Modelos para dados positivos

Universidade de Brasília
Instituto de Ciências Exatas - Departamento de Estatística
Modelos Lineares Generalizados

Alunos responsáveis:
Álvaro Jeronimo da Silva Kothe - 17/0004694
Marcos Augusto Daza Barbosa - 17/0017834
Mathews de Noronha Silveira Lisboa - 17/0042324
William Edward Rappel de Amorim - 17/0047385

Professor:
Guilherme Souza Rodrigues

29 de Outubro de 2019

Construção do modelo

Para a construção do modelo, primeiramente foram modificados os formatos de algumas variáveis, como tirar a ',' de milhar, o '$', separou-se as datas recebidas em ano, mês e dia da semana. Nas colunas em que havia '%', considerou-se os valores como numéricos, compreendidos na escala de zero a um. Valores faltantes em variáveis categóricas foram substituídos por uma nova categoria chamada de '*Unknown*'. A coluna *Amenities* foi separada em várias variáveis *dummys*, identificando se cada item identificado em *amenities* estava presente na moradia ou não.

Em seguida, foi realizada uma seleção de variáveis, com o objetivo de obter um modelo prévio, utilizando o método *forward*.

Após esta etapa, preencheu-se valores faltantes utilizando a informação do *host*. Caso não desse para ser estimado, foi imputada a média. Na maior parte, foi imputado considerando o seguinte critério:

- Se a variável for limitada inferiormente e superiormente, foi utilizada a regressão *beta*.
- Se a variável for de contagem, foi utilizada a regressão de Poisson.

As colunas *host_response_rate* e as que começavam com *reviews_scores* foram estimadas com regressão beta e transformadas para a escala original. As colunas *bathrooms*, *bedrooms*, *beds*, *reviews_per_month*, *days_first_review*, *days_last_review* foram estimadas utilizando regressão Poisson.

Foi criada uma *dummy* para cada coluna estimada identificando se teve que imputar naquela coluna ou não, e, na parte de seleção de variáveis, será verificado se ela é necessária.

Após isso, foi realizada uma seleção de variáveis utilizando o método *Stepwise*, com uma pequena amostra escolhida aleatoriamente do banco de dados com valores imputados. Em seguida, foram feitas pequenas alterações para chegar no modelo abaixo, com AIC com o valor de 7.486.994.

$$
\begin{aligned}
\ln(\mu_{price}) = {} & \beta_0 + \beta_{host\_response\_timewithinanhour} X_1 + \beta_{host\_response\_timewithinafewhours} X_2 \\
& + \beta_{host\_response\_timewithinaday} X_3 + \beta_{host\_response\_timeafewdaysormore} X_4 \\
& + \beta_{host\_is\_superhost} X_5 + \beta_{host\_total\_listings\_count} X_6 \\
& + \beta_{host\_has\_profile\_pic} X_7 + \beta_{neighbourhood\_group\_cleansedBeaconHill} X_8 \\
& + \beta_{neighbourhood\_group\_cleansedCapitolHill} X_9 \\
& + \beta_{neighbourhood\_group\_cleansedCascade} X_{10} \\
& + \beta_{neighbourhood\_group\_cleansedCentralArea} X_{11} \\
& + \beta_{neighbourhood\_group\_cleansedDelridge} X_{12} \\
& + \beta_{neighbourhood\_group\_cleansedDowntown} X_{13} \\
& + \beta_{neighbourhood\_group\_cleansedInterbay} X_{14} \\
& + \beta_{neighbourhood\_group\_cleansedLakeCity} X_{15} \\
& + \beta_{neighbourhood\_group\_cleansedMagnolia} X_{16} \\
& + \beta_{neighbourhood\_group\_cleansedNorthgate} X_{17} \\
& + \beta_{neighbourhood\_group\_cleansedOtherneighborhoods} X_{18} \\
& + \beta_{neighbourhood\_group\_cleansedQueenAnne} X_{19} \\
& + \beta_{neighbourhood\_group\_cleansedRainierValley} X_{20} \\
& + \beta_{neighbourhood\_group\_cleansedSewardPark} X_{21} \\
& + \beta_{neighbourhood\_group\_cleansedUniversityDistrict} X_{22} \\
& + \beta_{neighbourhood\_group\_cleansedWestSeattle} X_{23} + \beta_{latitude} X_{24} \\
& + \beta_{is\_location\_exact} X_{25} + \beta_{property\_typeBed\&Breakfast} X_{26} \\
& + \beta_{property\_typeBoat} X_{27} + \beta_{property\_typeBungalow} X_{28} + \beta_{property\_typeCabin} X_{29} \\
& + \beta_{property\_typeCamper/RV} X_{30} + \beta_{property\_typeChalet} X_{31} \\
& + \beta_{property\_typeCondominium} X_{32} + \beta_{property\_typeDorm} X_{33} \\
& + \beta_{property\_typeHouse} X_{34} + \beta_{property\_typeLoft} X_{35} + \beta_{property\_typeOther} X_{36} \\
& + \beta_{property\_typeTent} X_{37} + \beta_{property\_typeTownhouse} X_{38} \\
& + \beta_{property\_typeTreehouse} X_{39} + \beta_{property\_typeUnknown} X_{40} + \beta_{property\_typeYurt} X_{41} \\
& + \beta_{room\_typePrivateroom} X_{42} + \beta_{room\_typeSharedroom} X_{43} + \beta_{accommodates} X_{44} \\
& + \beta_{bathrooms} X_{45} + \beta_{bedrooms} X_{46} + \beta_{security\_deposit} X_{47} + \beta_{cleaning\_fee} X_{48} \\
& + \beta_{guests\_included} X_{49} + \beta_{number\_of\_reviews} X_{50} + \beta_{review\_scores\_rating} X_{51} \\
& + \beta_{review\_scores\_accuracy} X_{52} + \beta_{review\_scores\_location} X_{53} + \beta_{review\_scores\_value} X_{54} \\
& + \beta_{cancellation\_policymoderate} X_{55} + \beta_{cancellation\_policystrict} X_{56} \\
& + \beta_{calculated\_host\_listings\_count} X_{57} + \beta_{reviews\_per\_month} X_{58} + \beta_{monthfev} X_{59} \\
& + \beta_{monthmar} X_{60} + \beta_{monthabr} X_{61} + \beta_{monthmai} X_{62} + \beta_{monthjun} X_{63} \\
& + \beta_{monthjul} X_{64} + \beta_{monthago} X_{65} + \beta_{monthset} X_{66} + \beta_{monthout} X_{67} \\
& + \beta_{monthnov} X_{68} + \beta_{monthdez} X_{69} + \beta_{week\_dayseg} X_{70} + \beta_{week\_dayter} X_{71} \\
& + \beta_{week\_dayqua} X_{72} + \beta_{week\_dayqui} X_{73} + \beta_{week\_daysex} X_{74} + \beta_{week\_daysáb} X_{75} \\
& + \beta_{days\_first\_review} X_{76} + \beta_{days\_last\_review} X_{77} + \beta_{Internet} X_{78} + \beta_{TV} X_{79} \\
& + \beta_{WirelessInternet} X_{80} + \beta_{CableTV} X_{81} + \beta_{ElevatorinBuilding} X_{82} + \beta_{longitude} X_{83}
\end{aligned}
$$

