Consistência da Amostra de 1,27% do Censo de 1960

Rogério Jerônimo Barbosa

20 de agosto de 2018

## Amostra de 1,27%: origem dos dados e a necessidade de consistência

Como dito anteriormente, o próprio IBGE não disponibiliza ou comercializa qualquer versão do Censo Demográfico de 1960. Aquela possuída pelo Centro de Estudos da Metrópole da Amostra de 1,27% é fruto de uma doação do acervo pessoal de dados dos pesquisadores Carlos Antônio Costa Ribeiro Filho e Adalberto Cardoso, ambos ligados ao Instituto de Estudos Sociais e Políticos (IESP) da Universidade do Estado do Rio de Janeiro (UERJ), que por sua vez obtiveram os dados também como doação, feita pelos pesquisadores americano Edward Telles (UC-Santa Barbara) e Charles Wood (Univ. Florida).

Ainda que todos esses pesquisadores envolvidos sejam de competência extrema e publicamente atestada, o fato de que tais dados foram veiculados por meio de transferências interpessoais e não oficiais já atentaria para a necessidade da realização de chegagens e conferências, contra relatórios e tabulações oficiais – na expectativa de averiguar eventuais discrepâncias e alterações. Soma-se a isso o fato de que os métodos de gravação e meios de estocagem dos dados, à época de sua extração original (1965), eram muito menos confiáveis, podendo gerar erros e corrupção de informação. Por essas duas razões foi preciso proceder uma cuidadosa análise e consistência dos dados da Amostra de 1960, anterior a qualquer uso ou harmonização.

### Consistência, transparência e replicabilidade: RMarkdown e Github

Não foi possível, contudo, lançar mão de procedimentos exclusivamente automatizados em todas as etapas. Diversos momentos exigiram cuidadosa leitura manual de linhas do arquivo de dados em formato fixo, para que fosse possível identificar a natureza dos erros e procedimentos cabíveis para eventuais correções. Descrevemos detalhadamente neste documento todos passos, apresentando, no próprio corpo do texto as linhas de código utilizadas na linguagem R. O propósito fundamental é maximizar a transparência e a replicabilidade, uma vez que serão aplicadas modificações e correções sobre dados oficiais, cuja importância não é apenas histórica. Apesar das tecnicalidades envolvidas nessa estratégia de apresentação, zelaremos pela manutenção da simplicidade da exposição.

Este presente documento foi escrito na liguagem [RMarkdown](https://rmarkdown.rstudio.com/), um tipo simplificado de linguagem de marcação, semelhante ao HTML, XML ou LaTeX, mas completamente integrada à plataforma R, permitindo a inserção de blocos de códigos intercalados entre partes do corpo do texto. Isto traz a vantagem de centralizar todos os procedimentos de análise num único documentos que contém, ao mesmo tempo, scripts que carregam dados, executam transformações e geram os finais. Todos os arquivos adicionais utilizados aqui (sempre por meio de chamadas em linhas de código) encontram-se disponíveis on-line num repositório do GitHub: [<https://github.com/antrologos/ConsistenciaCenso1960Br>].

Iniciando os passos da análise, carregaremos aqui os pacotes necessários para todas as análises que serão realizadas posteriormente:

library(tidyverse)  
library(readxl)  
library(descr)  
library(data.table)

Carregaremos também um conjunto de funções criadas especificamente para auxiliar nas tarefas avaliação e implementação dos procedimentos de consistência realizados aqui:

source("https://raw.githubusercontent.com/antrologos/ConsistenciaCenso1960Br/master/Code/Utils.R")

## Arquivos originais e formato dos dados

A pasta dos arquivos doados do Censo de 1960 consistia de quatro itens:

1. [HHOLDA.txt](https://github.com/antrologos/ConsistenciaCenso1960Br/blob/master/Original%20Files/HHOLDA.txt)
2. [dicionario\_60\_last.doc](https://github.com/antrologos/ConsistenciaCenso1960Br/blob/master/Original%20Files/dicionario_60_last.doc)
3. [CENSO1960 - MONTA ARQUIVO.SPS](https://github.com/antrologos/ConsistenciaCenso1960Br/blob/master/Original%20Files/CENSO1960%20-%20MONTA%20ARQUIVO.SPS)
4. [GeraBookSAS CD60\_1%.txt](https://github.com/antrologos/ConsistenciaCenso1960Br/blob/master/Original%20Files/GeraBookSAS%20CD60_1%25.txt)

