Análise de Clusters com PCA

Adriano Abelaira Paz

15/05/2020

Análise de Clusters com PCA

Diretrizes:

Será usado o R para minerar os dados de interesse. Será seguido o seguinte roteiro na tarefa:

1. Entendimento do problema: identificar e definir o problema que está sendo abordado. Como uma solução de mineração de dados resolverá este problema?
2. Entendimento dos dados: descrever os dados (e fontes de dados) que darão suporte à mineração de dados para solucionar o problema. Fazendo alguns plots com descrições estatísticas. Analisar as correlações entre variáveis e suas distribuições. Essa será a oportunidade para levantar pontos de atenção, como dados faltantes, categorias mal-definidas, desbalanceamento entre categorias e variáveis altamente correlacionadas. As hipóteses e conclusões serão tiradas olhando essas descrições estatísticas.
3. Preparação dos dados: especificar quais pré-processamentos são necessários para a análise. Incluindo codificações de dados categóricos, normalizações e redução de dimensionalidade. Comparar o dataset antes do pré-processamento e depois.
4. Modelagem: escolher dois algoritmos de clusterização a serem utilizados. Identificar quais parâmetros podem ser variados nesses algoritmos afim de se obter o melhor resultado.
5. Avaliação: descubrir o melhor número de clusters e descrever interpretações para eles. Avaliando a diferença de interpretação entre os algoritmos.
6. Relatório: documentar através de textos e gráficos a análise.

Importando, analisando e tratando os *Data Set's*

Importando os dados de COVID19 por cidade.

```
covid <- fread('../data/covid19.csv')
```

Verificando a estrutura do *Data Set*.

Possui um conjunto de dados por cidade e somente dois tipos de locais: cidade e estado.

```
unique(covid$place_type)
```

```
## [1] "city"  "state"
```

Possui *2.382* códigos de cidade sendo uma linha com *NA*.

```r
length(unique(covid$city_ibge_code))
```

```
## [1] 2382
```

```r
unique(covid$city_ibge_code) %>% anyNA
```

```
## [1] TRUE
```

Possui 20 elementos do tipo cidade com *NA*.

```r
nrow(covid[is_last == TRUE & is.na(city_ibge_code),])
```

```
## [1] 20
```

```r
unique(covid[is_last == TRUE & is.na(city_ibge_code),]$place_type)
```

```
## [1] "city"
```

Obtendo somente os dados por cidade.

```r
covid_cidade <- covid[is_last == TRUE & place_type == 'city',]
```

A difetença entre o *length* e o *nrow* se dá porque existem 20 elementos com *NA* no *city_ibge_code* mas esses elementos são de UF diferentes, portanto estão em linhas diferentes.

```r
length(unique(covid_cidade$city_ibge_code))
```

```
## [1] 2355
```

```r
nrow(covid_cidade)
```

```
## [1] 2374
```

Retirando as linhas com atributo *city_ibge_code* com *NA*.

```r
covid_cidade <- covid_cidade[!is.na(city_ibge_code),]
length(unique(covid_cidade$city_ibge_code))
```

```
## [1] 2354
```

```r
nrow(covid_cidade)
```

```
## [1] 2354
```

Importando os dados de cadastro dos Municípios.

```
cidade_df <- read_xls('../data/CODIGO_MUNICIPIO.xls')
```

Selecioando os atributos de interesse.

```
cidade <- data.table()
cidade[, ':='(uf = as.integer(cidade_df$UF),
              nome_uf = cidade_df$Nome_UF,
              codigo_cidade = as.integer(cidade_df$Município),
              codigo_ibge_completo = as.integer(cidade_df$`Código Município Completo`),
              nome_cidade = cidade_df$Nome_Município,
              codigo_ibge_parcial =
                as.integer(str_extract(cidade_df$`Código Município Completo`,'^\\d{6}')))]
str(cidade)
```

```
## Classes 'data.table' and 'data.frame':   5570 obs. of  6 variables:
##  $ uf                  : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ nome_uf             : chr  "Rondônia" "Rondônia" "Rondônia" "Rondônia" ...
##  $ codigo_cidade       : int  15 379 403 346 23 452 31 601 49 700 ...
##  $ codigo_ibge_completo: int  1100015 1100379 1100403 1100346 1100023 1100452 1100031 1100601 110004
##  $ nome_cidade         : chr  "Alta Floresta D'Oeste" "Alto Alegre dos Parecis" "Alto Paraíso" "Alvo
##  $ codigo_ibge_parcial : int  110001 110037 110040 110034 110002 110045 110003 110060 110004 110070
##  - attr(*, ".internal.selfref")=<externalptr>
```

Importando os dados de leitos em hostpital para o SUS por cidade.

```
hospital <- fread('../data/RecursosFisicosHospitalar.csv')
```

Selecionando os atributos de interesse.

```
codigo_ibge <- str_extract(hospital$Município, "^\\d{6}")
hospital[,':='(codigo_ibge = as.integer(codigo_ibge),
               total_leitos = as.integer(Total))]
hospital <- hospital[,c('codigo_ibge', 'total_leitos')]
hospital <- hospital[!is.na(codigo_ibge),]
str(hospital)
```

```
## Classes 'data.table' and 'data.frame':   5597 obs. of  2 variables:
##  $ codigo_ibge : int  110001 110037 110040 110034 110002 110045 110003 110060 110004 110070 ...
##  $ total_leitos: int  49 16 16 35 240 66 9 26 360 19 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Importando os dados de renda por cidade.

```
renda_df <- read.dbf('../data/RENDABR10.dbf', as.is = TRUE)
str(renda_df)
```

```
## 'data.frame':    25400 obs. of  15 variables:
##  $ MUNCOD    : chr  "110001" "110001" "110001" "110001" ...
##  $ ANO       : chr  "2010" "2010" "2010" "2010" ...
##  $ CORRACA   : chr  "1" "2" "3" "4" ...
```

```
##  $ SITUACAO  : chr  "I" "I" "I" "I" ...
##  $ NUMRENDA  : num  6000725 378315 31516 4791228 68893 ...
##  $ DENRENDA  : int  10482 1079 95 11944 497 31842 7735 1904 48188 97 ...
##  $ DENCRIREND: int  2618 186 0 3348 182 8801 1472 553 14015 12 ...
##  $ NUMPOBRES : int  4831 539 46 5802 450 7899 2794 642 16513 25 ...
##  $ NUMPOBRESX: int  2804 182 12 3048 250 3446 1036 316 6712 12 ...
##  $ NUMCRIPOB : int  1645 144 0 2096 165 3194 807 243 6894 0 ...
##  $ NUMCRIPOBX: int  914 46 0 1160 107 1406 332 98 2933 0 ...
##  $ NUMDESOCUP: int  181 18 0 304 0 686 159 12 1181 0 ...
##  $ DENDESOCUP: int  4451 554 65 4906 67 15703 4535 966 22677 66 ...
##  $ NUMTRABINF: int  153 16 0 159 0 323 73 10 710 0 ...
##  $ DENTRABINF: int  1127 80 8 1583 108 3714 719 193 6360 0 ...
##  - attr(*, "data_types")= chr  "C" "C" "C" "C" ...
```

Tratando o atributo COR/RAÇA.

```
cor_raca <- case_when(renda_df$CORRACA == 1 ~ 'Branco',
                      renda_df$CORRACA == 2 ~ 'Negro',
                      renda_df$CORRACA == 3 ~ 'Amarelo',
                      renda_df$CORRACA == 4 ~ 'Pardo',
                      renda_df$CORRACA == 5 ~ 'Indígena',
                      renda_df$CORRACA == 0 ~ 'Sem declaração')
```

Selecionando os atributos de interesse.

```
renda <- data.table()
renda[, ':='(codigo_ibge = as.integer(renda_df$MUNCOD),
             cor_raca = factor(cor_raca,
                               levels = c('Branco', 'Negro', 'Amarelo', 'Pardo',
                                          'Indígena', 'Sem declaração')),
             renda_domiciliar = renda_df$NUMRENDA,
             populacao = renda_df$DENRENDA,
             num_pobre = renda_df$NUMPOBRES,
             num_pobre_extremo = renda_df$NUMPOBRESX,
             num_desocupado = renda_df$NUMDESOCUP,
             total_pop_economico = renda_df$DENDESOCUP,
             num_trab_infantil = renda_df$NUMTRABINF,
             total_pop_infantil = renda_df$DENTRABINF)]
str(renda)
```

```
## Classes 'data.table' and 'data.frame':   25400 obs. of  10 variables:
##  $ codigo_ibge        : int  110001 110001 110001 110001 110001 110002 110002 110002 110002 110002 .
##  $ cor_raca           : Factor w/ 6 levels "Branco","Negro",..: 1 2 3 4 5 1 2 3 4 5 ...
##  $ renda_domiciliar   : num  6000725 378315 31516 4791228 68893 ...
##  $ populacao          : int  10482 1079 95 11944 497 31842 7735 1904 48188 97 ...
##  $ num_pobre          : int  4831 539 46 5802 450 7899 2794 642 16513 25 ...
##  $ num_pobre_extremo  : int  2804 182 12 3048 250 3446 1036 316 6712 12 ...
##  $ num_desocupado     : int  181 18 0 304 0 686 159 12 1181 0 ...
##  $ total_pop_economico: int  4451 554 65 4906 67 15703 4535 966 22677 66 ...
##  $ num_trab_infantil  : int  153 16 0 159 0 323 73 10 710 0 ...
##  $ total_pop_infantil : int  1127 80 8 1583 108 3714 719 193 6360 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Consolidando os dados em um único *Data Set*.

```
cidade_hospital <- merge(cidade, hospital, by.x = 'codigo_ibge_parcial',
                         by.y = 'codigo_ibge', all = TRUE)
str(cidade_hospital)
```

```
## Classes 'data.table' and 'data.frame':   5597 obs. of  7 variables:
##  $ codigo_ibge_parcial : int  0 110000 110001 110002 110003 110004 110005 110006 110007 110008 ...
##  $ uf                  : int  NA NA 11 11 11 11 11 11 11 11 ...
##  $ nome_uf             : chr  NA NA "Rondônia" "Rondônia" ...
##  $ codigo_cidade       : int  NA NA 15 23 31 49 56 64 72 80 ...
##  $ codigo_ibge_completo: int  NA NA 1100015 1100023 1100031 1100049 1100056 1100064 1100072 1100080
##  $ nome_cidade         : chr  NA NA "Alta Floresta D'Oeste" "Ariquemes" ...
##  $ total_leitos        : int  NA NA 49 240 9 360 40 50 16 45 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "codigo_ibge_parcial"
```

```
cidade_hospital_renda <- merge(cidade_hospital, renda,
                               by.x = 'codigo_ibge_parcial',
                               by.y = 'codigo_ibge', all = TRUE)
str(cidade_hospital_renda)
```

```
## Classes 'data.table' and 'data.frame':   25432 obs. of  16 variables:
##  $ codigo_ibge_parcial : int  0 110000 110001 110001 110001 110001 110001 110002 110002 110002 ...
##  $ uf                  : int  NA NA 11 11 11 11 11 11 11 11 ...
##  $ nome_uf             : chr  NA NA "Rondônia" "Rondônia" ...
##  $ codigo_cidade       : int  NA NA 15 15 15 15 15 23 23 23 ...
##  $ codigo_ibge_completo: int  NA NA 1100015 1100015 1100015 1100015 1100015 1100023 1100023 1100023
##  $ nome_cidade         : chr  NA NA "Alta Floresta D'Oeste" "Alta Floresta D'Oeste" ...
##  $ total_leitos        : int  NA NA 49 49 49 49 49 240 240 240 ...
##  $ cor_raca            : Factor w/ 6 levels "Branco","Negro",..: NA NA 1 2 3 4 5 1 2 3 ...
##  $ renda_domiciliar    : num  NA NA 6000725 378315 31516 ...
##  $ populacao           : int  NA NA 10482 1079 95 11944 497 31842 7735 1904 ...
##  $ num_pobre           : int  NA NA 4831 539 46 5802 450 7899 2794 642 ...
##  $ num_pobre_extremo   : int  NA NA 2804 182 12 3048 250 3446 1036 316 ...
##  $ num_desocupado      : int  NA NA 181 18 0 304 0 686 159 12 ...
##  $ total_pop_economico : int  NA NA 4451 554 65 4906 67 15703 4535 966 ...
##  $ num_trab_infantil   : int  NA NA 153 16 0 159 0 323 73 10 ...
##  $ total_pop_infantil  : int  NA NA 1127 80 8 1583 108 3714 719 193 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "sorted")= chr "codigo_ibge_parcial"
```

```
cidade_hospital_renda <- cidade_hospital_renda[!is.na(codigo_ibge_completo),]

covid_cidade_ibge <- merge(cidade_hospital_renda, covid_cidade,
                           by.x = 'codigo_ibge_completo',
                           by.y = 'city_ibge_code', all = TRUE)
covid_cidade_ibge <- covid_cidade_ibge[,-c('codigo_ibge_parcial',
                                           'codigo_ibge_completo','date',
                                           'place_type','is_last','state','city')]
setnames(covid_cidade_ibge,
         c('confirmed','deaths','estimated_population_2019',
```

```
          'confirmed_per_100k_inhabitants','death_rate'),
        c('num_caso_confirmado','num_mortes','pop_estimada_2019',
          'caso_confirmado_100k_habitantes',
          'taxa_mortalidade'))

str(covid_cidade_ibge)
```

```
## Classes 'data.table' and 'data.frame':   25405 obs. of  19 variables:
##  $ uf                            : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ nome_uf                       : chr  "Rondônia" "Rondônia" "Rondônia" "Rondônia" ...
##  $ codigo_cidade                 : int  15 15 15 15 15 23 23 23 23 23 ...
##  $ nome_cidade                   : chr  "Alta Floresta D'Oeste" "Alta Floresta D'Oeste" "Alta Flores
##  $ total_leitos                  : int  49 49 49 49 49 240 240 240 240 240 ...
##  $ cor_raca                      : Factor w/ 6 levels "Branco","Negro",..: 1 2 3 4 5 1 2 3 4 5 ...
##  $ renda_domiciliar              : num  6000725 378315 31516 4791228 68893 ...
##  $ populacao                     : int  10482 1079 95 11944 497 31842 7735 1904 48188 97 ...
##  $ num_pobre                     : int  4831 539 46 5802 450 7899 2794 642 16513 25 ...
##  $ num_pobre_extremo             : int  2804 182 12 3048 250 3446 1036 316 6712 12 ...
##  $ num_desocupado                : int  181 18 0 304 0 686 159 12 1181 0 ...
##  $ total_pop_economico           : int  4451 554 65 4906 67 15703 4535 966 22677 66 ...
##  $ num_trab_infantil             : int  153 16 0 159 0 323 73 10 710 0 ...
##  $ total_pop_infantil            : int  1127 80 8 1583 108 3714 719 193 6360 0 ...
##  $ num_caso_confirmado           : int  1 1 1 1 1 89 89 89 89 89 ...
##  $ num_mortes                    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pop_estimada_2019             : int  22945 22945 22945 22945 22945 107863 107863 107863 107863 10
##  $ caso_confirmado_100k_habitantes: num  4.36 4.36 4.36 4.36 4.36 ...
##  $ taxa_mortalidade              : num  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Descrição dos dados:

Variável	Descrição
uf	Código unidade federativa
nome_uf	Nome da unidade federativa
codigo_cidade	Código IBGE da cidade
nome_cidade	Nome da cidade
total_leitos	Total de leitos disponíveis pelo SUS
cor_raca	Cor/Raça
renda_domiciliar	Somatório da renda média domiciliar per capita
populacao	População considerada censo IBGE 2010
num_pobre	População com renda média domiciliar per capita menor que 1/2 salário mínimo
num_pobre_extermo	População com renda média domiciliar per capita menor que 1/4 salário mínimo
num_desocupado	População residente economicamente ativa de 16 anos e mais que se encontra sem trabalho
total_pop_economico	População residente economicamente ativa de 16 anos e mais
num_trab_infantil	População residente com 10 a 15 anos de idade que se encontra trabalhando ou procurando trabalho
total_pop_infantil	População total residente com 10 a 15 anos de idade
num_caso_confirmado	Número de casos confirmados
num_mortes	Número de mortes

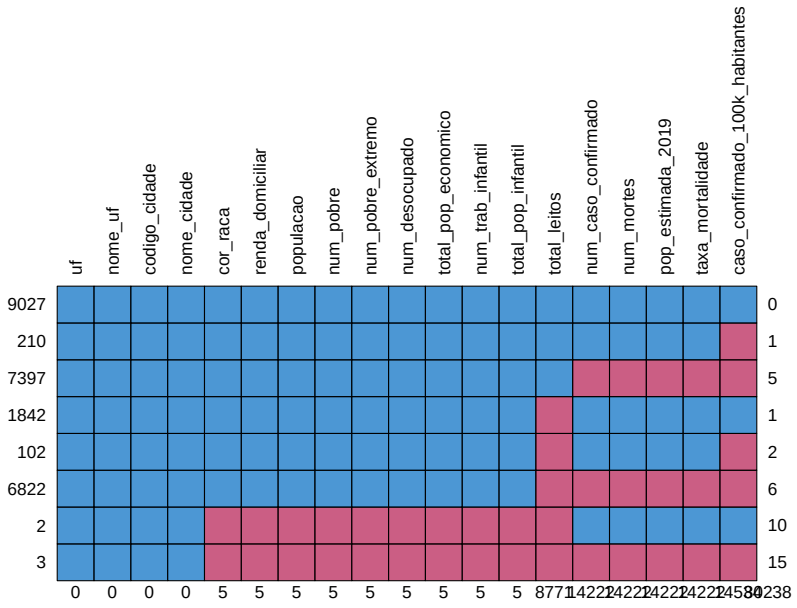
Variável	Descrição
pop_estimada_2019	População estimada para esse município/estado em 2019
caso_confirmado_100k_habitantes	Número de casos confirmados por 100000 habitantes
taxa_mortalidade	Taxa de mortalidade (mortes / confirmados)

Verificando e existência de *NA's*.