a) Ajuste um MLG para estimar a probabilidade de um dado apartamento (isso é, com um nível fixo das covariáveis) ser anunciado por um valor superior a $200. Qual é o erro quadrático médio do modelo no conjunto de teste?

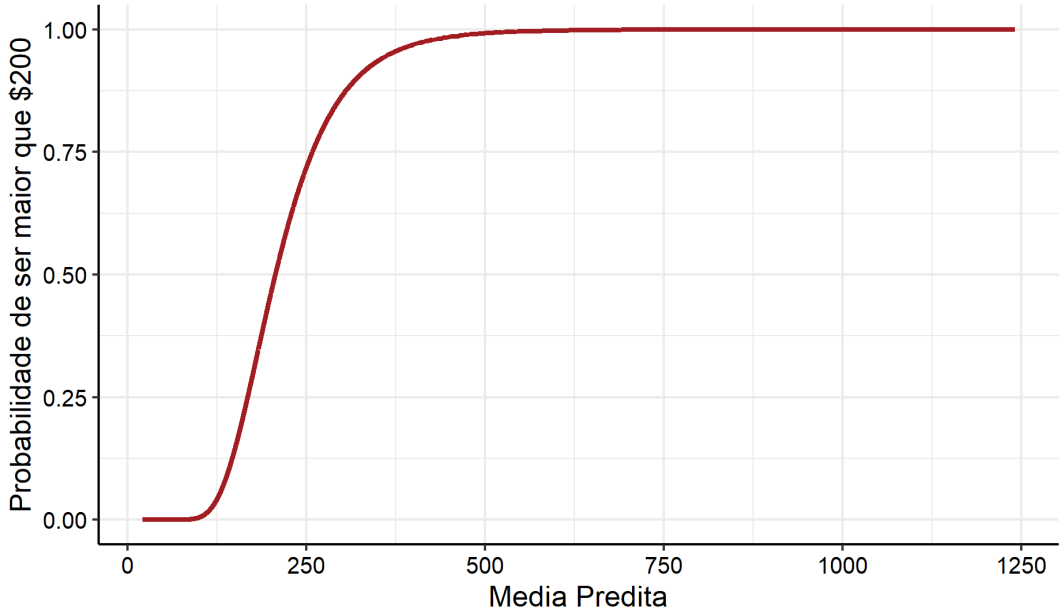
Para estimar a probabilidade de um dado apartamento ser anunciado por um valor superior a $200, será utilizada como distribuição da variável resposta a distribuição Gama, com função de ligação igual ao logaritmo natural. Utilizando a parametrização dos slides, tem-se que:

- $\alpha = \phi$
- $\beta = \frac{\phi}{\mu}$

Dessa forma, para estimar essas probabilidades, basta utilizar o valor estimado da média e da precisão (ϕ) para obter a distribuição estimada para cada observação e, para cada uma delas, calcular a cauda à direita, a partir do valor de preço igual a $200.

```
phi <- 1/MASS::gamma.dispersion(final) # estimativa da precisao
alpha <- phi
beta <- phi/predict(final, type = "response")
prob.200 <- 1 - pgamma(200, shape = alpha, rate = beta) # calculo
    das probabilidades
mu.hat <- predict(final, type = "response") # valores preditos
amostra <- sample(1:length(mu.hat), 10000) # amostra de pontos para
     o grafico
df10000 <- data.frame(mu.hat = mu.hat[amostra], prob.200 = prob
    .200[amostra])
ggplot(df10000, aes(x=mu.hat, y=prob.200)) +
  geom_line(colour="#A11D21", size=1) +
  labs(x="Media Predita", y="Probabilidade de ser maior que $200")
    +
  theme_bw() +
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        panel.border = element_blank(),
        axis.line = element_line(colour = "black"))
ggsave("prob_200.png", width = 158, height = 93, units = "mm")
```

Figura 1: Média Predita x Probabilidade Estimada do Preço Superar $200



Abaixo, segue um quadro contendo os erros quadráticos médios do modelo final avaliado no banco de dados de treinamento e no de teste.