O primeiro deles, é o próprio conjunto dos dados. Trata-se, como adiantado anteriormente, de um arquivo de formato fixo, em que as posições absolutados dos caracteres nas colunas do texto, sem qualquer separador ou marca adicional, determinam o conteúdo das variáveis. O conteúdo das dez primeiras linhas é o seguinte:

readLines("Original Files/HHOLDA.txt", n = 10) %>% writeLines()

## 0000110100014051111478950580206002 \0000001  
## 0000110100014051212716057149191415200400000076- \0000001  
## 0000110100014051312311657159191508210000000035- \0000001  
## 0000110100014052111482940580206001 \0000002  
## 000011010001405221271245704919041732000000008318728145\0000002  
## 0000110100014053111478949579208004 \0000003  
## 000011010001405321171566567019041821073015150387326146\0000003  
## 00001101000140533129103651392000- \0000003  
## 0000110100014053312814965159190415200730151534- \0000003  
## 0000110100014053312912365049190407322000000035- \0000003

Como podemos observar, aparentemente as linhas do arquivo de dados possuem o mesmo comprimento: 62 caracteres. A razão para isso é a presença de uma estrutura de dígitos bastante padronizada, que se inicia no caractere 55, iniciando por uma barra invertida – e.g.: \0000001. Trata-se de uma numeração das famílias que habitam os domicílios. Assim, todas as linhas marcadas com o mesmo sulfixo desse tipo pertencem à mesma família. As linhas não representam apenas os indivíduos entrevistados, contudo. O primeiro registro de uma família representa traz consigo as características do domicílio onde residem. Deste modo, dizemos que os dados dados acima apresentam uma estrutura hierárquica, pelo fato de que suas linhas contém tanto a unidade primária dos microdados (os indivíduos) como também as estruturas de agregação nas quais estão aninhados (famílias).

Além do arquivo de dados, a pasta continha também a) o dicionário de códigos informando o layout e as coordenadas para a interpretação do arquivo de texto em formato fixo; b) uma sintaxe de abertura em SAS; c) uma sintaxe de abertura em SPSS. Análises preliminares, contudo, indicaram que o dicionário de códigos se dirigia a um layout muito diferente daquele utilizado nas duas sintaxes de abertura, tanto em SAS, como em SPSS. Divergiam, assim, nas coordenadas dos caracteres para início de fim da leitura de praticamente todas as variáveis. O documento do dicionário de códigos, no item “Origem dos dados” informa que se trata da:

AMOSTRA DE 25% DO CENSO DEMOGRÁFICO DE 1960, EXTRAÍDO DE ARQUIVO DE DADOS GRAVADO EM FITA MAGNÉTICA, O QUAL INCLUI DADOS PARA OS SEGUINTES ESTADOS: RN, AL, BA, CE, PB, PE, SE, FN, GO, MT, DF, SA,MG, RJ, PR,SP E RS.

Com isso, inferimos que o layout desenhado para a amostra de 25% (com apenas algumas UFs) pelos técnicos do IBGE não é o mesmo daquele para a de 1,27% (com todas as UFs), contida no arquivo de dados.

Outra fonte de confusão diz respeito ao título e conteúdo do script em formato SAS, que informam que se trataria de uma amostra de 1%. No entanto, isso não parece ter fundamento. A amostra de 1,27% descrita no Volume 2 dos Resultados Preliminares do Censo Demográfico de 1960 (IBGE, 1965) teria sido gerada por sorteio aleatório e equiprobabilistico. Se isso é verdade, então as proporções de quaisquer categorias de variáveis no banco de dados seriam semelhantes às (convergiriam em probabilidade para as) populacionais; sem qualquer necessidade de correções por pesos amostrais. E, de fato, se criamos pesos amostrais idênticos para todos os casos (com valor igual à 1/0,0127 - um sobre a fração amostral) para servir de mero fator de expansão ) obtemos resultados e totais muito semelhantes aos números das tabulações oficiais. Não há razões para crer que o arquivo de dados diga respeito a outra versão, que não aquela originalmente extraída em 1965.

Como layout, deste modo, seguiremos a estrutura descrita nos arquivo em SPSS e SAS, ao invés daquela apresentada no dicionário. No entanto, é preciso pontuar, os valores e rótulos das categorias são os mesmos em todas essas três fontes, a despeito das divergencias nas localizações dos caracteres.