```
anyNA(covid_cidade_ibge)
```

```
## [1] TRUE
```

```
na_covid <- md.pattern(covid_cidade_ibge, rotate.names = TRUE)
```



Nota-se que só existem *9.027* observações completas, ou seja, sem nenhum atributo com *NA*.

Estruturando um *Data Set* com o resumo dos atributos com *NA*.

```
na_covid_resumo <- data.frame(atributo = names(na_covid[9,1:19]),
                              perc_na = na_covid[9,1:19]/nrow(covid_cidade_ibge) * 100)
rownames(na_covid_resumo) <- NULL
na_covid_resumo
```

```
##                          atributo     perc_na
## 1                              uf  0.00000000
```

```
## 2                         nome_uf  0.00000000
## 3                  codigo_cidade  0.00000000
## 4                    nome_cidade  0.00000000
## 5                       cor_raca  0.01968117
## 6                renda_domiciliar  0.01968117
## 7                       populacao  0.01968117
## 8                       num_pobre  0.01968117
## 9                num_pobre_extremo  0.01968117
## 10                  num_desocupado  0.01968117
## 11             total_pop_economico  0.01968117
## 12               num_trab_infantil  0.01968117
## 13              total_pop_infantil  0.01968117
## 14                    total_leitos 34.52469986
## 15             num_caso_confirmado 55.98110608
## 16                     num_mortes 55.98110608
## 17               pop_estimada_2019 55.98110608
## 18                taxa_mortalidade 55.98110608
## 19 caso_confirmado_100k_habitantes 57.20921079
```

```
na_covid[9,1:19]
```

```
##                              uf                         nome_uf
##                               0                               0
##                     codigo_cidade                     nome_cidade
##                               0                               0
##                        cor_raca                  renda_domiciliar
##                               5                               5
##                        populacao                       num_pobre
##                               5                               5
##                 num_pobre_extremo                   num_desocupado
##                               5                               5
##              total_pop_economico                num_trab_infantil
##                               5                               5
##               total_pop_infantil                     total_leitos
##                               5                            8771
##              num_caso_confirmado                      num_mortes
##                           14222                           14222
##                pop_estimada_2019                 taxa_mortalidade
##                           14222                           14222
## caso_confirmado_100k_habitantes
##                           14534
```

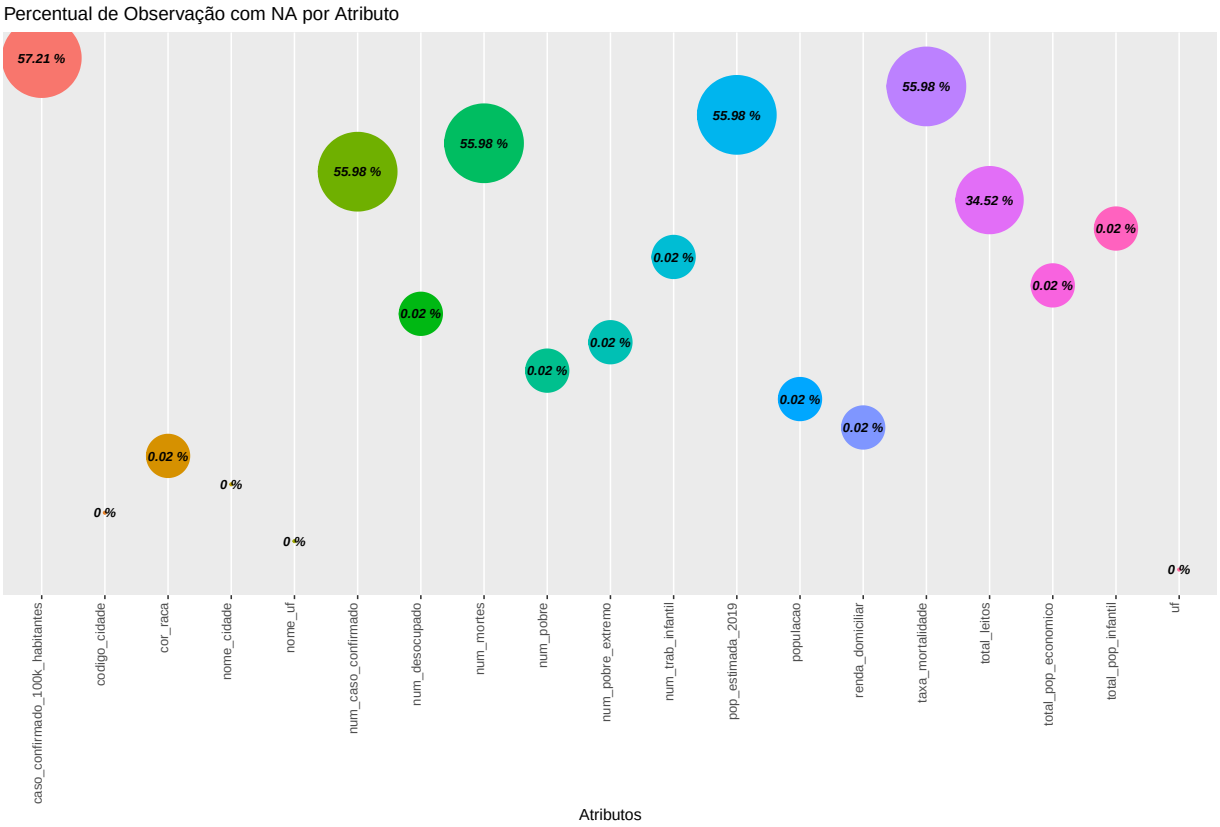
```
na_covid_resumo <-
  na_covid_resumo %>% mutate(linha = 1:19,
                             tamanho = as.integer(if_else(perc_na>0,10+perc_na/10,0)))
na_covid_resumo
```

```
##                          atributo     perc_na linha tamanho
## 1                              uf  0.00000000     1       0
## 2                         nome_uf  0.00000000     2       0
## 3                  codigo_cidade  0.00000000     3       0
## 4                    nome_cidade  0.00000000     4       0
## 5                       cor_raca  0.01968117     5      10
```

```
## 6                 renda_domiciliar  0.01968117     6       10
## 7                       populacao  0.01968117     7       10
## 8                       num_pobre  0.01968117     8       10
## 9                 num_pobre_extremo  0.01968117     9       10
## 10                   num_desocupado  0.01968117    10       10
## 11               total_pop_economico  0.01968117    11       10
## 12                num_trab_infantil  0.01968117    12       10
## 13               total_pop_infantil  0.01968117    13       10
## 14                     total_leitos 34.52469986    14       13
## 15               num_caso_confirmado 55.98110608    15       15
## 16                       num_mortes 55.98110608    16       15
## 17                 pop_estimada_2019 55.98110608    17       15
## 18                  taxa_mortalidade 55.98110608    18       15
## 19 caso_confirmado_100k_habitantes 57.20921079    19       15
```

O atributo num_mortes possui *14.222* observações com *NA*. Isso corresponde a cerca de *55,98%* das observações com *num_morte* com *NA*.

```
na_covid_resumo %>% ggplot(aes(x = as.factor(atributo), y = linha)) +
  geom_point(aes(size = perc_na, stroke = tamanho, color = as.factor(atributo),
                 fill = as.factor(atributo)),
             shape = 19, show.legend = FALSE) +
  geom_text(aes(label = paste(round(perc_na,2),"%")), size = 3,
            fontface = 'bold.italic') +
  scale_y_continuous(breaks = NULL) +
  labs(x = "Atributos", y = "", title = "Percentual de Observação com NA por Atributo") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0))
```



Os atributos *caso_confirmado_100k_habitantes* (Número de casos confirmados por 100k habitantes), *num_caso_confirmado* (Número de casos confirmados), *num_mortes* (Número de mortes), *pop_estimada_2019* (População estimada para localidade em 2019) e *taxa_mortalidade* (Taxa de mortalidade) apresentam alto índice de *NA's*, possuindo mais da metade das observações com dados faltantes.

O atributo *total_leitos* (Total de leitos disponíveis pelo SUS) apresenta um percentual de *NA's* ligeiramente mais baixo, ficando em torno de *30%*.

Os demais atributos encontram-se com os dados praticamente completos.

Como há o interesse de avaliar como o Número de mortes se relaciona com os demais atributo, o fato do *Data Set* possuir cerca de *50%* dos dados para esse atributo faltantes, isso prejudica essa análise. Portanto, optamos por retirar todas as observações de *num_morte* que contenha *NA*. Restando um *Data Set* com *11.183* observações. Prosseguiremos a análise com esse *Data Set* reduzido, mas sem observações com *NA* no atributo *num_mortes*.

Esse novo *Data Set* corresponde a *44,02%* (100-55,98) do *Data Set* original.

```
nrow(na.omit(covid_cidade_ibge))
```

```
## [1] 9027
```

```
covid_tratado <- covid_cidade_ibge[!is.na(num_mortes),]
nrow(covid_cidade_ibge)
```

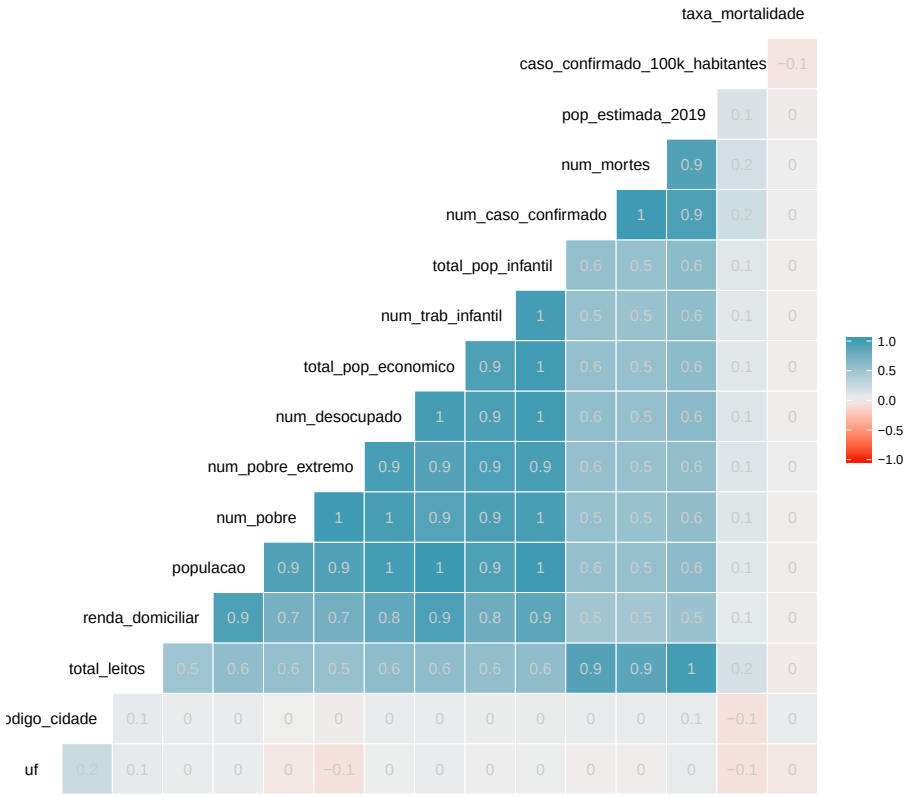
```
## [1] 25405
```

```
nrow(covid_tratado)
```

```
## [1] 11183
```

Gráfico com a correlação.