Banco de Dados	**Erro Quadrático Médio**
Treinamento	62,2379
Teste	62,4896

Por meio do quadro acima, percebe-se que os erros quadráticos médios de ambos os bancos são muito próximos, o que indica uma boa capacidade de previsão do preço, a partir das covariáveis mantidas no modelo.

b) Considerando fixas as demais covariáveis, qual é o ponto ou região da cidade com maiores valores anunciados?

Considerando fixas as demais covariáveis e levando em consideração o modelo ajustado, percebe-se que a latitude e a longitude possuem efeito β negativo; portanto, o incremento multiplicativo é menor que 1. Logo, para maximizar o preço, deve-se minimizar a longitude e a latitude. Observando o banco de dados, encontra-se que os menores valores para latitude e longitude são, respectivamente, 47,50728 e -122,41722. Sendo assim, a região que possui os maiores valores anunciados é Cascade.

Também é possível verificar isso observando o incremento multiplicativo β da variável "neighbourhood_group_cleansed" para o fator Cascade, pois possui o maior valor de β. Sendo assim, pode-se dizer que Cascade é a região com maior preço médio anunciado.

c) Descreva a influência da data no preço dos alugueis

É esperado que o mercado de alugueis de imóveis, tal qual é o *Airbnb*, seja fortemente influenciado pela data em que se planeja alugar o imóvel, visto que é uma procura sazonal.

Variável	**Coeficiente**
fev	0,00989
mar	0,03563
abr	0,05623
mai	0,08402
jun	0,12360
jul	0,13380
ago	0,12560
set	0,08305
out	0,04348
nov	0,03161
dez	0,03378

Ao observar os meses, pode-se notar que, em relação a janeiro, os meses com maiores valores de preços anunciados são Junho, Julho e Agosto, o que é coerente com o período de férias escolares de Verão nos EUA. Dessa forma, espera-se que nesses meses o preço seja maior.

Variável	**Coeficiente**
Segunda	-0,003849
Terça	-0,006828
Quarta	-0,007522
Quinta	-0,000819
Sexta	0,045930
Sábado	0,048040

Ao observar os dias da semana, pode-se notar que, comparado a domingo, sexta e sábado são os dias em que os preços são maiores, enquanto que, de segunda a quinta, os preços são menores. Este fato condiz com a ideia de que imóveis são mais procurados em finais de semana.

d) Quais são, em ordem decrescente, os cinco principais fatores na determinação do preço das unidades ofertadas?

Os cinco principais fatores, em ordem decrescente, na determinação do preço das unidades são longitude, latitude, "room_typeShared room", "property_typeTent" e "property_typeBoat". Essa constatação foi baseada nos maiores valores absolutos dos coeficientes que estão no quadro abaixo.

Variável	**Coeficiente**
latitude	-1,2722705
longitude	-0,9326195
room_typeShared room	-0,8230671
property_typeDorm	-0,7131058
property_typeTent	-0,6134797

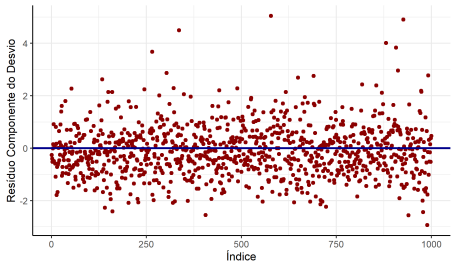
e) De que modo o preço é impactado pelo aumento das notas dos reviews?

Considerando o modelo ajustado, pode-se observar que o valor β para a variável "reviews_score_rating" indica um efeito positivo de aproximadamente $6,7 \times 10^{-3}$. Sendo assim, espera-se que tenha um incremento multiplicativo de $e^{6,7 \times 10^{-3}}$ no valor do preço final. Ou seja, o efeito das reviews não é tão impactante no preço anunciado do imóvel quanto se espera de um mecanismo de avaliação do imóvel.

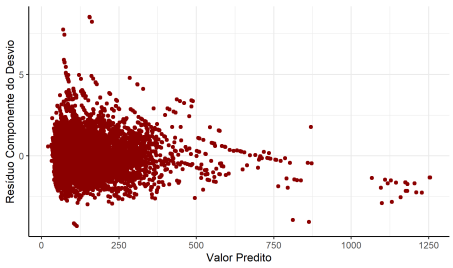
Talvez, a falta de um efeito mais significativo seja resultante do número de valores não observados que as variáveis de "reviews" têm. Foram mais de 140 mil valores faltantes.

f) Enumere as suposições feitas pelo modelo, comente a razoabilidade de cada uma delas e indique formas de melhorar o modelo (não é preciso implementar as estratégias elencadas).