A ausência de um dicionário dos dados não é um problema de todo ignorável: as duas sintaxes de abertura não fazem a leitura de todos os caracteres do arquivo de dados e, na ausência de uma documentação oficial e completa, não há como avaliar de modo unívoco a natureza das informações guardadas naquelas posições. Mais especificamente, os caracteres localizados entre as posições (colunas) de 7 a 16 do arquivo de texto estão sendo ignorados. Diversos testes e cruzamentos foram realizados na tentativa de identificar esses conteúdos, porém sem sucesso.

Além disso, as seguintes variaveis estao citadas no dicionario de codigos (feito para a amostra de 25%), mas nao estao sendo abertas pelas sintaxes construídas para o arquivo de dados (e nao parecem estar contidas nos caracteres de 7 a 16):

* V100 - TOTAL DE PESSOAS
* V119 - TOTAL FAMÍLIAS
* V120 - TOTAL MORADORES
* V121 - PESO DOMICÍLIO
* V122 - FILLER

A principio, é possivel calcular V100, V119 e V120 a posteriori. Com respeito à V121, se os dados forem mesmo uma amostra autoponderada, como supusemos acima, os pesos dos domicílios poderão ser calculadas também como iguais a 1/0,0127 para todos os registros. O conteudo da v122 não é descrito em nenhum outro lugar. Mas, pela sua denominação, parece ser tratar apenas de um caractere de preenchimento ou algum tipo de espaço vazio, separando seções dos dados, sem outro conteúdo substantivo.

Há também variaveis de identificacao listadas no inicio do dicionario, mas não que são referidas nas sintaxes:

* 1 - PASTA
* 2 - BOLETIM
* 3 - IDENTIFICAÇÃO
* 4 - DIGITO VERIFICADOR

Possivelmente, a elas se refere o conteúdo dos caracteres de 7 a 16: identificadores do domicílio, local de moradia e características que poderiam até auxiliar na compreensão do plano amostral. Mas não há método determístico e confiável para separar o teor dessas colunas, nem informações externas para validá-los. Deste modo os caracteres de 7 a 16 permanecerão ignorados.

Por fim, há duas variáveis bastante importantes listadas apenas nas sintaxes – e não no dicionário: são elas as variáveis sobre unidade da federacao (UF) e sobre o tipo de registro (RECD). A primeira identifica o lugar de residência do indivíduo, onde a entrevista foi realizada, a segunda, discrimina a natureza da informação contida nas linhas: se famílias ou pessoas.

**IMPORTANTE**:

O arquivo de dados de domicilios parece dizer respeito, na verdade, a dados de **FAMILIAS**. Os indícios são os seguintes:

1. A variavel V101 identifica familias dentro do domicilio particular. Se v101 = 4, trata-se da segunda familia dentro de um domicilio particular.
2. Registros identificados como 2a ou 3a familia (v101 == 4 ou v101 == 5) nao contêm informações para as variáveis v102 a 113. Presume-se que apenas a primeira familia traz consigo as informações estruturais sobre o domicílio (número de cômodos, acesso à energia elétrica etc.).

Sob esta suposição, apenas os registros identificados como v101 == 1, 2 ou 3 são de fato domicilios. Os procedimentos que aplicaremos deste ponto em diante partirão desta suposição.

## Abertura dos dados e identificação de registros de famílias e pessoas

Inicialmente, o arquivo de dados foi aberto como um grande arquivo de texto (formato character/string), sem separação de colunas.

# Objeto que guarda as os dados originais (em formato string/character)  
c60\_string <- readLines("Original Files/HHOLDA.txt")  
  
# Remove os hífens existentes no final das linhas  
c60\_string <- gsub(x = c60\_string, pattern = "-", " ")  
  
# Vetor que guarda o número de linhas existentes (será usado para criar IDs de pessoas, famílias e domicílios)  
n\_linhas <- length(c60\_string)

Todos os registros apresentaram o mesmo tamanho:

freq(nchar(c60\_string), plot = F)