```
ggcorr(covid_tratado[,-c('nome_uf','nome_cidade','cor_raca')],
       geom = 'tile',
       low = "#F21A00",
       high = "#3B9AB2",
       label = TRUE,
       label_size = 4,
       hjust = 0.9,
       label_color = 'gray80')
```



O gráfico mostra uma alta correlação entre os atributos *renda_domiciliar*, *populacao*, *num_pobre*, *num_pobre_extremo*, *num_desocupado*, *total_pop_economico*, *num_trab_infantil* e *total_pop_infantil*.

Também temos uma alta correlação entre *num_mortes*, *num_caso_confirmado*, *pop_estimada_2019* e *total_leitos*.

Não se nota uma correlação relevante, está em torno de *60%*, entre *num_mortes* e os atributos que possam caracterizar pobreza como os atributos *num_pobre*, *num_pobre_extremo*, *num_desocupado* e *num_trab_infantil*.

Eliminando o atributo *cor_raca* e separando as observações entre população branca e demais, construímos um novo *Data Set* em que o atributo *cor_raca* é eliminado, restando agora somente população branca e não branca.

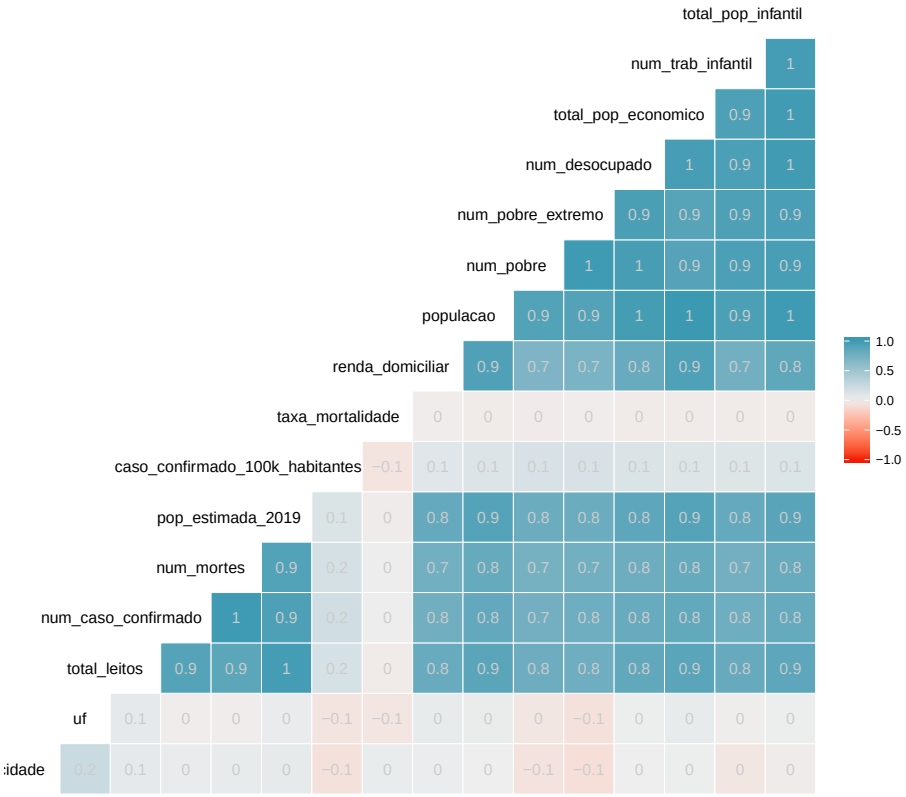
```
covid_tratado[,eh_branco := if_else(covid_tratado$cor_raca == 'Branco',TRUE,FALSE)]
covid_tratado2 <- covid_tratado[,.(nome_cidade = nome_cidade,
                                   uf = uf,
                                   nome_uf = nome_uf,
                                   total_leitos = total_leitos,
                                   num_caso_confirmado = num_caso_confirmado,
                                   num_mortes = num_mortes,
                                   pop_estimada_2019 = pop_estimada_2019,
                                   caso_confirmado_100k_habitantes = caso_confirmado_100k_habitantes,
                                   taxa_mortalidade = taxa_mortalidade,
                                   renda_domiciliar = sum(renda_domiciliar),
                                   populacao = sum(populacao),
                                   num_pobre = sum(num_pobre),
                                   num_pobre_extremo = sum(num_pobre_extremo),
                                   num_desocupado = sum(num_desocupado),
```

```
                                   total_pop_economico = sum(total_pop_economico),
                                   num_trab_infantil = sum(num_trab_infantil),
                                   total_pop_infantil = sum(total_pop_infantil)),
                                 by = .(codigo_cidade,eh_branco)] %>% unique
str(covid_tratado2)
```

```
## Classes 'data.table' and 'data.frame':   4706 obs. of  19 variables:
##  $ codigo_cidade                  : int  15 15 23 23 31 31 49 49 98 98 ...
##  $ eh_branco                      : logi  TRUE FALSE TRUE FALSE TRUE FALSE ...
##  $ nome_cidade                    : chr  "Alta Floresta D'Oeste" "Alta Floresta D'Oeste" "Ariquemes"
##  $ uf                             : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ nome_uf                        : chr  "Rondônia" "Rondônia" "Rondônia" "Rondônia" ...
##  $ total_leitos                   : int  49 49 240 240 9 9 360 360 54 54 ...
##  $ num_caso_confirmado            : int  1 1 89 89 0 0 2 2 1 1 ...
##  $ num_mortes                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pop_estimada_2019              : int  22945 22945 107863 107863 5312 5312 85359 85359 32374 32374
##  $ caso_confirmado_100k_habitantes: num  4.36 4.36 82.51 82.51 NA ...
##  $ taxa_mortalidade               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ renda_domiciliar               : num  6000725 5269952 26759062 33641644 1665509 ...
##  $ populacao                      : int  10482 13615 31842 57924 2910 3393 32635 45506 13211 15299 .
##  $ num_pobre                      : int  4831 6837 7899 19974 1156 1880 9179 15905 4606 6064 ...
##  $ num_pobre_extremo              : int  2804 3492 3446 8076 379 921 4536 6976 2093 2322 ...
##  $ num_desocupado                 : int  181 322 686 1352 33 36 892 1372 223 418 ...
##  $ total_pop_economico            : int  4451 5592 15703 28244 1392 1374 16449 22121 6653 7566 ...
##  $ num_trab_infantil              : int  153 175 323 793 49 44 423 820 317 456 ...
##  $ total_pop_infantil             : int  1127 1779 3714 7272 338 379 3189 5631 1300 2052 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Gráfico com a correlação utilizando o novo *Data Set*.

```
ggcorr(covid_tratado2[,-c('nome_uf','nome_cidade','eh_branco')],
       geom = 'tile',
       low = "#F21A00",
       high = "#3B9AB2",
       label = TRUE,
       label_size = 4,
       hjust = 0.9,
       label_color = 'gray80')
```



Com o novo Data Set já percebemos que houve um aumento entre a correlação entre o atributo *num_mortes* e os atributos *num_pobre*, *num_pobre_extremo*, *num_desocupado* e *num_trab_infantil*.

O tratamento aplicado ao *Data Set* teve como base o fato que a pobreza no país possui cor, ou seja, a maioria da população pobre é não branca. Ao eliminarmos o atributo *cor_raca*, agregarmos os atributos que caracterizam a probleza nesse *Data Set* (*num_pobre*, *num_pobre_extremo*, *num_desocupado* e *num_trab_infantil*). em dois tipos branco e não branco.

Conforme esperado, notamos que o atributo num_mortes está altamente correlacionado com os atributos que caracterizam a pobreza nesse *Data Set*.

Percebemos que essa doença, faz mais vítimas entre a população mais pobre do país.

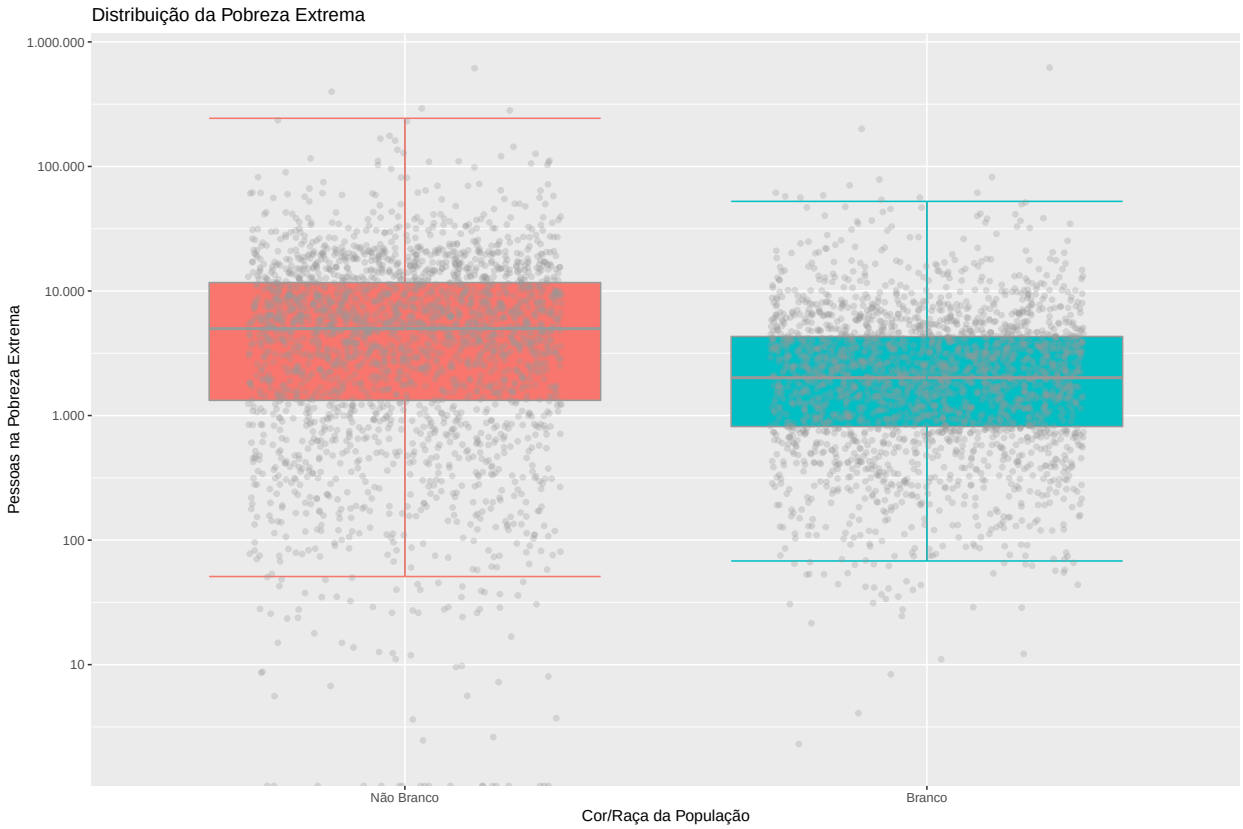
```
summary(covid_tratado2)
```

```
##  codigo_cidade   eh_branco       nome_cidade              uf
##  Min.   :   13   Mode :logical   Length:4706        Min.   :11.00
##  1st Qu.: 3605   FALSE:2352      Class :character   1st Qu.:23.00
##  Median : 8107   TRUE :2352      Mode  :character   Median :31.00
##  Mean   :13394   NA's :2                            Mean   :30.69
##  3rd Qu.:17600                                      3rd Qu.:35.00
##  Max.   :71303                                      Max.   :53.00
##
##    nome_uf           total_leitos     num_caso_confirmado   num_mortes
##  Length:4706        Min.   :    1.0   Min.   :    0.00    Min.   :   0.000
##  Class :character   1st Qu.:   25.0   1st Qu.:    1.00    1st Qu.:   0.000
##  Mode  :character   Median :   50.0   Median :    3.00    Median :   0.000
##                     Mean   :  194.9   Mean   :   43.21    Mean   :   3.004
```

```
##                    3rd Qu.:  120.0   3rd Qu.:   10.00    3rd Qu.:   1.000
##                    Max.   :27847.0   Max.   :19822.00    Max.   :1673.000
##                    NA's   :872
##  pop_estimada_2019  caso_confirmado_100k_habitantes taxa_mortalidade
##  Min.   :    1149   Min.   :  0.6433               Min.   :0.00000
##  1st Qu.:   12568   1st Qu.:  7.4351               1st Qu.:0.00000
##  Median :   25228   Median : 15.3304               Median :0.00000
##  Mean   :   75099   Mean   : 30.3014               Mean   :0.08078
##  3rd Qu.:   54772   3rd Qu.: 33.7154               3rd Qu.:0.06060
##  Max.   :12252023   Max.   :914.7337               Max.   :1.00000
##                     NA's   :136
##  renda_domiciliar      populacao         num_pobre       num_pobre_extremo
##  Min.   :6.734e+03   Min.   :     12   Min.   :      0   Min.   :     0
##  1st Qu.:2.351e+06   1st Qu.:   6239   1st Qu.:   2426   1st Qu.:   938
##  Median :6.787e+06   Median :  15982   Median :   6532   Median :  2922
##  Mean   :3.794e+07   Mean   :  46574   Mean   :  15345   Mean   :  7004
##  3rd Qu.:1.976e+07   3rd Qu.:  37430   3rd Qu.:  14916   3rd Qu.:  7108
##  Max.   :1.250e+10   Max.   :6779147   Max.   :1177224   Max.   :545291
##  NA's   :2           NA's   :2         NA's   :2         NA's   :2
##  num_desocupado     total_pop_economico num_trab_infantil  total_pop_infantil
##  Min.   :     0.0   Min.   :      5     Min.   :    0.00   Min.   :     0.0
##  1st Qu.:   154.0   1st Qu.:   2706     1st Qu.:   75.75   1st Qu.:   717.5
##  Median :   450.5   Median :   7088     Median :  191.00   Median :  1874.5
##  Mean   :  1755.6   Mean   :  22624     Mean   :  426.76   Mean   :  5043.1
##  3rd Qu.:  1223.5   3rd Qu.:  17009     3rd Qu.:  434.00   3rd Qu.:  4395.0
##  Max.   :223276.0   Max.   :3593339     Max.   :35721.00   Max.   :562964.0
##  NA's   :2          NA's   :2           NA's   :2          NA's   :2
```

Após a eliminação das observações em que o atributo *num_mortes* apresentavam *NA's*, notamos que ainda restam bastante observação com *NA's* no atributo *total_leitos* e *caso_confirmado*.

```
covid_tratado2 %>% ggplot(aes(x = eh_branco, y = num_pobre_extremo)) +
  geom_boxplot(aes(fill = as.factor(eh_branco)), outlier.alpha = 0, color = 'gray60',
               show.legend = FALSE) +
  stat_boxplot(geom = 'errorbar', aes(colour = as.factor(eh_branco)),
               show.legend = FALSE) +
  geom_jitter(width = 0.3, height = 0.1, color = 'gray60', alpha = 0.3) +
  scale_y_log10(breaks = c(10,1e02,1e03,1e04,1e05,1e06),
                label = c('10','100','1.000','10.000','100.000','1.000.000')) +
  scale_x_discrete(na.translate = FALSE, breaks = c(FALSE,TRUE),
                   label = c('Não Branco','Branco')) +
  labs(x = 'Cor/Raça da População', y = 'Pessoas na Pobreza Extrema',
       title = "Distribuição da Pobreza Extrema")
```