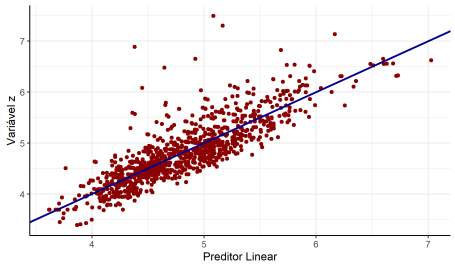
- Independência das observações
- A distribuição da variável "price" é uma Gama
- A forma do preditor linear é adequada
- A função de ligação é adequada



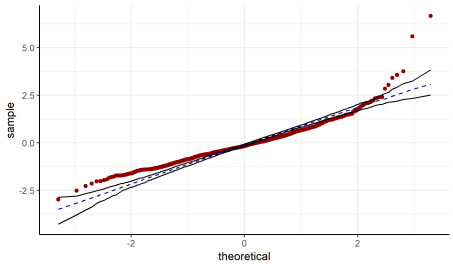
(a) RCD vs. Índice



(b) RCD vs. Valores Preditos



(c) Variável Z vs. Preditor Linear



(d) Gráfico de Envelope

Observando a figura (a), pode-se notar que a suposição de independência é razoável.

Voltando a atenção para a figura (b), pode-se notar que não há muitos valores discrepantes, e que o maior ponto de alavanca encontrado foi o de valor 0,00697 aproximadamente.

Ao observar o gráfico da figura (c), pode-se notar que a forma do preditor linear é razoavelmente adequada, visto que os pontos estão seguindo a reta. Porém, pode-se notar um certo afastamento dessa forma linear para alguns pontos da variável Z.

Em relação ao gráfico da figura (d), observa-se o gráfico de envelope, que é crucial para identificar a adequabilidade do modelo. Observando o gráfico, pode-se notar que o modelo não parece estar totalmente adequado, levando em consideração que muitos pontos observados não estão dentro do intervalo criado por simulações do modelo.

Ou seja, não consegue-se simular, a partir do modelo, um comportamento similar ao comportamento dos dados observados, principalmente para os primeiros quantis e para valores mais à direita, ou seja, para os últimos quantis.

Para melhorar o modelo, poderia ser considerada uma outra distribuição para o preço. Além disso, seria possível utilizar modelos mistos, considerando o efeito do bairro, ou outros estratos. Como haviam muitas variáveis de data no banco, também poderia ser considerado um modelo de séries temporais.