## nchar(c60\_string)   
## Frequência Percentual  
## 62 1074328 100  
## Total 1074328 100

Foi preciso então identificar e separar os registros de famílias e pessoas. A princípio, de acordo com o layout descrito nas sintaxes de abertura, essa informação estaria contida no caractere da posição 17 no arquivo de dados. A frequencia das categorias dessa variável revela a seguinte distribuição:

teste\_char17\_dom\_pess = substr(c60\_string, start = 17, stop = 17)  
freq(teste\_char17\_dom\_pess, plot = F)

## teste\_char17\_dom\_pess   
## Frequência Percentual  
## 1 174467 16.24  
## 2 174472 16.24  
## 3 725389 67.52  
## Total 1074328 100.00

De acordo com informações contidas nas sintaxes, o valor 1 identificaria os registros de domicílios – por conseguinte, os valores 2 e 3, diriam respeito às pessoas. Não há na documentação, contudo, rótulo para o conteúdo desses dois valores. Porém, posteriormente, cruzando essa informação com a variável sobre relação com o chefe da família, é possível inferir os seguintes significados das alternativas:

1. Registro de familia
2. Registro da pessoa na posição de chefe da familia
3. Registro de outros moradores

A princípio, deveria haver um número idêntico de registros de famílias de chefes. Há, no entanto, uma pequena diferença, de 5 casos, como aponta a tabela acima - uma primeira inconsistência detectada aqui.

A partir das sintaxes de abertura, um arquivo de layout foi elaborado para pessoas e famílias, para facilitar a leitura das posições das variáveis, além de guardar outras informações relevantes sobre as variáveis. É a esse arquivo que faremos referência a partir daqui, quando nos referirmos ao layout dos dados:

input\_file <- "Auxiliary Files/Census1960\_input\_Sample\_1.27.xlsx"

### Registros de famílias

Utilizando da informação acima, criamos então um objeto separado para os dados de famílias, selecionando apenas as linhas que, no caractere 17, apresentaram o valor um:

# String apenas com registros de famílias  
c60\_string\_hh <- c60\_string[which(teste\_char17\_dom\_pess == 1)]  
num\_linha\_dom <- (1:n\_linhas)[which(teste\_char17\_dom\_pess == 1)]

Abrimos também o arquivo de layout refere às familias:

input\_hh <- read\_xlsx(input\_file, sheet = "Family\_open")  
input\_hh

## # A tibble: 23 x 5  
## Var Start End Valid\_values obs   
## <chr> <dbl> <dbl> <chr> <chr>   
## 1 UF 1 2 sheet <NA>   
## 2 V116 3 6 <NA> <NA>   
## 3 place\_holder\_~ 7 8 <NA> pode ser o distrito do mu~  
## 4 place\_holder\_~ 9 16 <NA> pode ser um ID de Família~  
## 5 REC\_TYPE 17 17 1;2;3 <NA>   
## 6 V118 18 18 1;3;5 <NA>   
## 7 V101 19 19 1;2;3;4;5;9 <NA>   
## 8 V102 20 20 ;4;5;6;7 <NA>   
## 9 V103 21 21 ;7;8;9;0 <NA>   
## 10 V104 22 22 ;0;1;2;3;4;5;6;7~ <NA>   
## # ... with 13 more rows

A partir da análise e leitura do layout, observamos:

* O caractere 32 não está sendo lido ou nem parece fazer parte de qualquer variável
* Os caracteres de 35 a 54 (intervalo de 20 posições), igualmente, não são lidos

No primeiro caso, na posição 32, encontramos 1.257 espaços vazios, 173.209 valores zero, 1 (um) valor 2, como mostra a tabela abaixo.

teste\_char32\_dom = substr(c60\_string\_hh, start = 32, stop = 32)  
freq(teste\_char32\_dom, plot = F)

## teste\_char32\_dom   
## Frequência Percentual  
## 1257 0.7204801  
## 0 173209 99.2789467  
## 2 1 0.0005732  
## Total 174467 100.0000000

Esses resultados parecem indicar que o conteúdo esperado para essa célula seria o valor zero. Contudo, possivelmente por inconsistencia dos registros, outros conteúdos figuram. No caso do intervalo entre os caracteres 35 e 54, parece ocorrer um menor número de problemas – que, porém, não são desprovidos de importância, como mostra a tabela a seguir:

teste\_char\_35\_54\_dom = substr(c60\_string\_hh, start = 35, stop = 54)  
freq(teste\_char\_35\_54\_dom, plot = F)

## teste\_char\_35\_54\_dom   
## Frequência Percentual  
## 174465 99.9988537  
## 02 1 0.0005732  
## 81 07 601 71 11 4789 1 0.0005732  
## Total 174467 100.0000000

Veremos adiante que esses dois casos problemáticos são registros corrompidos – provavelmente devido a erros de gravação, típicos dos antigos métodos de gravação serial (por meio de fitas magnéticas). Possivelmente se devem a algum tipo de descarrilhamento dos cabeçotes de leitura/gravação. Haverá muitos outros problemas da mesma natureza.