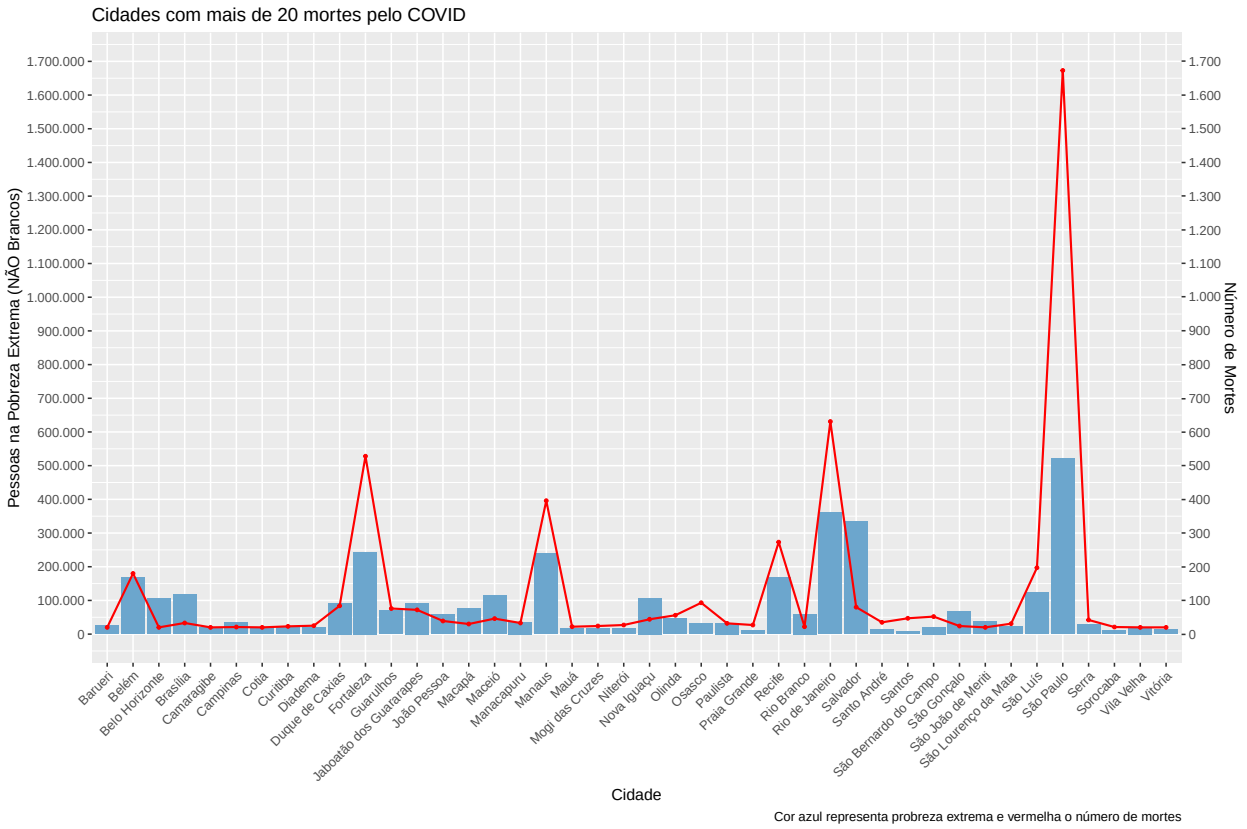


Esse gráfico retrata a diferença na quantidade de possoas que estão na probreza extrema entre *Brancos* e *Não Brancos*. Nota-se que existem muito mais Não Brancos na pobreza extrema que Branco nas cidades brasileiras.

Com base nessa evidência, pegamos um subconjunto do *Data Set* em que o foco é nas pessoas *Não Brancas*. Verificamos o comportamento dos atributos *num_mortes* e *num_pobre_extremo* associados por cidade.

```
covid_tratado2[num_mortes >= 20 & eh_branco == FALSE,] %>%
  ggplot(aes(x = nome_cidade, y = num_pobre_extremo)) +
  geom_col(fill = 'skyblue3') +
  geom_line(aes(x = as.integer(as.factor(nome_cidade)), y = num_mortes*1e03),
            colour = '#FF0000', size = 0.7) +
  geom_point(aes(y = num_mortes*1e03), colour = '#FF0000', shape = 20) +
  scale_y_continuous(limits = c(0,17e05),
                     breaks = c(0,1e05,2e05,3e05,4e05,5e05,6e05,7e05,8e05,9e05,
                                10e05,11e05,12e05,13e05,14e05,15e05,16e05,17e05),
                     labels = c('0','100.000','200.000','300.000','400.000','500.000',
                                '600.000','700.000','800.000','900.000','1.000.000',
                                '1.100.000','1.200.000','1.300.000','1.400.000',
                                '1.500.000','1.600.000','1.700.000'),
                     sec.axis = sec_axis(~ ./1e03,
                                         breaks = c(0,1e02,2e02,3e02,4e02,5e02,6e02,7e02,
                                                    8e02,9e02,10e02,11e02,12e02,13e02,
                                                    14e02,15e02,16e02,17e02),
                                         labels = c('0','100','200','300','400','500',
                                                    '600','700','800','900','1.000',
                                                    '1.100','1.200','1.300','1.400',
                                                    '1.500','1.600','1.700'),
```

```
                                    name = 'Número de Mortes')) +
  labs(x = 'Cidade', y = 'Pessoas na Pobreza Extrema (NÃO Brancos)',
       title = 'Cidades com mais de 20 mortes pelo COVID',
       caption = 'Cor azul representa probreza extrema e vermelha o número de mortes') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



O gráfico confirma visualmente uma relação entre num_pobre_extrema e num_mortes apresentada pela matriz de correlação, ou seja, em cidades onde o número de pessoas na pobreza extrema é mais alto apresenta um número de mortes pelo COVID mais elevado. No gráfico podemos ver as barras em azul que representam o número de pessoas na situação de pobreza extrema com escala no eixo vertical à esquerda e a linha vermelha que representa o número de mortes pelo COVID com escala no eixo vertical à direita.

Redução de Dimensionalidade do *Data Set*

Na tentativa de identificar padrões ocultos, remover o ruído e redundância no *Data Set*, iniciamos o procedimento para uma Análise por Componentes Principais (*PCA - Principal Component Analysis*).

```
ds_covid <- covid_tratado2 %>% na.omit
pca <- ds_covid[,-c('nome_cidade','nome_uf')] %>% prcomp(scale = TRUE)

summary(pca)
```

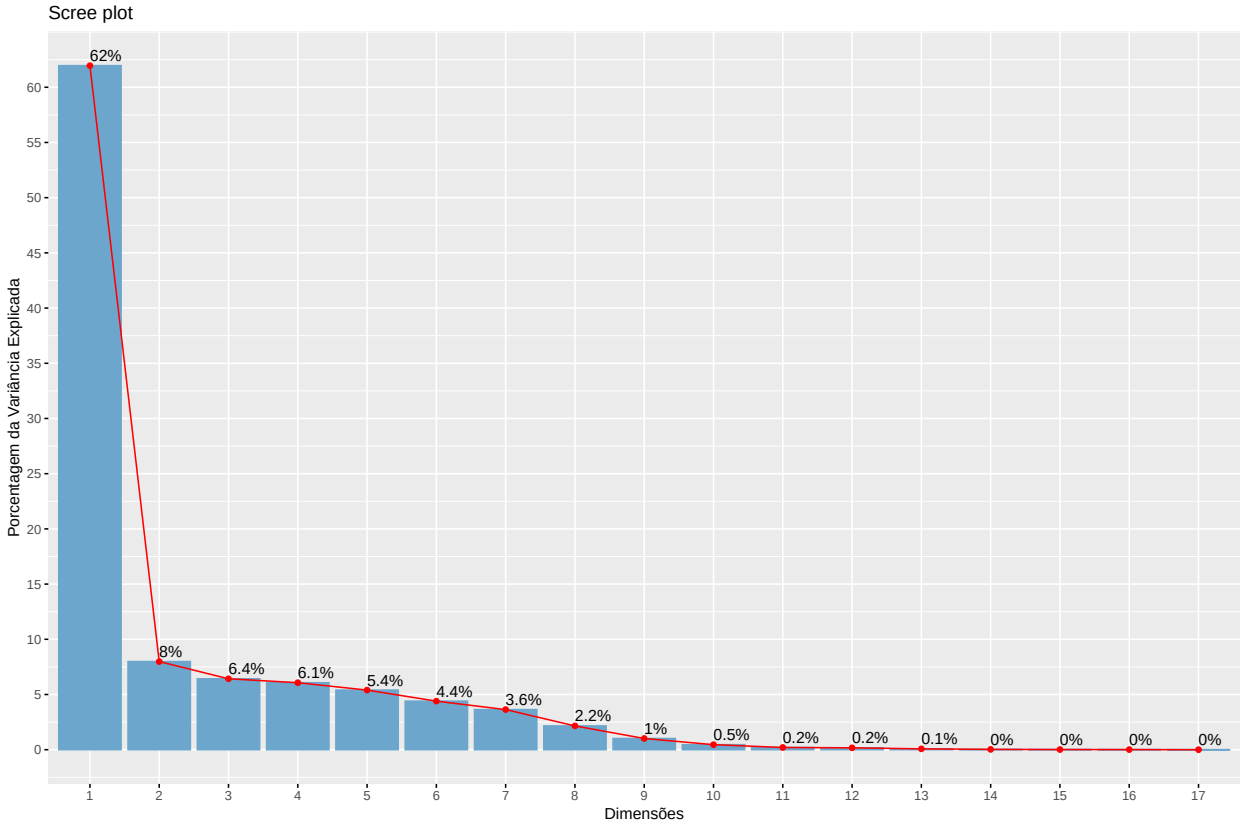
```
## Importance of components:
##                           PC1     PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     3.2454 1.16492 1.04498 1.0158 0.95807 0.86475 0.78605
```

```
## Proportion of Variance 0.6196 0.07983 0.06423 0.0607 0.05399 0.04399 0.03635
## Cumulative Proportion  0.6196 0.69940 0.76364 0.8243 0.87833 0.92232 0.95866
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.60541 0.41533 0.27856 0.18538 0.17081 0.11474 0.06936
## Proportion of Variance 0.02156 0.01015 0.00456 0.00202 0.00172 0.00077 0.00028
## Cumulative Proportion  0.98022 0.99037 0.99494 0.99696 0.99867 0.99945 0.99973
##                           PC15    PC16    PC17
## Standard deviation     0.05072 0.04251 0.01393
## Proportion of Variance 0.00015 0.00011 0.00001
## Cumulative Proportion  0.99988 0.99999 1.00000
```

```
get_eigenvalue(pca)
```

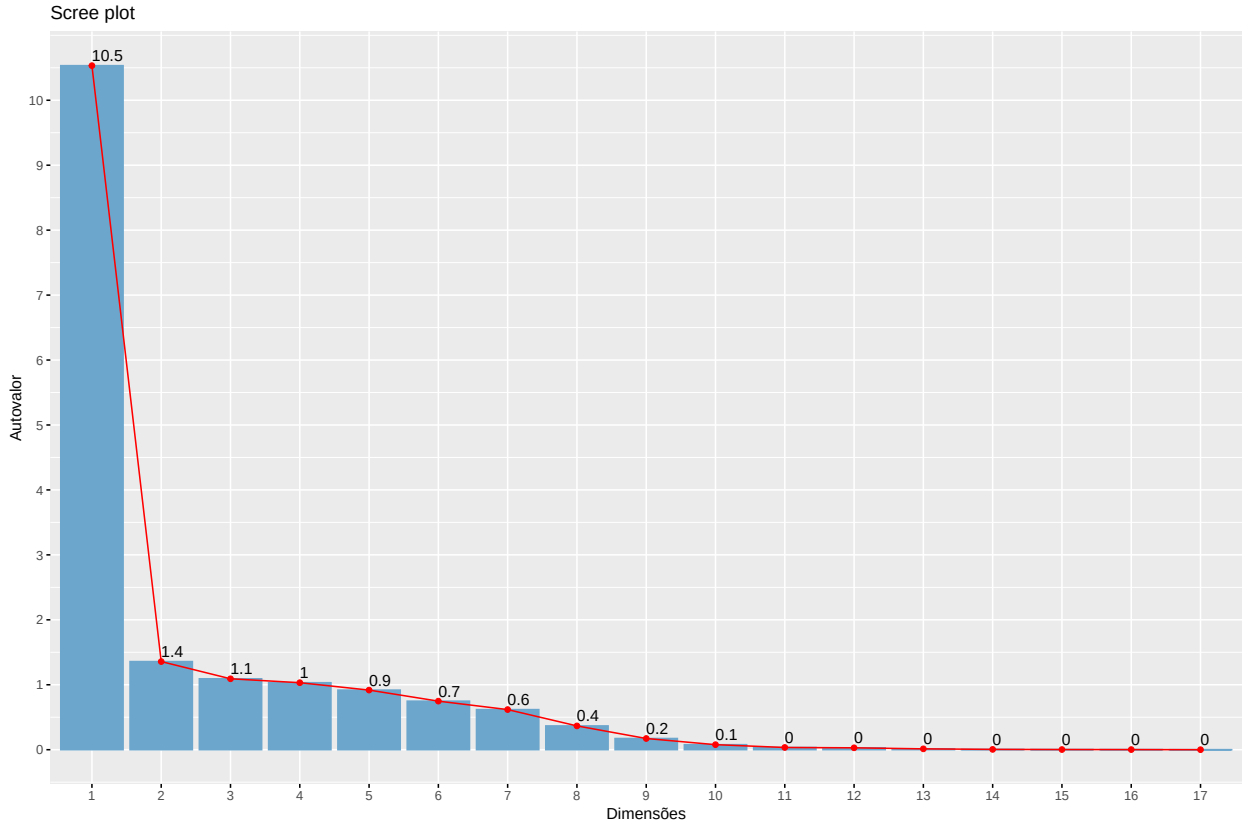
```
##          eigenvalue variance.percent cumulative.variance.percent
## Dim.1  1.053283e+01     61.957818727                    61.95782
## Dim.2  1.357042e+00      7.982597385                    69.94042
## Dim.3  1.091979e+00      6.423403018                    76.36382
## Dim.4  1.031867e+00      6.069806157                    82.43363
## Dim.5  9.178977e-01      5.399398492                    87.83302
## Dim.6  7.477997e-01      4.398821578                    92.23185
## Dim.7  6.178795e-01      3.634585464                    95.86643
## Dim.8  3.665198e-01      2.155998554                    98.02243
## Dim.9  1.724986e-01      1.014697402                    99.03713
## Dim.10 7.759488e-02      0.456440478                    99.49357
## Dim.11 3.436656e-02      0.202156247                    99.69572
## Dim.12 2.917700e-02      0.171629390                    99.86735
## Dim.13 1.316427e-02      0.077436859                    99.94479
## Dim.14 4.811270e-03      0.028301588                    99.97309
## Dim.15 2.572898e-03      0.015134697                    99.98823
## Dim.16 1.807444e-03      0.010632022                    99.99886
## Dim.17 1.941302e-04      0.001141943                   100.00000
```

```
fviz_eig(pca, addlabels = TRUE, choice = 'variance',
         barfill = 'skyblue3', barcolor = 'skyblue3', linecolor = 'red',
         ncp = nrow(pca$rotation), xlab = "Dimensões",
         ylab = "Porcentagem da Variância Explicada",
         ggtheme = theme_gray(), yticks.by = 5)
```



De acordo com o gráfico em função da variância acumulada, vemos que o número de dimensões ideal está entre 5 e 7, pois nesse intervalo temos uma variância explicada entre *87,83%* e *95,87%*. Com 5 dimensões explicamos *87,83%* da variância, com 6 explicamos *92,23%* e 7 explicamos *95,87%*.

