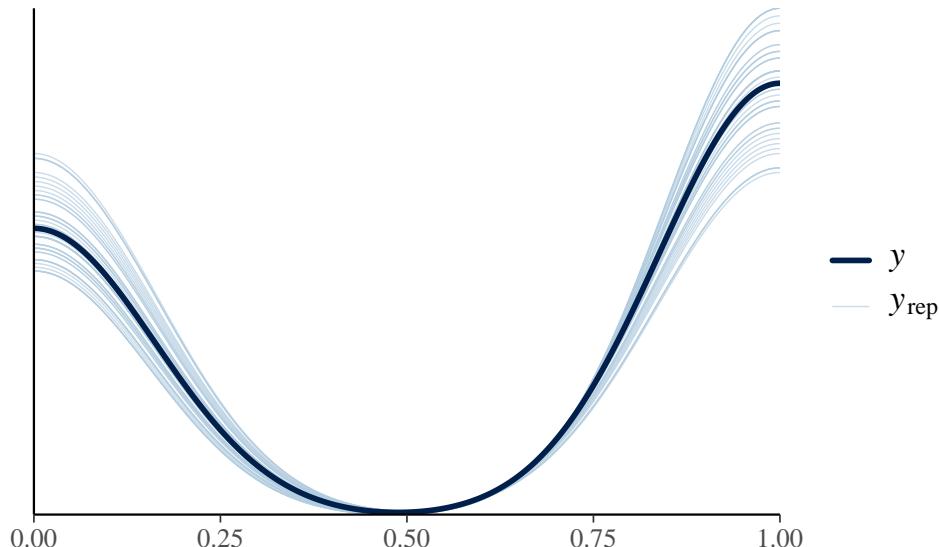
tidy(model, effects = "fixed", conf.int = TRUE, conf.level = 0.90) %>%
  select(-std.error)

## # A tibble: 6 x 4
##   term          estimate conf.low conf.high
##   <chr>        <dbl>    <dbl>     <dbl>
## 1 (Intercept)  1.87     0.853     2.95
## 2 title_words -0.116    -0.211    -0.0231
## 3 title_caps  -0.932    -1.45     -0.419
## 4 title_excl  -1.31     -2.27     -0.463
## 5 posnegratio  0.100     0.0980    0.102
## 6 title_words:title_caps 0.0509   0.0172    0.0852
```

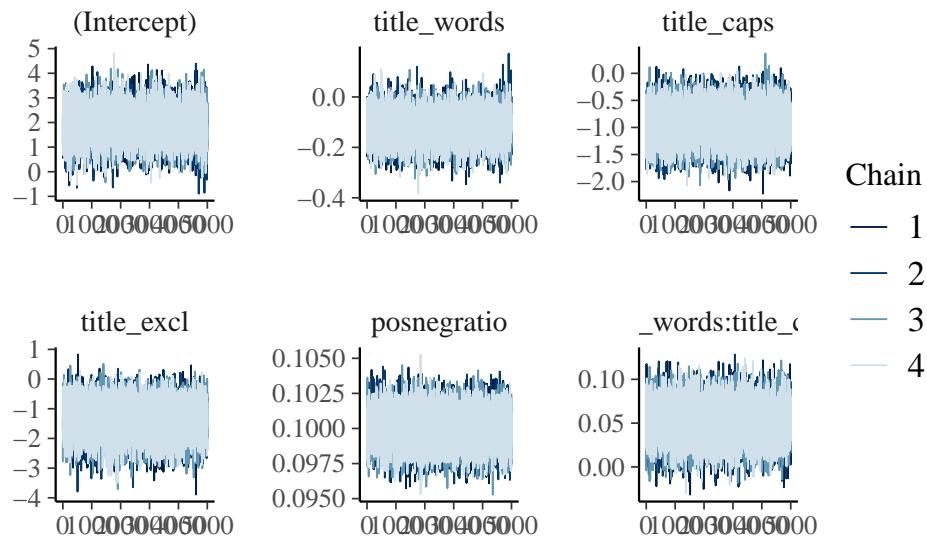
Można zauważyc, że wartości  $\beta$  dla końcowego modelu występują zarówno dodatnie jak i ujemne. Jeżeli parametr  $\beta_n$  jest dodatni, to przy interpretacji  $e^{\beta_n}$  oznacza, że wzrost danego parametru  $X_n$  w moim modelu o 1 zwiększa szanse, że artykuł będzie prawdziwy. Analogiczne ujemna wartość  $\beta_n$  oznacza, że szanse na prawdziwość artykułu maleją gdy wartość zmiennej objaśniającej zwiększa się. Zgadza się to z poprzednimi wykresami, ilość słów, wykrzykników czy wielkich liter w tytule zmniejsza jego szanse na bycie prawdziwym, zatem dla nich parametry  $\beta_n$  są ujemne. Odwrotnie jest w przypadku posnegratio, im większa przewaga pozytywnych słów nad tymi negatywnymi, tym zwiększa się szansa na prawdziwość artykułu, stąd  $\beta_4 > 0$ . Wartość  $\beta_0$  dla rozkładu aposteriori jest większa od 0, zatem w sytuacji gdy wyzeruje wszystkie zmienne  $X_n$  prawdopodobieństwo że artykuł jest prawdziwy będzie większe niż 50%.

Jeszcze raz sprawdzę jakość dopasowania modelu.

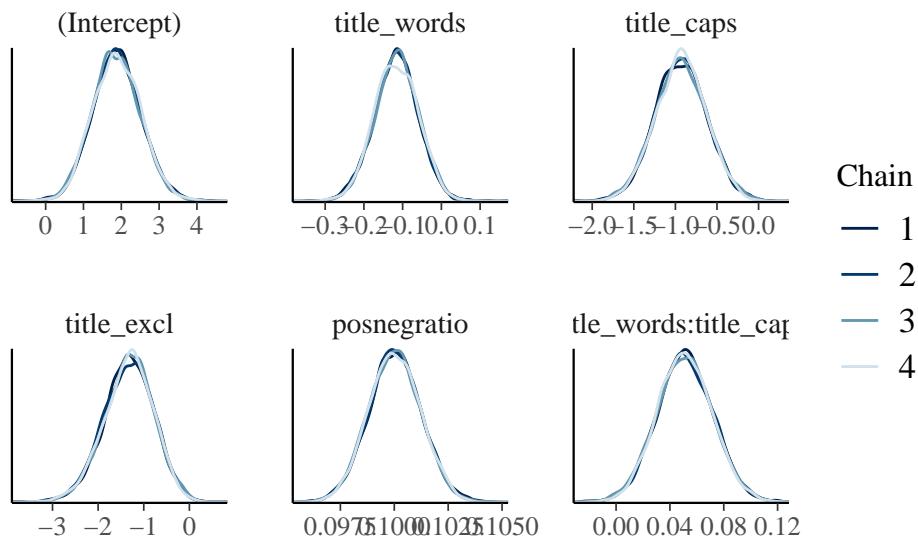
```
pp_check(model)
```



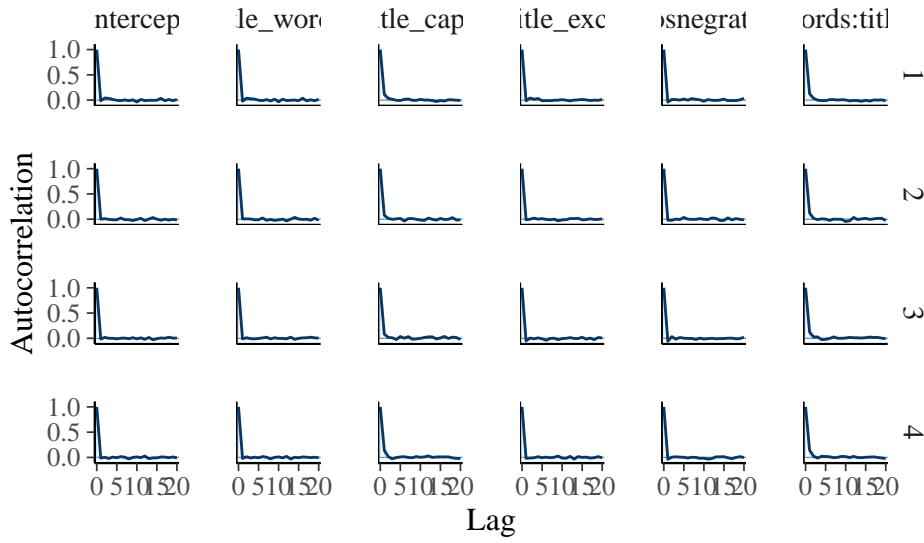
```
mcmc_trace(model)
```