Aplicamos então a função especificamente construída para aplicar o arquivo de layout sobre os dados, separando as colunas e testando se os valores das observacoes em cada variavel estão de fato listadas dentro do escopo de possibilidades definidas no dicionario:

censo1960\_hh <- aplicaTesta\_layout(dados\_string = c60\_string\_hh,  
 input = input\_hh) %>%  
 mutate(  
 # adicionamos uma coluna que indica a linha onde o registro se encontra no arquivo de dados  
 num\_linha\_dom = num\_linha\_dom  
 ) %>%   
 select(num\_linha\_dom, everything())

O resultado é um banco de dados no qual todas as colunas ainda possuem o formato texto (string/character), guardando todas as informações originais (sem transformar campos numéricos em integer ou doubles, por exemplo, o que removeria os zeros à esquerda). Além disso, uma coluna adicional de verificação foi criada ao lado de cada variável, indicando TRUE quando os valores observados são válidos (listados no dicionário) e FALSE, caso contrário.

censo1960\_hh

## # A tibble: 174,467 x 43  
## num\_linha\_dom UF test\_UF V116 test\_V116 place\_holder\_07~  
## <int> <chr> <lgl> <chr> <lgl> <chr>   
## 1 1 00 TRUE 0011 TRUE 01   
## 2 4 00 TRUE 0011 TRUE 01   
## 3 6 00 TRUE 0011 TRUE 01   
## 4 27 00 TRUE 0011 TRUE 01   
## 5 32 00 TRUE 0011 TRUE 01   
## 6 34 00 TRUE 0011 TRUE 01   
## 7 40 00 TRUE 0011 TRUE 01   
## 8 44 00 TRUE 0011 TRUE 01   
## 9 55 00 TRUE 0011 TRUE 01   
## 10 67 00 TRUE 0011 TRUE 01   
## # ... with 174,457 more rows, and 37 more variables:  
## # place\_holder\_09\_16 <chr>, REC\_TYPE <chr>, test\_REC\_TYPE <lgl>,  
## # V118 <chr>, test\_V118 <lgl>, V101 <chr>, test\_V101 <lgl>, V102 <chr>,  
## # test\_V102 <lgl>, V103 <chr>, test\_V103 <lgl>, V104 <chr>,  
## # test\_V104 <lgl>, V105 <chr>, test\_V105 <lgl>, V106 <chr>,  
## # test\_V106 <lgl>, V107 <chr>, test\_V107 <lgl>, V108 <chr>,  
## # test\_V108 <lgl>, V109 <chr>, test\_V109 <lgl>, V110 <chr>,  
## # test\_V110 <lgl>, V111 <chr>, test\_V111 <lgl>, V112 <chr>,  
## # test\_V112 <lgl>, place\_holder\_32\_32 <chr>, V113 <chr>,  
## # test\_V113 <lgl>, place\_holder\_35\_54 <chr>, barra <chr>,  
## # test\_barra <lgl>, ID <chr>, test\_ID <lgl>

O segundo passo é identificar as linhas que contêm ao menos um erro. Para isso, criamos uma coluna adicional que aponta os registros em que há pelo menos um valor FALSE nas variáveis de teste:

censo1960\_hh$contains\_invalid <- censo1960\_hh %>%   
 select(starts\_with("test\_")) %>%  
 mutate\_all(function(x) as.numeric(!(x))) %>%  
 replace(is.na(.), 0) %>%  
 rowSums(.)

Devemos também nos assegurar de o ID das famílias (valor que se inicia com \0000001 e assim por diante) de fato é único para cada registro. Trata-se de uma informacao que parece ter sido adicionada ao banco posteriormente, sempre a partir do caractere 55de cada linha. O seguintes aspectos devem ser observados: a) como se trata de um valor, esse campo deve poder ser transformada em numerico sem gerar qualquer “caso perdido” (*missing*); b) não pode haver valores repetidos (afinal se trata de um ID).