```
fviz_eig(pca, addlabels = TRUE, choice = 'eigenvalue',
         barfill = 'skyblue3', barcolor = 'skyblue3', linecolor = 'red',
         ncp = nrow(pca$rotation), xlab = "Dimensões", ylab = "Autovalor",
         ggtheme = theme_gray(), yticks.by = 1)
```



Olhando para o gŕafico em função dos autovalores e seguindo o critério da *Raiz Latente* ou *Kaiser*, que descarta dimensões com Autovalor menor que 1, vemos que o número ideal de dimensões estaria entre 4 e 5. Apesar da dimensão 5 ser menor que 1, seu Autovalor está muito próximo de 1 (0,9179), assim foi considerado também como ponto de corte.

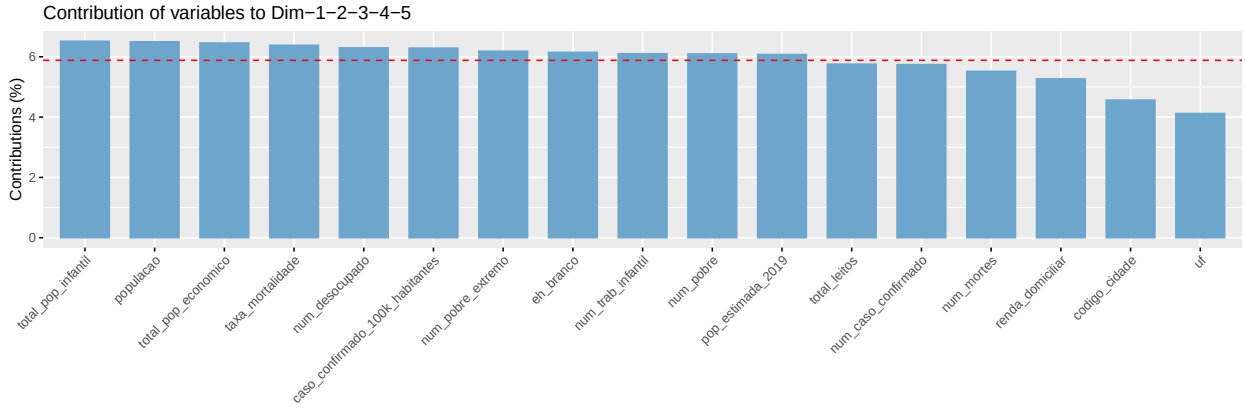
O critério da *Raiz Latente* ou critério de *Kaiser* estabelece que o número de dimensões deve ser aquela em que o número de autovalores seja maior ou igual à média das variâncias das variáveis analisadas. As variáveis estando padronizadas, todas possuem variância igual a 1, portanto o critério descarta dimensões que estejam com Autovalor menor que 1. Esse critério faz sentido se pensarmos que Autovalor menor que 1 não explica nem ao menos a variância de uma variável isolada. Portanto, não deve fazer parte do conjunto.

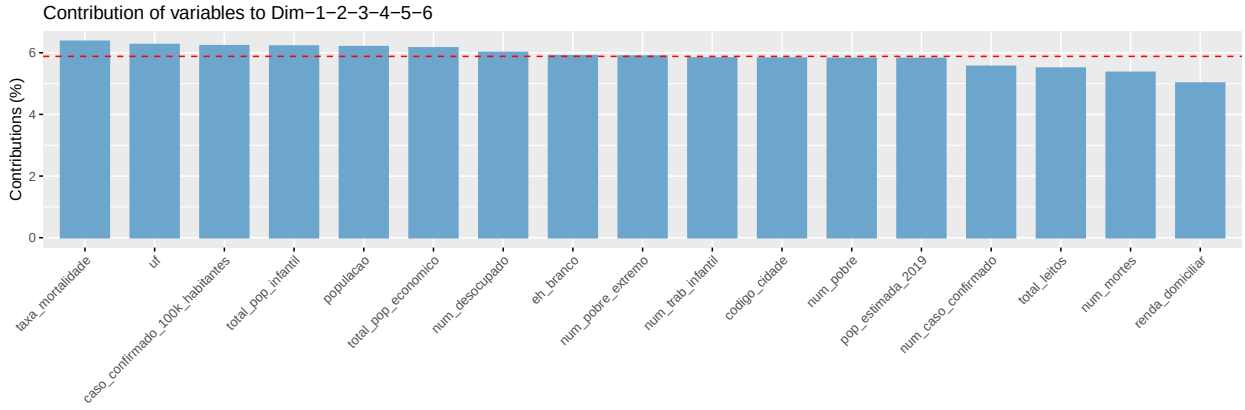
Analisando os dois critérios, optamos pelo corte do PCA em 5 dimensões e, também testaremos um corte com 6 dimensões para comparação.

```
# contribuição das variáveis nas 5 primeiras cargas
g1 <- fviz_contrib(pca, choice = 'var', axes = 1:5,
                   fill = 'skyblue3', color = 'skyblue3', ggtheme = theme_gray(),
                   yticks.by = 2)

# contribuição das variáveis nas 6 primeiras cargas
g2 <- fviz_contrib(pca, choice = 'var', axes = 1:6,
                   fill = 'skyblue3', color = 'skyblue3', ggtheme = theme_gray(),
                   yticks.by = 2)

grid.arrange(g1,
             g2,
             nrow = 2,
             ncol = 1)
```





Percebemos uma mudança na contribuição das primeiras variáveis quando cortamos em 5 dimensões e em 6. A variável *uf* passou de última em 5 dimensões para ser a segunda com 6 dimensões. Assim como *codigo_cidade* que tornou-se mais relevante com 6 dimensões. Isso mostra uma contribuição da localização mais acentuada quando passamos a ter 6 dimensões.

```
pca_5dim <- pca$x[,1:5]
pca_6dim <- pca$x[,1:6]
```

Agrupamento

Utilizando o Data Set com 5 Dimensoes.

Padronizando o Data Set.

```
pca_5dim_std <- scale(pca_5dim)
```

Calculando as medidas de dissimilaridade.

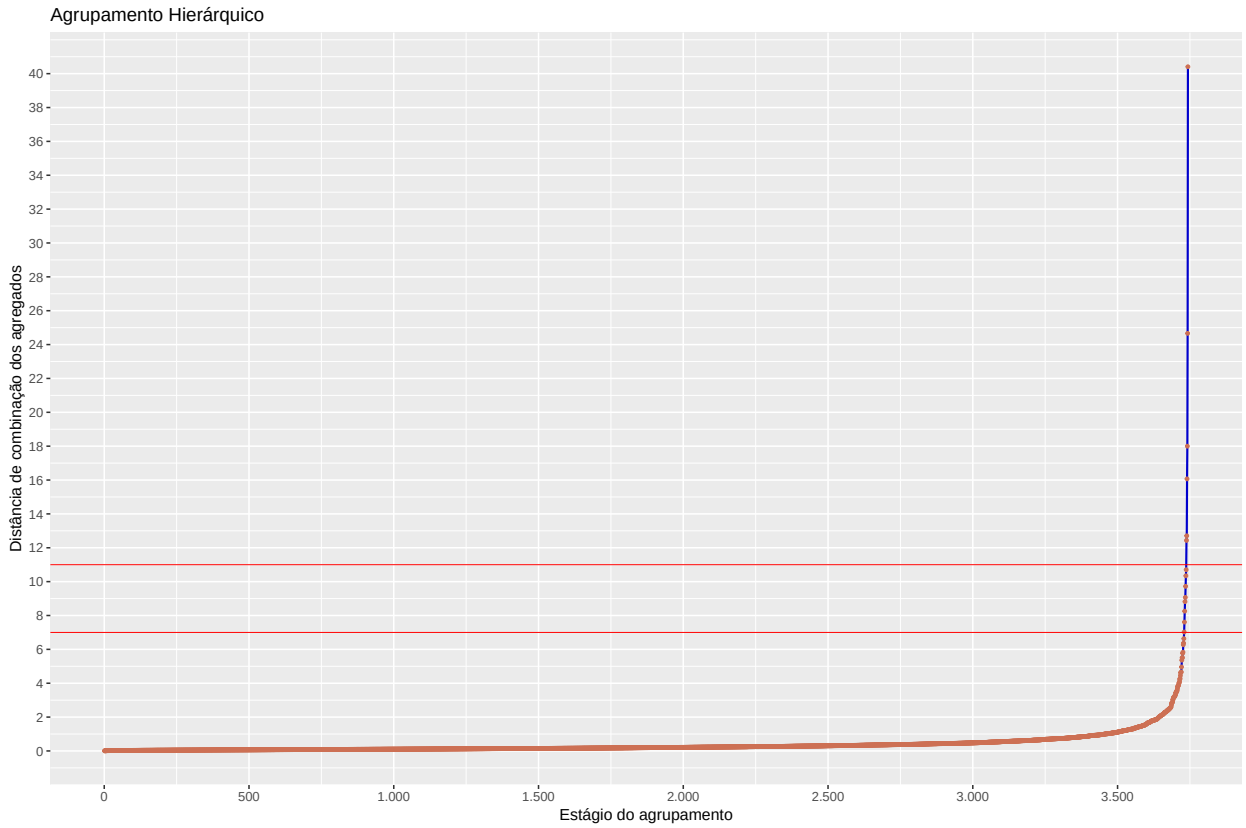
```
dist_5dim <- dist(pca_5dim_std, method = "euclidean")
```

Calcualndo o agrupamento pelo modelo *Hierárquico*.

```
set.seed(314)
hier_5dim_comp <- hclust(dist_5dim, method = 'complete')
```

```r
hier_height <- data.frame(stage = 1:length(hier_5dim_comp$height),
                          height = hier_5dim_comp$height,
                          cum_height = cumsum(hier_5dim_comp$height))

# Gráfico da distância de combinação dos agregados no agrupamento hierárquico
hier_height %>% ggplot(aes(x = stage, y = height)) +
  geom_line(color = 'blue3', size = 0.7) +
  geom_point(color = 'salmon3', shape = 20) +
  geom_hline(yintercept = 7, color = 'red', size = 0.1) +
  geom_hline(yintercept = 11, color = 'red', size = 0.1) +
  scale_y_continuous(breaks = c(0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40),
                     labels = c('0','2','4','6','8','10','12','14','16','18','20',
                                '22','24','26','28','30','32','34','36','38','40')) +
  scale_x_continuous(breaks = c(0,500,1000,1500,2000,2500,3000,3500),
                     labels = c('0','500','1.000','1.500','2.000','2.500','3.000',
                                '3.500')) +
  labs(x = 'Estágio do agrupamento', y = 'Distância de combinação dos agregados',
       title = 'Agrupamento Hierárquico')
```



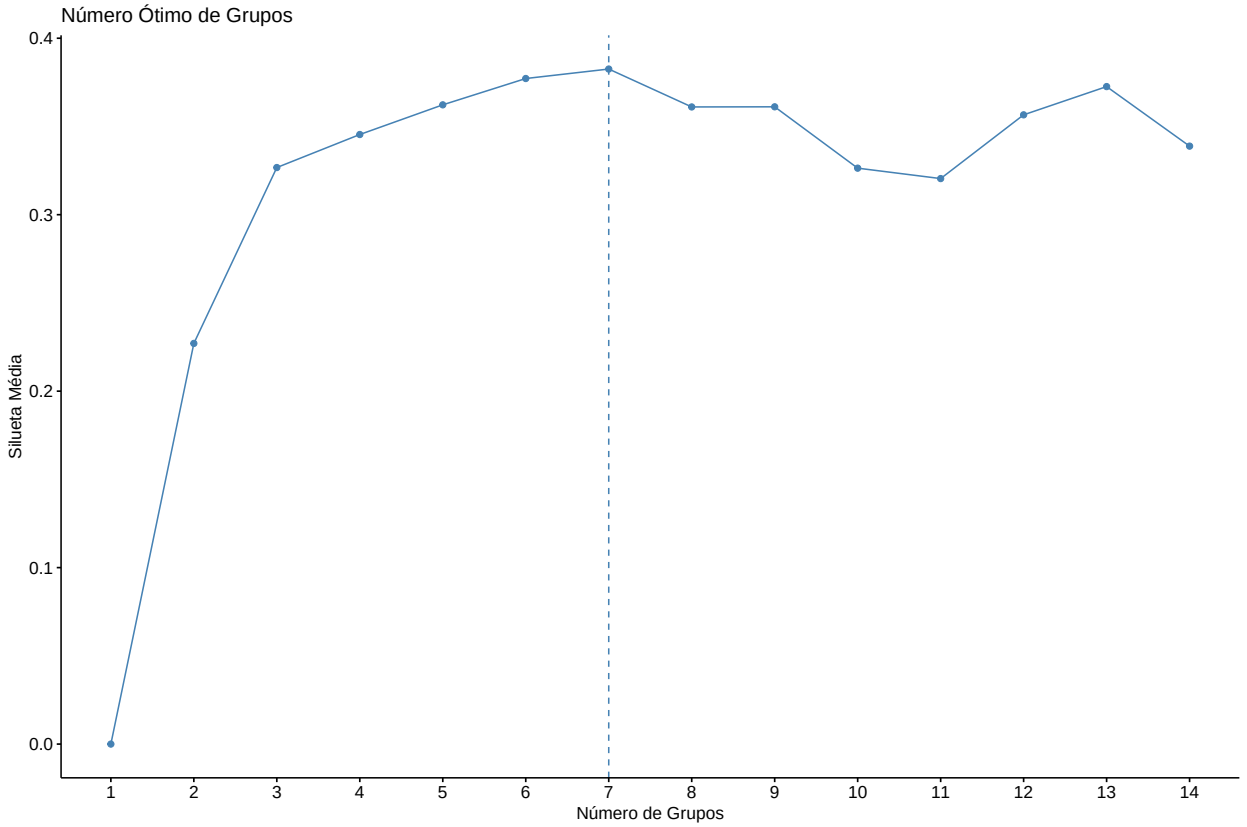
O gráfico mostra que entre as distâncias de 7 e 11 temos uma maior separação entre os pontos que representam os estágios de agrupamento, indicando que o estágio desse agrupamento foi feito entre agregados que tinham pouca similaridade. Assim, se tivéssemos que escolher um ponto para podar a árvore hierárquica do agrupamento, faríamos entre esses pontos.