A Código R

```r
# Arrumando o banco de dados --------------------------

### Carregando os pacotes exigidos
library(readr)
require(tidyverse)
set.seed(22019)
setwd("C:\\Users\\alvar\\Dropbox\\6Semestre\\Modelos_Lineares_Generalizados")

### Carregando os dados
listings <- read_csv("listings.csv")
calendar <- read_csv("calendar.csv")
### Limpando a base
calendar <- calendar %>%
  filter(!is.na(price)) %>% # filtrando os imóveis disponíveis
  mutate(price=str_replace(.$price, "\\$", "") %>%
           str_replace("\\,", "") %>%
           as.numeric()) # transformando os dados de preços em numérico
### Dividindo o banco de dados em treino e teste
n.obs <- nrow(calendar)
treino <- sample(n.obs, n.obs*.8)
teste <- setdiff(1:n.obs, treino)
calendar.treino <- calendar[treino, ]
calendar.teste <- calendar[teste, ]

### Definindo as informações a serem consideradas no modelo
covariaveis <- c("date", "host_since", "host_location", "host_response_time",
                 "host_response_rate", "host_acceptance_rate", "host_is_superhost",
                 "host_total_listings_count", "host_has_profile_pic", "host_identity_verified",
                 "neighbourhood_group_cleansed", "zipcode", "latitude", "longitude", "is_location_exact",
                 "property_type", "room_type", "accommodates", "bathrooms", "bedrooms", "beds",
                 "bed_type", "amenities", "square_feet", "security_deposit", "cleaning_fee",
                 "guests_included", "extra_people", "minimum_nights", "maximum_nights","number_of_reviews",
                 "first_review", "last_review", "review_scores_rating", "review_scores_accuracy",
                 "review_scores_cleanliness", "review_scores_checkin", "review_scores_communication",
                 "review_scores_location", "review_scores_value", "requires_license", "instant_bookable",
                 "cancellation_policy", "require_guest_profile_picture","require_guest_phone_verification",
                 "calculated_host_listings_count", "reviews_per_month")

# Início do nosso trabalho
listings2 = listings %>% dplyr::select(id, covariaveis[-1])
calendar2 = calendar %>% dplyr::select(-available)
df = left_join(calendar2, listings2, by = c("listing_id" = "id"))
```

```r
df = df[,-1]
df.treino = df[treino, ]
df.teste = df[teste, ]
dir.create('D:/AirBnB/cache')
lapply(ls(), function(i) saveRDS(get(i), paste0('D:/AirBnB/cache/', i, '.rds')))
rm(list = ls()[!ls()%in%c("df.treino", "df.teste")])
gc()

# Descritiva Preço
summary(df.treino$price)
hist(df.treino$price) # Escolhemos a Gama

# Arrumando as variáveis
require(lubridate)
df.treino2 <- df.treino %>%
  mutate(year = factor(year(date)),
         month = month(date, label = TRUE),
         week_day = wday(date, label = TRUE),
         days_as_host = as.numeric(date - host_since),
         host_response_time = factor(host_response_time,
                                     levels = c("N/A",
                                                "within an hour",
                                                "within a few hours",
                                                "within a day",
                                                "a few days or more"),
                                     labels = c("Unknown",
                                                "within an hour",
                                                "within a few hours",
                                                "within a day",
                                                "a few days or more"),
                                     ordered = TRUE),
         host_response_rate = as.numeric(sub("%","",host_response_rate))/100,
         host_acceptance_rate = as.numeric(sub("%","",host_acceptance_rate))/100,
         property_type = ifelse(is.na(property_type), "Unknown", property_type),
         room_type = factor(room_type),
         bed_type = factor(bed_type),
         security_deposit = as.numeric(gsub(",|[$]","",security_deposit)),
         cleaning_fee = as.numeric(gsub(",|[$]","",cleaning_fee)),
         extra_people = as.numeric(gsub(",|[$]","",extra_people)),
         days_first_review = as.numeric(date - first_review),
         days_last_review = as.numeric(date - last_review),
         cancellation_policy = factor(cancellation_policy)) %>%
  dplyr::select(-date,-host_since,-host_location,-zipcode,-first_review,-last_review,-requires_license)

amen = str_split(str_remove_all(df.treino2$amenities, '[{}\"]'),
                 ',', simplify = T) %>% as.data.frame() %>%
  gather(V, word) %>% filter(str_length(word) > 0) %>% distinct(word) %>% na.omit() %>% as_vector()
```

```r
names(amen) <- NULL

for(i in amen){
  df.treino2 = df.treino2 %>%
    mutate(!!i := str_detect(amenities, i))
}

df.treino2 <- dplyr::select(df.treino2, -amenities)

df.treino3 = df.treino2 %>%
  dplyr::select(-square_feet, -host_acceptance_rate) %>%
  replace_na(list(cleaning_fee = 0,
                  security_deposit = 0))
covars <- names(sort(colMeans(is.na(df.treino3))[colMeans(is.na(df.treino3)) > 0],