Como se pode observar, nenhum valor missing é produzido pela conversão do campo em numérico:

id\_numeric <- as.numeric(censo1960\_hh$ID)  
paste("Quantidade de valores missing:", sum(is.na(id\_numeric)))

## [1] "Quantidade de valores missing: 0"

Além disso, também não se encontra valores repetidos:

paste("Quantidade de valores repetidos:", sum(duplicated(id\_numeric)))

## [1] "Quantidade de valores repetidos: 0"

Os dois critérios são satisfeitos. Examinemos então os casos com pelo menos um registro problemático, segundo o critério aventado acima. Os testes preliminares revelam que são 12 casos problemáticos, em que se se verificou pelo menos um valor inválido, não listado no dicionário de códigos. Gravaremos então num objeto o número da linha desses registros identificados:

# Problema 1: presenca de caracteres ou valores não listados como validos pelo dicionario de codigos:  
hh\_linhas\_problemas\_1 <- censo1960\_hh %>%   
 filter(contains\_invalid > 0) %>%  
 .$num\_linha\_dom

Um segundo tipo de problema diz respeito à presença de outros tipos de caracteres em quaisquer posições do registro (inclusive aquelas não lidas, como os caracteres de 7 a 16, 32 e de 35 a 54). Fazemos então uma busca por expressões regulares e registramos os números das linhas dos casos problemáticos:

# Problema 2: presenca de outros tipos de caracteres invalidos  
hh\_linhas\_problemas\_2 = grep(pattern = "[[:alpha:]]|[[:cntrl:]]|[[:punct:]]" ,   
 x = gsub(pattern = "[\\]",   
 replacement = "",   
 x = c60\_string\_hh))  
hh\_linhas\_problemas\_2 = censo1960\_hh[hh\_linhas\_problemas\_2,"num\_linha\_dom"] %>% unlist()

São exemplos desse tipo de ocorrência:

writeLines(c60\_string[hh\_linhas\_problemas\_2[1:7]])

## 212166X722366154151478384680205003 \0037520  
## 54554131553021421114849'2570202001 \0104316  
## 54554232553422051314821'0570204002 \0104558  
## 54554232553422591314849'2570206003 \0104612  
## 60623477616981071114099@2570204002 \0124092  
## 60623477616981091114869'2579105004 \0124094  
## 71724Z0771038216151598279680202001 \0145606

O terceiro tipo de problema está ligado à existência de espaços em branco adicionais inseridos entre valores válidos, levando o registro a experimentar uma espécie de “descarrilhamento” da gravação. Em boa parte dos casos dessa natureza, basta remover os espaços brancos e então o padrão dos dados passa a fazer sentido: encontramos valores válidos para as variáveis e padrões de resposta consistentes. A identificação desses espaços vazios é feita através do código abaixo:

# Problema 3: presenca de espacos (até 3) entre caracteres  
hh\_linhas\_problemas\_3 = grep(pattern = "[[:digit:]][[:blank:]]{1,3}[[:digit:]]",  
 x = c60\_string\_hh)  
hh\_linhas\_problemas\_3 = censo1960\_hh[hh\_linhas\_problemas\_3,"num\_linha\_dom"] %>% unlist()

São exemplos desse tipo de ocorrência:

writeLines(c60\_string[hh\_linhas\_problemas\_3[1:10]])

## 313131013143809415157838 680207002 \0047256  
## 5151680151242001111483 42580204001 \0088188  
## 52531101534000171314 81525 0205002 \0096545  
## 545441025438200711148294257920 001 \0099422  
## 5454410354382129111485 42579104001 \0099536  
## 60647201647541301114 42 20 001 \0132129  
## 60663601655660031 11 478941579207002 0136455  
## 74750901746262331515782896 0202001 \0153837  
## 182 1000000008371 323326\81 82620181 07 601 71 11 478940155539  
## 818265098206021715148726967 210004 \0161029

Observe que a última linha do exemplo acima é justamente um dos registros identificados como problemáticos anteriormente, por apresentar valores numéricos no intervalo entre os caracteres 35 e 54, que deveria estar vazio.