```r
cutpoint <- hier_height %>% filter(height >= 7 & height <= 11)
grupos <- t(unique(cutree(hier_5dim_comp, h = cutpoint[,"height"])))
apply(grupos, c(1), max)
```

```
## 7.02639498165588 7.61075608071052 8.25626862290339  8.8170158806144
##               14               13               12               11
## 9.06798783158662 9.72280655774466 10.3426989710642 10.7035987485033
##               10                9                8                7
```

Percebemos que entre as distâncias de combinação de agregados no intervalo de 7 a 11, geram 14 a 7 grupos no agrupamento hierárquico. Portanto, iremos avaliar o k-Means nesse intervalo de número de grupos.

```
# Gráfico apontando a escolha da quantidade de grupos com k-Means
set.seed(314)
fviz_nbclust(pca_5dim_std, kmeans, method = 'silhouette', k.max = 14,
             diss = dist_5dim) +
  labs(x = 'Número de Grupos', y = 'Silueta Média', title = 'Número Ótimo de Grupos')
```



Analisando o gráfico com a simulação do k-Means para o número de grupos de 2 até 14, verficamos que o número de grupos em que a silueta atinge o melhor valor é quando temos 7 grupos. Este valor está dentro do primeiro filtro realizado com o agrupamento hierárquico.

A métrica silueta avalia o quanto um grupo está coeso, quanto maior o seu valor melhor a qualidade do agrupamento.

```
# Ajustando o processamento paralelo em CORES (NÚCLEOS)
cl <- makePSOCKcluster(8)
registerDoParallel(cl)
```

```r
clmethods <- c("hierarchical","kmeans","pam")
arquivo <- '../model/Cluster_5dim.rds'
set.seed(314)
cluster_5dim <- clValid(pca_5dim_std, nClust = 5:15, maxitems = nrow(pca_5dim_std),
                        clMethods = clmethods, method = 'complete',
                        validation = "internal")
saveRDS(cluster_5dim, file = arquivo)

arquivo <- '../model/Cluster_5dim2.rds'
set.seed(314)
cluster_5dim2 <- clValid(pca_5dim_std, nClust = 5:15, maxitems = nrow(pca_5dim_std),
                         clMethods = 'hierarchical', method = 'ward',
                         validation = "internal")
saveRDS(cluster_5dim, file = arquivo)

# Resumo
summary(cluster_5dim)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  5 6 7 8 9 10 11 12 13 14 15
##
## Validation Measures:
##                                   5        6        7        8        9       10       11       12
##
## hierarchical Connectivity   16.4663  19.5813  74.2774  76.7544  78.7544 120.8944 124.7524 128.2448 1
##              Dunn            0.1245   0.1272   0.0327   0.0338   0.0360   0.0291   0.0299   0.0320
##              Silhouette      0.7408   0.6929   0.4733   0.4491   0.4489   0.3652   0.3654   0.3573
## kmeans       Connectivity   90.4821 117.1778 132.5464 163.1250 165.1250 159.1377 185.1294 193.6726 2
##              Dunn            0.0140   0.0148   0.0191   0.0191   0.0191   0.0254   0.0200   0.0212
##              Silhouette      0.3392   0.3407   0.3447   0.3858   0.3855   0.3447   0.3437   0.3437
## pam          Connectivity  273.8369 322.9163 536.4889 707.7524 698.4127 745.7016 789.5524 735.5730 7
##              Dunn            0.0016   0.0014   0.0012   0.0012   0.0012   0.0012   0.0011   0.0011
##              Silhouette      0.3516   0.3750   0.2860   0.2041   0.2164   0.2466   0.2365   0.2744
##
## Optimal Scores:
##
##              Score   Method       Clusters
## Connectivity 16.4663 hierarchical 5
## Dunn          0.1272 hierarchical 6
## Silhouette    0.7408 hierarchical 5
```

```r
summary(cluster_5dim2)
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
```

```
##  5 6 7 8 9 10 11 12 13 14 15
##
## Validation Measures:
##                                  5        6        7        8        9       10       11       12
##
## hierarchical Connectivity  193.9127 220.7960 264.7675 282.3552 323.3873 386.6274 478.3528 479.7635 5
##              Dunn            0.0016   0.0018   0.0018   0.0019   0.0019   0.0019   0.0019   0.0019
##              Silhouette      0.3097   0.3481   0.3041   0.3343   0.3259   0.2720   0.2384   0.2435
##
## Optimal Scores:
##
##              Score    Method       Clusters
## Connectivity 193.9127 hierarchical 5
## Dunn           0.0041 hierarchical 15
## Silhouette     0.3481 hierarchical 6
```

```r
# Parando o processamento paralelo
stopCluster(cl)
```

Utilizando como métrica balizadora a *Silueta*, pecebe-se que o modelo *hierárquico* utilizando o método para medir a distÂncia entre grupos *complete* se sai melhor com 5 grupos. O *k-Means* atinge seu auge com 8 ou 9 grupos e o *PAM (Partitioning Around Medoids)* com 6 grupos. O *hierárquico* usando o método para medir a distância entre grupos *ward* tem seu auge com 6 grupos mais obtendo um valor de métrica muito inferior ao *hierárquico* usando *complete*. De todos o *hierárquico* foi o que se saiu melhor, seguido pelo *k-Means*, mas este ficou longe da performance do *hierárquico*.

Utilizando o Data Set com 6 Dimensões.

Padronizando o *Data Set*.

```r
pca_6dim_std <- scale(pca_6dim)
```

Calculando as medidas de dissimilaridade.

```r
dist_6dim <- dist(pca_6dim_std, method = "euclidean")
```

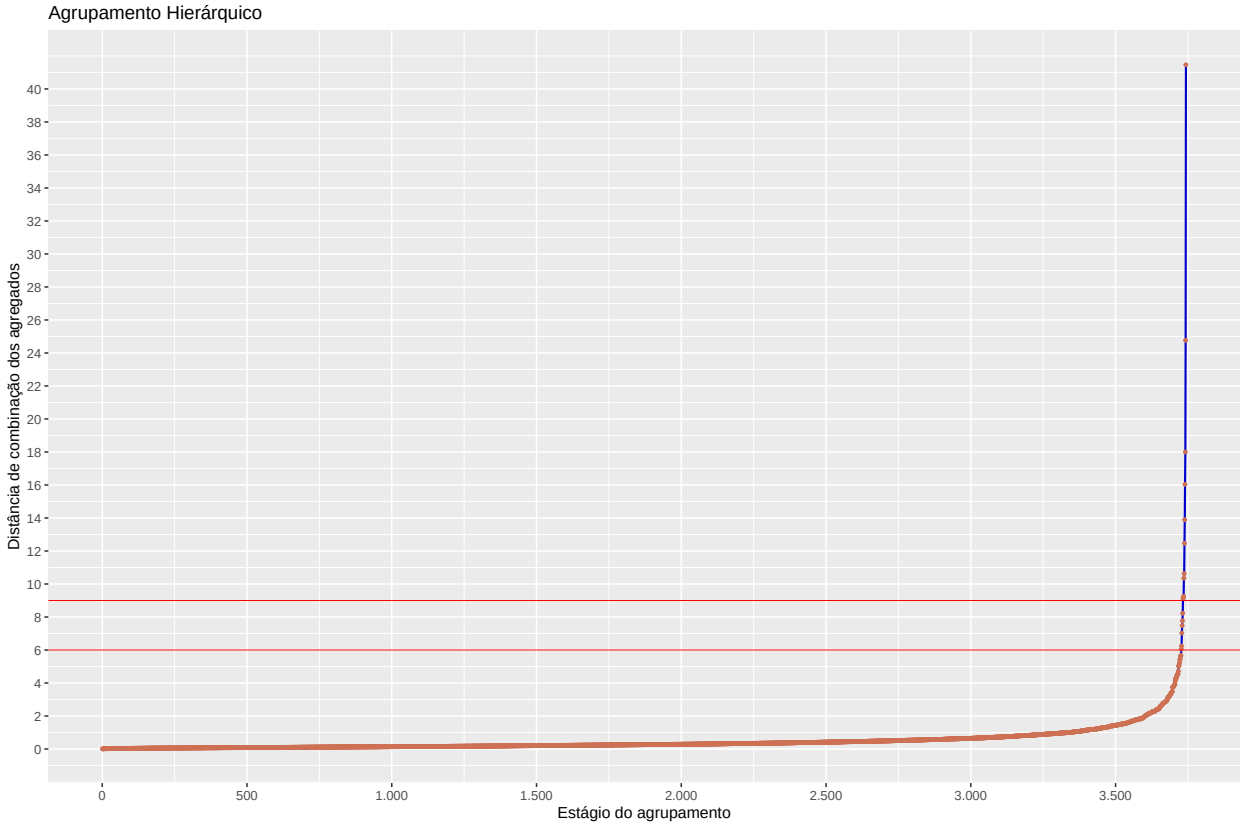
Calcualndo o agrupamento pelo modelo *Hierárquico*.

```r
set.seed(314)
hier_6dim_comp <- hclust(dist_6dim, method = 'complete')

hier_height <- data.frame(stage = 1:length(hier_6dim_comp$height),
                          height = hier_6dim_comp$height,
                          cum_height = cumsum(hier_6dim_comp$height))

# Gráfico da distância de combinação dos agregados no agrupamento hierárquico
hier_height %>% ggplot(aes(x = stage, y = height)) +
  geom_line(color = 'blue3', size = 0.7) +
  geom_point(color = 'salmon3', shape = 20) +
  geom_hline(yintercept = 6, color = 'red', size = 0.1) +
  geom_hline(yintercept = 9, color = 'red', size = 0.1) +
  scale_y_continuous(breaks = c(0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40),
                     labels = c('0','2','4','6','8','10','12','14','16','18','20',
```

```
                                '22','24','26','28','30','32','34','36','38','40')) +
  scale_x_continuous(breaks = c(0,500,1000,1500,2000,2500,3000,3500),
                     labels = c('0','500','1.000','1.500','2.000','2.500','3.000',
                                '3.500')) +
  labs(x = 'Estágio do agrupamento', y = 'Distância de combinação dos agregados',
       title = 'Agrupamento Hierárquico')
```



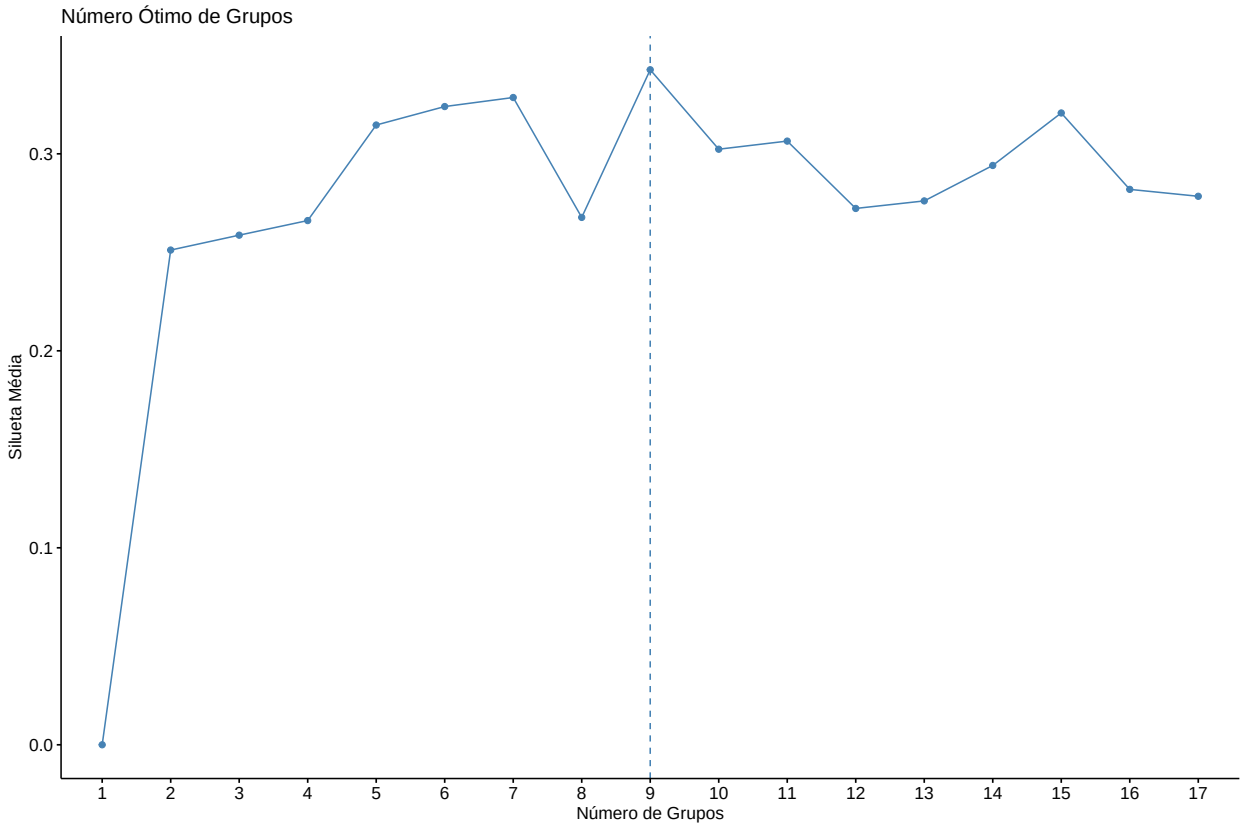
O gráfico mostra que entre as distâncias de 6 e 8 temos uma maior separação entre os pontos que representam os estágios de agrupamento, indicando que o estágio desse agrupamento foi feito entre agregados que tinham pouca similaridade. Assim, se tivéssemos que escolher um ponto para podar a árvore hierárquica do agrupamento, faríamos entre esses pontos.

```
cutpoint <- hier_height %>% filter(height >= 6 & height <= 9)
grupos <- t(unique(cutree(hier_6dim_comp, h = cutpoint[,"height"])))
apply(grupos, c(1), max)
```

```
## 6.05558375091314 6.23936123217352 7.02507071721051 7.47481856419386
##               17               16               15               14
##  7.7725939581485 8.22838990203297
##               13               12
```

Percebemos que entre as distâncias de combinação de agregados no intervalo de 6 a 9, geram 17 a 12 grupos no agrupamento hierárquico. Portanto, iremos avaliar o k-Means nesse intervalo de número de grupos.

```r
# Gráfico apontando a escolha da quantidade de grupos com k-Means
set.seed(314)
fviz_nbclust(pca_6dim_std, kmeans, method = 'silhouette', k.max = 17,
             diss = dist_6dim) +
  labs(x = 'Número de Grupos', y = 'Silueta Média', title = 'Número Ótimo de Grupos')
```



Analisando o gráfico com a simulação do k-Means para o número de grupos de 2 até 17, verficamos que o número de grupos em que a silueta atinge o melhor valor é quando temos 9 grupos. Este valor não está dentro do primeiro filtro realizado com o agrupamento hierárquico.