                     decreasing = T))

df.treino3 %>% filter_at(vars(covars), all_vars(!is.na(.))) %>% nrow()

df.treino4 = df.treino3 %>% filter_at(vars(covars), all_vars(!is.na(.))) %>%
  mutate_if(is.logical, list(~as.numeric(.)))

rm(list = ls()[!ls()%in%c("df.treino4")])
gc()

mod3 <- glm(price ~ . -`Dog(s)` - `Cat(s)` -`Other pet(s)`, family = Gamma(link="log"), data = df.treino4)
summary(mod3)

require(MASS)
mod0 = glm(price ~ 1, family = Gamma(link="log"), data = df.treino4)

ultimo_modelo <- glm(price ~ accommodates + room_type + neighbourhood_group_cleansed +
                       bedrooms + cleaning_fee + month + bathrooms + days_last_review +
                       host_is_superhost + review_scores_location + reviews_per_month +
                       property_type + `Elevator in Building` + week_day + calculated_host_listings_count +
                       latitude + bed_type + guests_included + longitude + `Cable TV` +
                       review_scores_value + review_scores_rating + host_has_profile_pic +
                       Doorman + `Laptop Friendly Workspace` + Shampoo + `Suitable for Events` +
                       `Smoking Allowed` + review_scores_accuracy + host_total_listings_count +
                       `24-Hour Check-in` + `Pets live on this property` + `Indoor Fireplace` +
                       `Air Conditioning` + host_identity_verified + `Fire Extinguisher` +
                       security_deposit + `Wireless Internet` + Internet + maximum_nights +
                       Iron + Hangers + is_location_exact + `Hot
```

```
    Tub` + host_response_time +
132                         `Wheelchair Accessible` + Kitchen + require_
    guest_profile_picture +
133                         `Safety Card` + beds + `Lock on Bedroom Door
    ` + `Hair Dryer` +
134                         `Washer / Dryer` + Heating + cancellation_
    policy + `Smoke Detector` +
135                         Breakfast + days_first_review + number_of_
    reviews + minimum_nights +
136                         days_as_host + instant_bookable + TV + host_
    response_rate +
137                         `Family/Kid Friendly` + `Carbon Monoxide
    Detector` + review_scores_checkin +
138                         `Buzzer/Wireless Intercom` + extra_people +
    Essentials +
139                         review_scores_cleanliness + Pool + year +
    review_scores_communication +
140                         `First Aid Kit` + `Free Parking on Premises`
     + require_guest_phone_verification +
141                         `Pets Allowed`, family = Gamma(link="log"),
    data = df.treino4)
142
143 mod_step = stepAIC(ultimo_modelo, direction = "forward", scope =
    list(upper = mod3, lower = mod0))
144
145
146
147 saveRDS(mod_step, 'D:/AirBnB/cache/modelo_stepAIC.rds')
148 saveRDS(mod3, 'D:/AirBnB/cache/modelo_3.rds')
149
150 rm(mod_step, mod3, df.treino4)
151 gc()
152 # Imputação ----------------------------
153
154 require(tidyverse)
155 require(lubridate)
156 df.teste <- readRDS('D:/AirBnB/cache/df.teste.rds')
157 df.teste2 <- df.teste %>%
158   mutate(year = factor(year(date)),
159          month = month(date, label = TRUE),
160          week_day = wday(date, label = TRUE),
161          days_as_host = as.numeric(date - host_since),
162          host_response_time = factor(host_response_time,
163                                      levels = c("N/A",
164                                                 "within an hour",
165                                                 "within a few hours
    ",
166                                                 "within a day",
167                                                 "a few days or more
    "),
168                                      labels = c("Unknown",
169                                                 "within an hour",
170                                                 "within a few hours
    ",
171                                                 "within a day",
172                                                 "a few days or more
    "),
173                                      ordered = TRUE),
```

```
174           host_response_rate = as.numeric(sub("%","",host_response_
    rate))/100,
175           host_acceptance_rate = as.numeric(sub("%","",host_
    acceptance_rate))/100,
176           property_type = ifelse(is.na(property_type), "Unknown",
    property_type),
177           room_type = factor(room_type),
178           bed_type = factor(bed_type),
179           security_deposit = as.numeric(gsub(",|[$]","",security_
    deposit)),
180           cleaning_fee = as.numeric(gsub(",|[$]","",cleaning_fee)),
181           extra_people = as.numeric(gsub(",|[$]","",extra_people)),
182           days_first_review = as.numeric(date - first_review),
183           days_last_review = as.numeric(date - last_review),
184           cancellation_policy = factor(cancellation_policy)) %>%
185   dplyr::select(-date,-host_since,-host_location,-zipcode,-first_
    review,-last_review,-requires_license)