```
mcmc_dens_overlay(model)
```



```
mcmc_acf(model)
```



Mój model dobrze dopasowuje się do rozkładu. Wykresy łańcuchów pokazują szum i losowość w symulacji dla każdego parametru. Wykresy gęstości indywidualnych parametrów się pokrywają, co też jest porządnym wynikiem. Autokorelacja dla każdego z 4 łańcuchów i dla każdej z 5 zmiennych zeruje się, co też pozwala twierdzić, że symulacja była wiarygodna. Ostatnią diagnostyką modelu będzie badanie współczynników neff\_ratio i rhat:

```
neff_ratio(model)
```

```
##          (Intercept)      title_words      title_caps
## 0.98035           0.98420           0.78965
## title_excl      posnegratio title_words:title_caps
## 1.00260           1.04065           0.76510
```

```
rhat(model)
```

```
##          (Intercept)      title_words      title_caps
## 1.0000822       1.0000372       1.0001880
## title_excl      posnegratio title_words:title_caps
## 0.9999590       0.9999782       1.0000819
```

Dla obu współczynników wartości są w granicach 0.7-1, z większością oscylującą wokół 1, co oznacza dobrą symulację w modelu. Zatem końcowo mogę stwierdzić, że końcowy model jest dobrym modelem dla zmiennej binarnej type.

## WNIOSKI

1. Najlepszym modelem okazał się model z 5 zmiennymi title\_words, title\_caps, title\_excl, posnegratio i interakcją title\_words:title\_caps.
2. Największy wpływ na prawdziwość artykułu miała ilość słów pisanych wielkimi literami w tytule.
3. Żaden z modeli nie miał problemu z symulacją MC oraz każdy z nich dobrze dopasowywał się do rzeczywistych danych.
4. Z wyników można stwierdzić, że prawdziwość artykułu o wiele bardziej zależy od tytułu, niż samego tekstu.
5. Gdy wyzeruje wartości wszystkim zmiennym, to prawdopodobieństwo że artykuł będzie prawdziwy będzie bardzo duże (ok. 85%).