```r
# Ajustando o processamento paralelo em CORES (NÚCLEOS)
cl <- makePSOCKcluster(8)
registerDoParallel(cl)

clmethods <- c("hierarchical","kmeans","pam")
arquivo <- '../model/Cluster_6dim.rds'
set.seed(314)
cluster_6dim <- clValid(pca_6dim_std, nClust = 7:17, maxitems = nrow(pca_6dim_std),
                        clMethods = clmethods, method = 'complete',
                        validation = "internal")
saveRDS(cluster_6dim, file = arquivo)

arquivo <- '../model/Cluster_6dim2.rds'
set.seed(314)
cluster_6dim2 <- clValid(pca_6dim_std, nClust = 7:17, maxitems = nrow(pca_6dim_std),
                         clMethods = 'hierarchical', method = 'ward',
                         validation = "internal")
```

```r
saveRDS(cluster_6dim, file = arquivo)

# Resumo
summary(cluster_6dim)
```

```
##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  7 8 9 10 11 12 13 14 15 16 17
##
## Validation Measures:
##                                   7        8        9       10       11       12       13       14
##
## hierarchical Connectivity   58.6095  60.6095  67.1992  67.4421  69.0151  72.8730 157.4925 162.0254 1
##              Dunn            0.0542   0.0557   0.0622   0.0629   0.0632   0.0701   0.0280   0.0291
##              Silhouette      0.4713   0.4711   0.4067   0.4274   0.4270   0.4267   0.3043   0.2939
## kmeans       Connectivity  316.0500 318.0500 367.7718 351.4440 269.2687 354.7044 413.3079 310.1262 3
##              Dunn            0.0050   0.0050   0.0050   0.0063   0.0117   0.0085   0.0085   0.0157
##              Silhouette      0.1769   0.1766   0.1840   0.2261   0.3007   0.2266   0.2245   0.3006
## pam          Connectivity  407.9393 419.5179 507.9179 627.1528 652.5060 692.1758 713.9393 709.6151 6
##              Dunn            0.0021   0.0021   0.0019   0.0006   0.0006   0.0006   0.0010   0.0012
##              Silhouette      0.3181   0.3350   0.3226   0.2864   0.2833   0.2547   0.2626   0.2629
##
## Optimal Scores:
##
##              Score   Method       Clusters
## Connectivity 58.6095 hierarchical 7
## Dunn          0.0701 hierarchical 12
## Silhouette    0.4713 hierarchical 7
```

```r
summary(cluster_6dim2)
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  7 8 9 10 11 12 13 14 15 16 17
##
## Validation Measures:
##                                   7        8        9       10       11       12       13       14
##
## hierarchical Connectivity  290.2353 295.0024 341.3504 359.8238 359.8238 440.8726 454.5341 466.8984 4
##              Dunn            0.0025   0.0029   0.0029   0.0029   0.0029   0.0029   0.0029   0.0029
##              Silhouette      0.2711   0.2809   0.2894   0.2629   0.2436   0.2411   0.2202   0.2096
##
## Optimal Scores:
##
##              Score    Method       Clusters
## Connectivity 290.2353 hierarchical 7
```

```
## Dunn           0.0029 hierarchical 8
## Silhouette     0.2894 hierarchical 9
```

```r
# Parando o processamento paralelo
stopCluster(cl)
```

Utilizando a Silueta como métrica balizadora, vemos que o modelo *Hierárquico* se sai melhor com 7 grupos, o *k-Means* melhora com o aumento dos grupos atingidndo seu auge com 17 grupos. O *PAM (Partitioning Around Medoids)* tem sua melhor performance com 8 grupos.

O mdelo *hierárquico* produz um tendência a agrupar todas as observações num número pequeno de grupos. Outro ponto a notar foi a performance melhor do modelo *hierárquico* frente aos demais. Isso sugere que os atributos possuem uma relação hierárquica que de certa forma é explicado pelo tipo de atributos que o *Data Set* possui. São informações do estodo e município. Sabemos que as informações do estado são um condensado das informações do município.

O modelo *k-Means* produziu uma perfomance fraca, mostrando que talvez não seja a melhor opção.

Optou-se por fazer mais uma simulação utilizando o *Data Set* com 5 dimensões agrupando o modelo *hierárquico* e *k-Means* de 2 até 5, que é o número de regiões do país. As localidades dentro das regiões guardam mais semalhanças entre si que localidades de regiões diferentes. Como os dados tem uma relação com a localização geográfica, seria prudente mantermos essa premissa.

```r
# Ajustando o processamento paralelo em CORES (NÚCLEOS)
cl <- makePSOCKcluster(8)
registerDoParallel(cl)

clmethods <- c("hierarchical","kmeans")
arquivo <- '../model/Cluster_5dim3.rds'
set.seed(314)
cluster_5dim3 <- clValid(pca_5dim_std, nClust = 2:5, maxitems = nrow(pca_5dim_std),
                         clMethods = clmethods, method = 'complete',
                         validation = "internal")
saveRDS(cluster_5dim3, file = arquivo)

arquivo <- '../model/Cluster_6dim3.rds'
set.seed(314)
cluster_6dim3 <- clValid(pca_6dim_std, nClust = 2:5, maxitems = nrow(pca_6dim_std),
                          clMethods = clmethods, method = 'complete',
                          validation = "internal")
saveRDS(cluster_6dim3, file = arquivo)

# Resumo
summary(cluster_5dim3)
```

```
##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                 2       3       4       5
```

```
##
## hierarchical Connectivity    3.8579   5.2687  14.8790  16.4663
##              Dunn            0.6084   0.1687   0.0985   0.1245
##              Silhouette      0.9228   0.7741   0.7556   0.7408
## kmeans       Connectivity    6.9659  78.9266 103.9762  90.4821
##              Dunn            0.2849   0.0112   0.0146   0.0140
##              Silhouette      0.8973   0.4966   0.5096   0.3392
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.8579 hierarchical 2
## Dunn         0.6084 hierarchical 2
## Silhouette   0.9228 hierarchical 2
```

```
summary(cluster_6dim3)
```

```
##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                  2        3        4        5
##
## hierarchical Connectivity    3.8579   5.3798  11.7917  47.1952
##              Dunn            0.6868   0.1857   0.2084   0.0415
##              Silhouette      0.9140   0.7518   0.7520   0.4684
## kmeans       Connectivity    6.8825  89.9544 120.0317 319.3222
##              Dunn            0.3026   0.0125   0.0169   0.0059
##              Silhouette      0.8847   0.4555   0.4540   0.1788
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.8579 hierarchical 2
## Dunn         0.6868 hierarchical 2
## Silhouette   0.9140 hierarchical 2
```

```
# Parando o processamento paralelo
stopCluster(cl)
```

A simulação sugere o uso do *Hierárquico* ou *k-Means* com 2 grupos e 5 dimensões.

Podando o agrupamento hierárquico com 5 dimensões para conter somente 2 grupos.

```
hier_5dim.cluster <- cutree(hier_5dim_comp, k = 2)
sil_5dim <- silhouette(hier_5dim.cluster, dist_5dim)
fsil_5dim <- fviz_silhouette(sil_5dim)
```

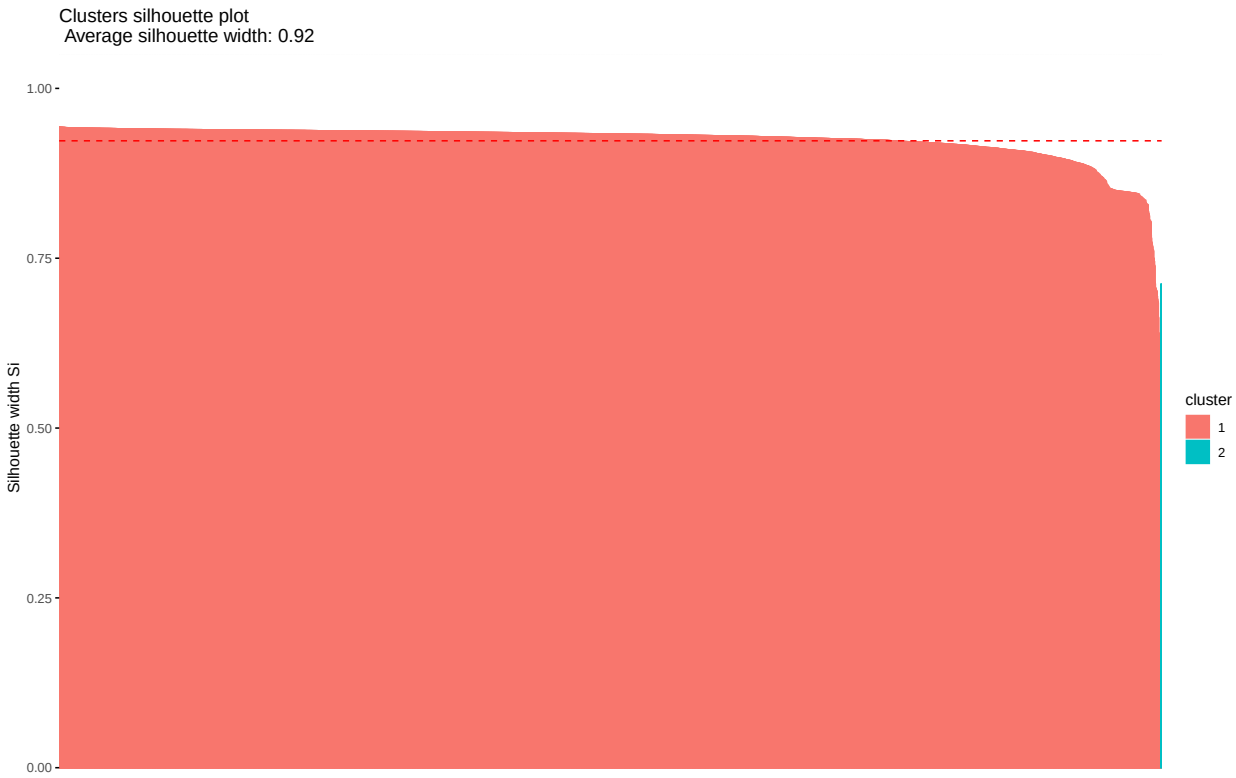
```
##   cluster size ave.sil.width
```

```
## 1       1 3742          0.92
## 2       2    2          0.68
```

```
sil <- fsil_5dim$data %>% group_by(cluster) %>% summarise(mean = mean(sil_width),
                                                          sd = sd(sil_width),
                                                          n = n())
summary(sil_5dim)
```

```
## Silhouette of 3744 units in 2 clusters from silhouette.default(x = hier_5dim.cluster, dist = dist_5di
##  Cluster sizes and average silhouette widths:
##      3742         2
## 0.9229740 0.6809646
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1037  0.9234  0.9328  0.9228  0.9374  0.9434
```

```
fsil_5dim
```



Percebe-se que o modelo hierárquico tem a tendência a agrupar todos em um mesmo grupo, deixando de fora as obervações consideradas *outliers*.

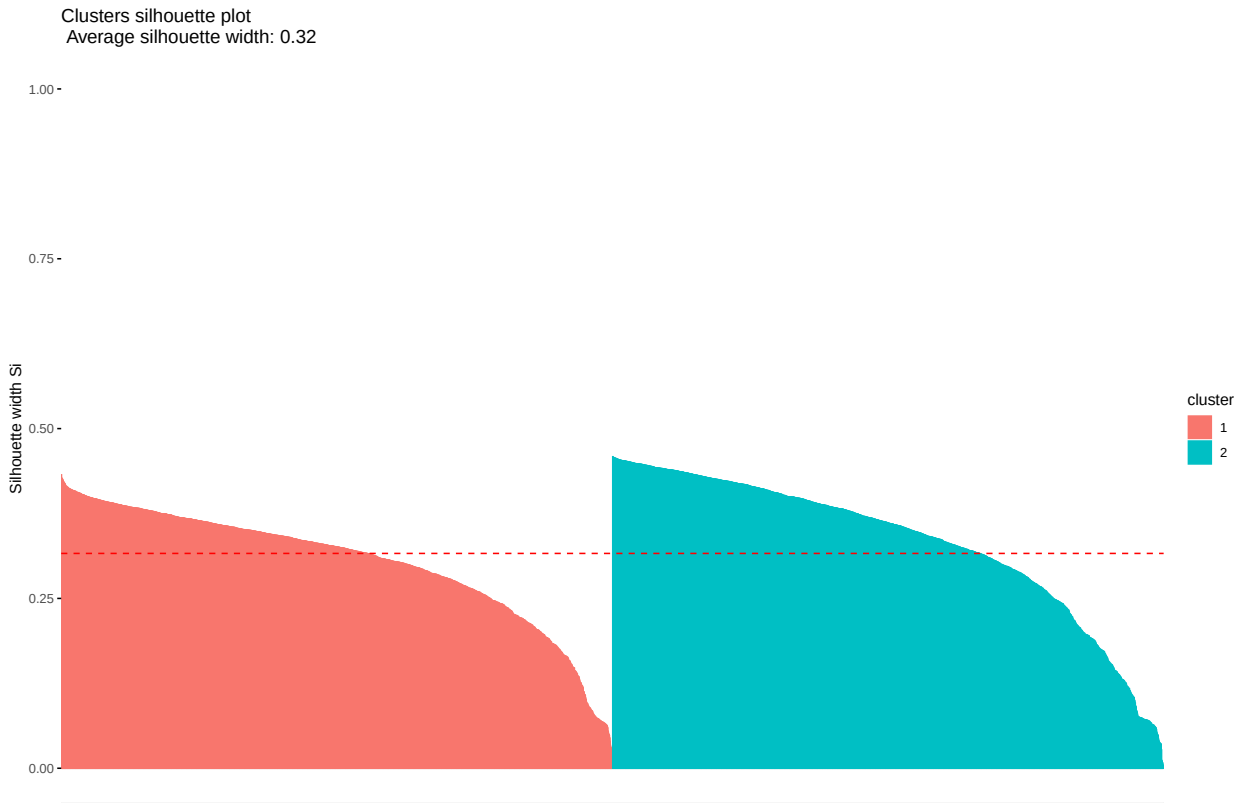
```
nGrupos <- 2
set.seed(314)
kmeans_5dim <- kmeans(pca_5dim_std, nGrupos, nstart = 314, iter.max = 100)
sil_5dim <- silhouette(kmeans_5dim$cluster, dist_5dim)
fsil_5dim <- fviz_silhouette(sil_5dim)
```

```
##   cluster size ave.sil.width
## 1       1 1873          0.30
## 2       2 1871          0.33
```

```
sil <- fsil_5dim$data %>% group_by(cluster) %>% summarise(mean = mean(sil_width),
                                                          sd = sd(sil_width),
                                                          n = n())
summary(sil_5dim)
```

```
## Silhouette of 3744 units in 2 clusters from silhouette.default(x = kmeans_5dim$cluster, dist = dist_5
##  Cluster sizes and average silhouette widths:
##      1873      1871
## 0.3016122 0.3308551
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000605 0.268848 0.338855 0.316226 0.386172 0.458567
```

```
fsil_5dim
```



Com o modelo *k-Means*, nota-se um equilíbrio maior no agrupamento, tanto na métrica utilizada quanto no número de participantes, são semelhantes. Mas possui um valor de *Silueta* fraco.

Como o modelo *hierárquico* tende a agrupar todas as observações num grupo único, deixando de fora somente poucas observações, vamos prosseguir com a análise utilizando tanto o *k-Means* quanto o *hierárquico*, apesar do valor baixo da métrica *Silueta* para o modelo *k-Means*.

Conclusão

Criando o *Data Set* final com os agrupamentos.