186
187 amen <- str_split(str_remove_all(df.teste2$amenities, '[{}\"]'),
188                   ',', simplify = T) %>% as.data.frame() %>%
189   gather(V, word) %>% filter(str_length(word) > 0) %>% distinct(
    word) %>% na.omit() %>% as_vector()
190
191 names(amen) <- NULL
192
193 for(i in amen){
194   df.teste2 <- df.teste2 %>%
195     mutate(!!i := str_detect(amenities, i))
196 }
197
198 df.teste2 <- dplyr::select(df.teste2, -amenities)
199
200 saveRDS(df.teste2, 'D:/AirBnB/cache/df.teste2.rds')
201
202 df.treino3 <- readRDS('D:/AirBnB/cache/df.treino3.rds')
203 df.teste2 <- readRDS('D:/AirBnB/cache/df.teste2.rds')
204 teste <- df.teste2
205
206
207 # Colunas numericas faltando
208
209 colSums(is.na(df.treino3))[colMeans(is.na(df.treino3))>0] # Remove
    os 67 errados
210
211 df_correto <- df.treino3 %>% filter(!is.na(days_as_host))
212
213 colSums(is.na(df_correto))[colMeans(is.na(df_correto))>0] # Todos
    que tinham 67 desapareceram
214 col_incomp <- names(colMeans(is.na(df_correto))>0)
215
216 col_comp <- names(df_correto)[colMeans(is.na(df_correto))==0][-1] #
     usar as colunas completas para predizer as incompletas
    retirando price
217
218 fo_host <- paste0('~', str_subset(col_comp, 'host')[-1], '~',
    collapse = '+') # Predizer apenas para as colunas de host, não
    converge se usar todas
219 # as colunas completas
```

```r

# Métodos escolhidos ---------------
# host_response_rate, reviews_scores: Regressão beta
# bathrooms, bedrooms, beds, reviews_per_month, days_first_review, days_last_review: regressão poisson

# regressão beta ------------------

require(betareg)
# host_response_rate
temp <- df_correto
temp$host_response_rate[temp$host_response_rate==1] <- .99
temp$host_response_rate[temp$host_response_rate==0] <- .01


hrr_betareg_mod <- betareg(as.formula(paste('host_response_rate ~', fo_host)), data = temp)
hrr_predicted <- predict(hrr_betareg_mod, newdata = df_correto)
df_correto <- df_correto %>%
  mutate(host_response_rate_informed = as.numeric(!is.na(host_response_rate)),
         host_response_rate = coalesce(host_response_rate, hrr_predicted))

test_pred <- predict(hrr_betareg_mod, newdata = teste)
teste <- teste %>%
  mutate(host_response_rate_informed = as.numeric(!is.na(host_response_rate)),
         host_response_rate = coalesce(host_response_rate, test_pred, mean(host_response_rate, na.rm = T)))

saveRDS(hrr_betareg_mod, 'D:/AirBnB/cache/host_response_rate_betareg.rds')

# review_scores_rating escala de 0 a 100

temp <- df_correto
temp <- temp %>%
  mutate_at(vars(starts_with('review_scores_')), list(~./max(., na.rm = T))) %>%
  mutate_at(vars(starts_with('review_scores_')), list(~case_when(
    .==1~.99,
    .==0~.01,
    TRUE~.
  )))

rev_sc_rating <- betareg(as.formula(paste('review_scores_rating ~', fo_host)), data = temp)
rev_sc_predicted <- predict(rev_sc_rating, newdata = df_correto)

df_correto <- df_correto %>%
  mutate(review_scores_rating_informed = as.numeric(!is.na(review_scores_rating)),
         review_scores_rating = coalesce(review_scores_rating, rev_sc_predicted*100))

test_pred <- predict(rev_sc_rating, newdata = teste)
```

```r
teste <- teste %>%
  mutate(review_scores_rating_informed = as.numeric(!is.na(review_scores_rating)),
         review_scores_rating = coalesce(review_scores_rating, test_pred*100, mean(review_scores_rating, na.rm = T)))
rm(rev_sc_rating, rev_sc_predicted, hrr_betareg_mod, hrr_predicted)
gc()
# Os outros reviews_scores possui escala de 0 a 10
col_reviews <- str_subset(col_incomp, 'review_scores_(?!rating)')
for(j in col_reviews){
  # j= col_reviews[1]
  j <- as.name(j)


  rev_sc_reg <- betareg(as.formula(paste(j, '~', fo_host)), data = temp)
  rev_sc_predicted <- predict(rev_sc_reg, newdata = df_correto)

  df_correto <- df_correto %>%
    mutate(!!paste0(j, '_informed') := as.numeric(!is.na(!!j)),
           !!j := coalesce(!!j, rev_sc_predicted*10))

  test_pred <- predict(rev_sc_reg, newdata = teste)
  teste <- teste %>%
    mutate(!!paste0(j, '_informed') := as.numeric(!is.na(!!j)),
           !!j := coalesce(!!j, test_pred*10, mean(!!j, na.rm = T))
    )

  saveRDS(rev_sc_reg, paste0('D:/AirBnB/cache/', j, '_betareg.rds')
    )
  cat('writted ', paste0('D:/AirBnB/cache/', j, '_betareg.rds'), '\n')
}
rm(temp, col_reviews); gc()
# Regressão Poisson ---------------

poi_reg_var <- unlist(str_split('bathrooms,bedrooms,beds,reviews_per_month,days_first_review,days_last_review', ','))

for(j in poi_reg_var){
  # j= poi_reg_var[2]
  j <- as.name(j)


  poi_reg <- glm(as.formula(paste(j, '~', fo_host)), data = df_correto, family = poisson('log'))
  poi_predicted <- predict(rev_sc_reg, newdata = df_correto, type='response')

  df_correto <- df_correto %>%
    mutate(!!paste0(j, '_informed') := as.numeric(!is.na(!!j)),
           !!j := coalesce(!!j, rev_sc_predicted))

  test_pred <- predict(poi_reg, newdata = teste)
  teste <- teste %>%
    mutate(!!paste0(j, '_informed') := as.numeric(!is.na(!!j)),
           !!j := coalesce(!!j, test_pred, mean(!!j, na.rm = T)))

```

```
316
317   saveRDS(rev_sc_reg, paste0('D:/AirBnB/cache/', j, '_poisson_reg.
      rds'))
318   cat('writted ',paste0('D:/AirBnB/cache/', j, '_poisson_reg.rds'),
        '\n')
319 }
320 teste <- teste %>% replace_na(list(cleaning_fee = 0,
321                                    security_deposit = 0))
322 saveRDS(teste, 'D:/AirBnB/cache/teste_imputed.rds')
323 saveRDS(df_correto, 'D:/AirBnB/cache/df_imputed.rds')
324
325 rm(list = ls()); gc()
326
327 # Construção de Modelo -----------------------------
328
329 require(tidyverse)
330 df_imputed <- readRDS('D:/AirBnB/cache/df_imputed.rds')
331 df_imputed <- df_imputed %>% mutate_if(is.factor, list(~factor(.,
      ordered = F))) %>%
332   mutate_if(is.logical, list(~as.numeric(.)))
333 # Modelo escolhido pelo método automático, com algumas alterações
334 fit <- glm(price ~ host_response_time + host_is_superhost + host_
      total_listings_count +
335              host_has_profile_pic + neighbourhood_group_cleansed +
      latitude +
336              is_location_exact + property_type + room_type +
      accommodates +
337              bathrooms + bedrooms + security_deposit + cleaning_fee
       +
338              guests_included + number_of_reviews + review_scores_
      rating +
339              review_scores_accuracy + review_scores_location +
      review_scores_value +
340              cancellation_policy + calculated_host_listings_count +
       reviews_per_month +
341              month + week_day + days_first_review + days_last_
      review +
342              Internet + TV + `Wireless Internet` + `Cable TV` +
343              `Elevator in Building` + longitude, family = Gamma('
      log'), data = df_imputed)
344
345 summary(fit)
346
347 teste_inputed <- readRDS('D:/AirBnB/cache/teste_imputed.rds')
348 teste_inputed <- teste_inputed %>%
349   replace_na(list(host_is_superhost = T, # imputado pela Moda
350                   host_response_time = 'Unknown', # Nova categoria
351                   host_has_profile_pic = T)) %>% # Moda
352   mutate(host_total_listings_count = ifelse(is.na(host_total_
      listings_count), mean(host_total_listings_count, na.rm = T),
353                                             host_total_listings_
      count)) %>% # host_total_listings_count imputado pela média
354   mutate_if(is.factor, list(~factor(., ordered = F))) %>%
355   mutate_if(is.logical, list(~as.numeric(.)))
356
357
358
359
```

```r
predito_teste <- predict(fit, newdata = teste_inputed, type = 'response') # Predizendo

# Calculando o RMSE
(RMSE_treino <- sqrt(mean((df_imputed$price - fit$fitted.values)^2))) # EQM treino
(RMSE_teste <- sqrt(mean((teste_inputed$price - predito_teste)^2))) # EQM teste
cbind(RMSE_treino, RMSE_teste)


# Gráficos de Diagnóstico

h <- hatvalues(fit)



ro <- residuals.glm(fit, type = 'deviance')

fi <- MASS::gamma.shape(fit)$alpha


RcD <- ro*sqrt(fi/(1 - h))



require(ggplot2)

ggplot(data.frame(x = 1:1000, y = RcD[1:1000]), aes(x, y)) + geom_point(color = 'darkred') +
  geom_hline(yintercept = 0, color = 'darkblue', size = 1.0) +
  labs(x = 'Índice', y = 'Resíduo Componente do Desvio') +
  theme_bw() +
  theme(panel.border = element_blank(),
        axis.line = element_line('black'))


ggplot(data.frame(x = fit$fitted.values[1:10000], y = RcD[1:10000]), aes(x, y)) + geom_point(color = 'darkred') +
  labs(x = 'Valor Predito', y = 'Resíduo Componente do Desvio') +
  theme_bw() +
  theme(panel.border = element_blank(),
        axis.line = element_line('black'))

w <- fit$weights
eta <- predict(fit)
z <- eta + resid(fit, type="pearson")/sqrt(w)
ggplot(data.frame(x = eta[1:1000], y = z[1:1000]), aes(x, y)) + geom_point(color = 'darkred') +
  labs(x = 'Preditor Linear', y = 'Variável z') +
  theme_bw() +
  theme(panel.border = element_blank(),
        axis.line = element_line('black')) +
  geom_abline(intercept = 0, slope = 1, color = 'darkblue', size = 1.0)


# Gráfico de Envelope -------
```

```r
set.seed(22019)
ind = sample(1:747566 , 1000)
df_graf = df_imputed[ind, ]
fit_graf = glm(price ~ host_response_time + host_is_superhost +
    host_total_listings_count +
                   host_has_profile_pic + neighbourhood_group_
    cleansed + latitude +
                   is_location_exact + property_type + room_type +
    accommodates +
                   bathrooms + bedrooms + security_deposit + cleaning
    _fee +
                   guests_included + number_of_reviews + review_
    scores_rating +
                   review_scores_accuracy + review_scores_location +
    review_scores_value +
                   cancellation_policy + calculated_host_listings_
    count + reviews_per_month +
                   month + week_day + days_first_review + days_last_
    review +
                   Internet + TV + `Wireless Internet` + `Cable TV` +
                   `Elevator in Building` + longitude, family = Gamma
    ('log'), data = df_graf)
envelgama <- function(fit.model,ligacao,estphi){
  # fit.model: objeto com o resultado do ajuste do MLG obtido
    atravs da funo glm
  # ligacao: funo de ligao (mesmo nome usado pela funo glm (colocar
     entre aspas)
  # estphi: mtodo de estimao para o parmetro phi
  #         1 - Mxima verossimilhana
  #         2 - Mtodo dos Momentos
  #         3 - Default do R

  #par(mfrow=c(1,1))
  X <- model.matrix(fit.model)
  n <- nrow(X)
  p <- ncol(X)
  h <- hatvalues(fit.model)
  ro <- resid(fit.model,type="response")
  #fi <- (n-p)/sum((ro/(fitted(fit.model)))^ 2)
  #library(MASS)
  #fi <- gamma.shape(fit.model)$alpha
  if (estphi == 1)
  {
    library(MASS)
    fi <- gamma.shape(fit.model)$alpha
  }
  else if (estphi == 2)
  {
    #ro <- resid(fit.model,type="response")
    fi <-(n-p)/sum((ro/(fitted(fit.model)))^ 2)}
  else if (estphi == 3)
  {
    fi <- 1/summary(fit.model)$dispersion
  }
  td <- resid(fit.model,type="deviance")*sqrt(fi/(1-h))
  #
  e <- matrix(0,n,100)
  #
```

```
458   for(i in 1:100){
459     resp <- rgamma(n,fi)
460     resp <- (fitted(fit.model)/fi)*resp
461     fit <- glm(resp ~ X, family=Gamma(link=ligacao))
462     h <- hatvalues(fit)
463     if (h==1){h = 0.99}
464     ro <- resid(fit,type="response")
465     #phi <- (n-p)/sum((ro/(fitted(fit)))^ 2)
466     #library(MASS)
467     #fi <- gamma.shape(fit.model)$alpha
468     if (estphi == 1)
469     {
470       library(MASS)
471       phi <- gamma.shape(fit.model)$alpha
472     }
473     else if (estphi == 2)
474     {
475       #ro <- resid(fit.model,type="response")
476       phi <-(n-p)/sum((ro/(fitted(fit.model)))^ 2)}
477     else if (estphi == 3)
478     {
479       phi <- 1/summary(fit.model)$dispersion
480     }
481
482     e[,i] <- sort(resid(fit,type="deviance")*sqrt(phi/(1-h)))}
483   #
484   e1 <- numeric(n)
485   e2 <- numeric(n)
486   #
487   for(i in 1:n){
488     eo <- sort(e[i,])
489     e1[i] <- (eo[2]+eo[3])/2
490     e2[i] <- (eo[97]+eo[98])/2}
491   #
492   med <- apply(e,1,mean)
493   faixa <- range(td,e1,e2)
494   #par(pty="s")
495   qqnorm(td, xlab="Percentil da N(0,1)",
496          ylab="Resduo Componente do Desvio", ylim=faixa, pch=16,
       main="",cex=.5,cex.axis=1.1,cex.lab=1.1)
497   par(new=T)
498   #
499   qqnorm(e1,axes=F,xlab="",ylab="",type="l",ylim=faixa,lty=1, main=
       "")
500   par(new=T)
501   qqnorm(e2,axes=F,xlab="",ylab="", type="l",ylim=faixa,lty=1, main
       ="")
502   par(new=T)
503   qqnorm(med,axes=F,xlab="", ylab="", type="l",ylim=faixa,lty=2,
       main="")
504   #-------------------------------------------------------------#
505   td <<- td
506   med <<- med
507   e1 <<- e1
508   e2 <<- e2
509 }
510
511 envelgama(fit_graf, 'log', 3)
```

```
ggplot(data.frame(td, med, e1, e2)) +
  stat_qq(aes(sample = td), col = 'darkred') +
  stat_qq(aes(sample = e1), geom = 'line') +
  stat_qq(aes(sample = e2), geom = 'line') +
  stat_qq(aes(sample = med), geom = 'line', linetype = 2, color = '
   darkblue') +
  theme_bw() +
  theme(panel.border = element_blank(),
        axis.line = element_line('black'))
```