```
ds_final <- ds_covid
ds_final[, ':='(kmeans_5dim = kmeans_5dim$cluster,
                hier_5dim = hier_5dim.cluster)]
ds_final[, perc_pobre_extremo := num_pobre_extremo / sum(num_pobre_extremo),
         by = nome_cidade]
str(ds_final)
```

```
## Classes 'data.table' and 'data.frame':   3744 obs. of  22 variables:
##  $ codigo_cidade                  : int  15 15 23 23 49 49 98 98 106 106 ...
##  $ eh_branco                      : logi  TRUE FALSE TRUE FALSE TRUE FALSE ...
##  $ nome_cidade                    : chr  "Alta Floresta D'Oeste" "Alta Floresta D'Oeste" "Ariquemes"
##  $ uf                             : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ nome_uf                        : chr  "Rondônia" "Rondônia" "Rondônia" "Rondônia" ...
##  $ total_leitos                   : int  49 49 240 240 360 360 54 54 97 97 ...
##  $ num_caso_confirmado            : int  1 1 89 89 2 2 1 1 6 6 ...
##  $ num_mortes                     : int  0 0 0 0 0 0 0 0 2 2 ...
##  $ pop_estimada_2019              : int  22945 22945 107863 107863 85359 85359 32374 32374 46174 461
##  $ caso_confirmado_100k_habitantes: num  4.36 4.36 82.51 82.51 2.34 ...
##  $ taxa_mortalidade               : num  0 0 0 0 0 ...
##  $ renda_domiciliar               : num  6000725 5269952 26759062 33641644 30951426 ...
##  $ populacao                      : int  10482 13615 31842 57924 32635 45506 13211 15299 8336 31593
##  $ num_pobre                      : int  4831 6837 7899 19974 9179 15905 4606 6064 2793 15663 ...
##  $ num_pobre_extremo              : int  2804 3492 3446 8076 4536 6976 2093 2322 1254 8218 ...
##  $ num_desocupado                 : int  181 322 686 1352 892 1372 223 418 273 827 ...
##  $ total_pop_economico            : int  4451 5592 15703 28244 16449 22121 6653 7566 3689 12404 ...
##  $ num_trab_infantil              : int  153 175 323 793 423 820 317 456 57 247 ...
##  $ total_pop_infantil             : int  1127 1779 3714 7272 3189 5631 1300 2052 898 4443 ...
##  $ kmeans_5dim                    : int  2 1 2 1 2 1 2 1 2 1 ...
##  $ hier_5dim                      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ perc_pobre_extremo             : num  0.445 0.555 0.299 0.701 0.394 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

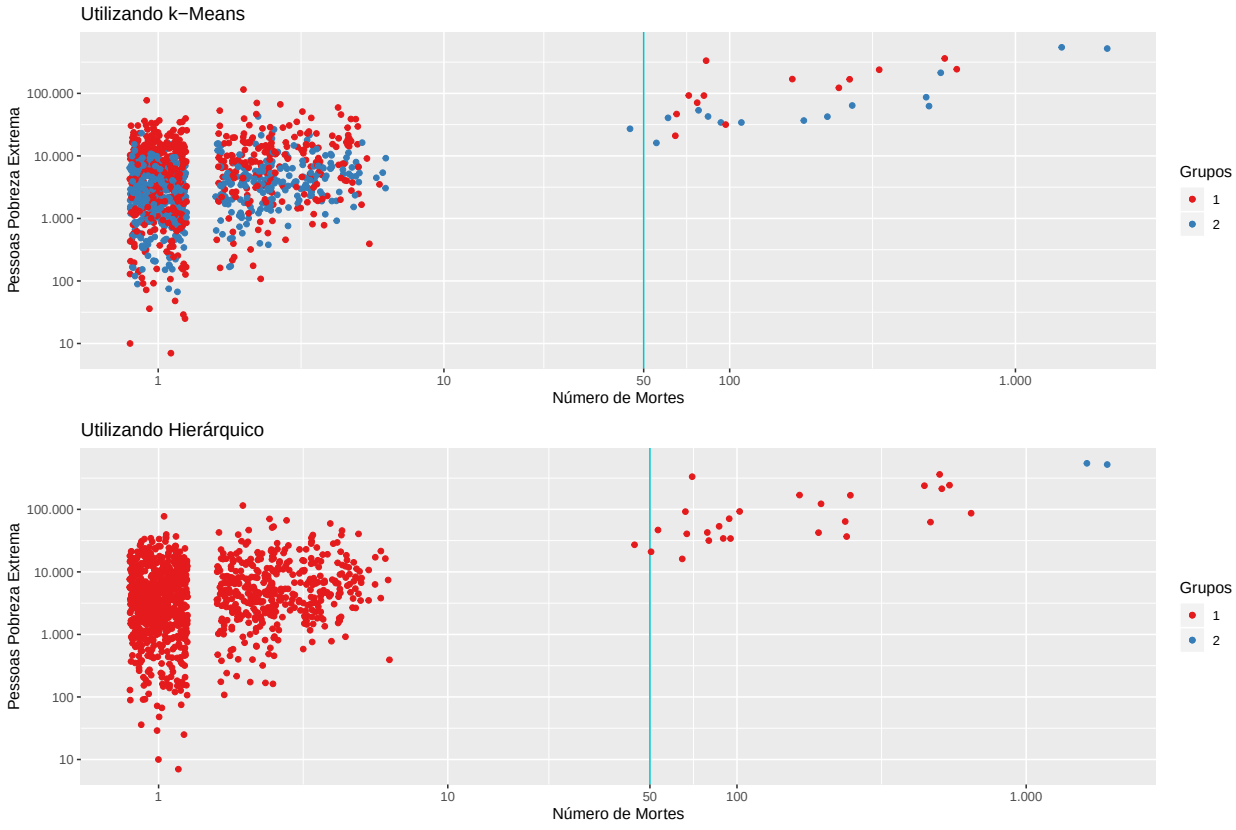
O gráfico mostra que a relação entre pobreza externa e número de mortes pelo COVID é crescente, ou seja, quanto maior a pobreza maior o número de mortos. Outra informação que o gráfico traz é que são poucas cidades que possuem um número de mortes maior que 50 e que o número de mortes está relacionado com a pobreza extrema.

```
g1 <- ds_final[num_mortes>0 & num_mortes<=5,] %>%
  ggplot(aes(x = num_mortes, y = num_pobre_extremo)) +
  geom_jitter(aes(color = as.factor(kmeans_5dim)), width = 0.1) +
  geom_vline(xintercept = 50, color = 'turquoise3') +
  geom_jitter(data = ds_final[num_mortes>50,],
              aes(x = num_mortes, y = num_pobre_extremo, color = as.factor(kmeans_5dim)),
              width = 0.1) +
  scale_color_brewer(palette = 'Set1') +
  scale_x_log10(breaks = c(1,10,50,100,1000),
                labels = c('1','10','50','100','1.000')) +
  scale_y_log10(breaks = c(1e1,1e2,1e3,1e4,1e5),
                labels = c('10','100','1.000','10.000','100.000')) +
```

```
  labs(x = 'Número de Mortes', y = 'Pessoas Pobreza Extrema', color = 'Grupos',
       title = 'Utilizando k-Means')

g2 <- ds_final[num_mortes>0 & num_mortes<=5,] %>%
  ggplot(aes(x = num_mortes, y = num_pobre_extremo)) +
  geom_jitter(aes(color = as.factor(hier_5dim)), width = 0.1) +
  geom_vline(xintercept = 50, color = 'turquoise3') +
  geom_jitter(data = ds_final[num_mortes>50,],
              aes(x = num_mortes, y = num_pobre_extremo, color = as.factor(hier_5dim)),
              width = 0.1) +
  scale_color_brewer(palette = 'Set1') +
  scale_x_log10(breaks = c(1,10,50,100,1000),
                labels = c('1','10','50','100','1.000')) +
  scale_y_log10(breaks = c(1e1,1e2,1e3,1e4,1e5),
                labels = c('10','100','1.000','10.000','100.000')) +
  labs(x = 'Número de Mortes', y = 'Pessoas Pobreza Extrema', color = 'Grupos',
       title = 'Utilizando Hierárquico')

grid.arrange(g1,
             g2,
             nrow = 2,
             ncol = 1)
```

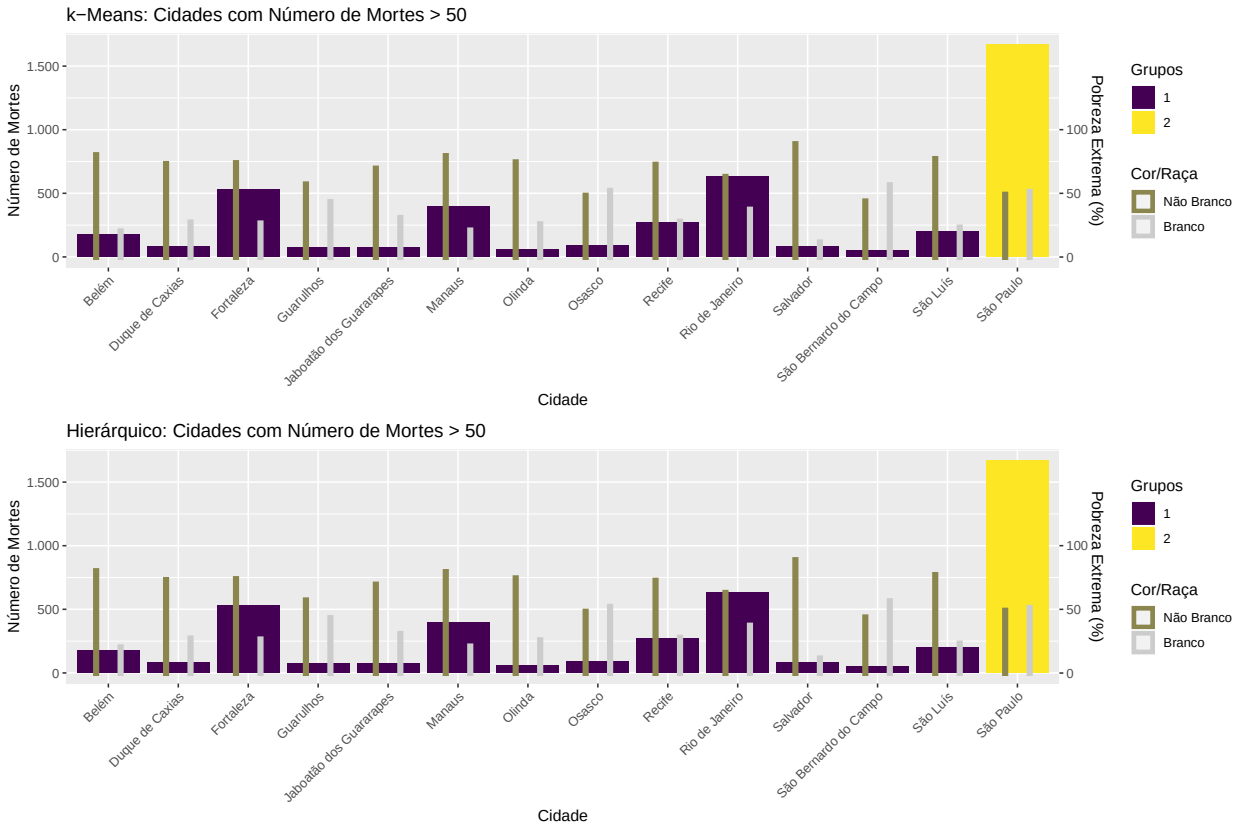


O gráfico mostra que as cidades com número de mortes maior que 50 possuem um percentual de pobreza de pessoas não brancas maior que de pessoas brancas. Com exceção de São Paulo que aponta para um equilíbrio.

```r
limite_inferior <- 50
g1 <- ds_final[num_mortes >= limite_inferior & eh_branco == FALSE,] %>%
  ggplot(aes(x = as.factor(nome_cidade), y = num_mortes)) +
  geom_col(aes(fill = as.factor(kmeans_5dim))) +
  geom_col(data = ds_final[num_mortes >= limite_inferior,],
               aes(x = as.factor(nome_cidade), y = perc_pobre_extremo*1e3,
                   color = as.factor(eh_branco)),
           width = 0, size = 2, alpha = 0, position = position_dodge(width = 0.7)) +
  scale_y_continuous(breaks = c(0, 500, 1000, 1500),
                     labels = c('0', '500', '1.000', '1.500'),
                     sec.axis = sec_axis(~ ./10,
                                         breaks = c(0,50,100),
                                         labels = c('0','50','100'),
                                         name = 'Pobreza Extrema (%)')) +
  scale_color_manual(values = c('khaki4','gray80'), breaks = c(FALSE,TRUE),
                     labels = c('Não Branco','Branco')) +
  scale_fill_viridis_d() +
  labs(x = 'Cidade', y = 'Número de Mortes', color = 'Cor/Raça', fill = 'Grupos',
       title = 'k-Means: Cidades com Número de Mortes > 50') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

g2 <- ds_final[num_mortes >= limite_inferior & eh_branco == FALSE,] %>%
  ggplot(aes(x = as.factor(nome_cidade), y = num_mortes)) +
  geom_col(aes(fill = as.factor(hier_5dim))) +
  geom_col(data = ds_final[num_mortes >= limite_inferior,],
           aes(x = as.factor(nome_cidade), y = perc_pobre_extremo*1e3,
               color = as.factor(eh_branco)),
           width = 0, size = 2, alpha = 0, position = position_dodge(width = 0.7)) +
  scale_y_continuous(breaks = c(0, 500, 1000, 1500),
                     labels = c('0', '500', '1.000', '1.500'),
                     sec.axis = sec_axis(~ ./10,
                                         breaks = c(0,50,100),
                                         labels = c('0','50','100'),
                                         name = 'Pobreza Extrema (%)')) +
  scale_color_manual(values = c('khaki4','gray80'), breaks = c(FALSE,TRUE),
                     labels = c('Não Branco','Branco')) +
  scale_fill_viridis_d() +
  labs(x = 'Cidade', y = 'Número de Mortes', color = 'Cor/Raça', fill = 'Grupos',
       title = 'Hierárquico: Cidades com Número de Mortes > 50') +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(g1,
             g2,
             nrow = 2,
             ncol = 1)
```



Percebemos também que a relação entre o atributo num_mortes e demais atributos que expressam a pobreza não foram determinantes no agrupamento. Apesar dessa relação ser importante e essas variáveis estarem bem correlacionadas, essa relação por si só não responde pelo agrupamento obtido. Esse fato nos leva a supor que existam outras variáveis também são muito determinantes para fazer a separação dos grupos.