Combinamos agora, num só objeto, as linhas que contemplam os três tipos de problemas definidos acima:

hh\_linhas\_problemas = c(hh\_linhas\_problemas\_1, hh\_linhas\_problemas\_2, hh\_linhas\_problemas\_3) %>%  
 unique() %>%  
 sort()

Trata-se, assim de 29 registros problemáticos, que deverão receber atenção focada – manualmente, através de leitura e interpretação, o que permitirá elencar qual o tipo de solução (caso exista alguma) pode ser aplicada em cada caso. Salvamos, deste modo, um arquivo contendo as informações completas e os testes sobre cada variável para todos aqueles registros identificados como problemáticos.

write.csv2(x = censo1960\_hh %>%   
 mutate(string = c60\_string\_hh) %>%  
 filter(num\_linha\_dom %in% hh\_linhas\_problemas) %>%  
 select(num\_linha\_dom, string, starts\_with("test\_"), contains\_invalid),  
 file = "Auxiliary Files/linhas\_problematicas\_domicilios.csv",  
 row.names = F)

Após tal avaliação detida, foi adicionada uma coluna de diagnóstico, indicando, para cada um dos registros problemáticos, a a gravidade e a naturezado problema encontrado.

checks\_hh <- read\_xlsx("Auxiliary Files/check\_line\_by\_line.xlsx", sheet = "households")  
freq(checks\_hh$diagnostico, plot = F)

## checks\_hh$diagnostico   
## Frequência  
## corrupted\_registry 1  
## del\_space\_between=2; insert\_space\_end=1; insert\_bar= 1 1  
## ok 27  
## Total 29  
## Percentual  
## corrupted\_registry 3.448  
## del\_space\_between=2; insert\_space\_end=1; insert\_bar= 1 3.448  
## ok 93.103  
## Total 100.000

Os registros marcados como ok são aqueles que trazem problemas em apenas uma variável, devido à presença de caracteres ou valores inválidos – no entanto, o restante dos valores do mesmo registro não aprensentam problemas. O registro marcado como del\_space\_between=2; insert\_space\_end=1; insert\_bar= 1 é um caso um pouco mais grave: apresenta, em duas posições, espaços adicionais separando valores válidos. Além disso, não contém a barra investida que usualmente antecede a informação sobre o ID da família. Mas trata-se de um problema com solução: remover os dois espaços adicionais entre valores, inserindo, em seguida a barra (com um espaço a antecedendo) resolve o problema. Trata-se, deste modo, de um registro corrompido, mas recuperável. Há certamente o suposto de que nenhum outro valor foi alterado. Mais adiante executaremos esses procedimentos de correção. Outros testes de consistência, contudo, devem ser realizados. É preciso avaliar a coerência entre respostas de diferentes questões.

* Para domicilios coletivos (V101 == 3), todas as variaveis entre V102 e V113 devem estar em branco – são questões que não se aplicam a esses registros. Todos as frequencias abaixo mostram que esse critério foi satisfeito:

censo1960\_hh %>%  
   
 # neste passo nao analisaremos os registros problematicos  
 filter(!(num\_linha\_dom %in% hh\_linhas\_problemas)) %>%   
 filter(V101 == 3) %>%  
 select(starts\_with("V")) %>%  
 select(V102:V113) %>%  
 map(., function(x) freq(x, plot=F))

## $V102  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V103  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V104  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V105  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V106  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V107  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V108  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V109  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V110  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V111  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V112  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100  
##   
## $V113  
## x   
## Frequência Percentual  
## 835 100  
## Total 835 100

* O cruzamento entre a especie de domicilio (v101: particular unico, particular com mais de uma familia ou coletivo) e o tipo de domicílio (v102: duravel, rustico, improvisado etc) deve apresentar um padrão específico: \* Se v101 = particular unico (1) ou 1a familia (2), v102 pode assumir quaisquer valores entre 4 e 7 \* Se v101 = coletivo (3), v102 deve ser deixada em branco \* Se v101 = 2a ou 3a familia ou boletim individual (4 ou 5), v102 deve ser deixada em branco

O cruzamento abaixo mostra que todas essas condições são satisfeitas.

with(censo1960\_hh %>% filter(!(num\_linha\_dom %in% hh\_linhas\_problemas)), {  
 table(V101, V102)  
})

## V102  
## V101 4 5 6 7  
## 1 0 125114 47710 49 9  
## 2 0 291 57 0 0  
## 3 835 0 0 0 0  
## 4 345 0 0 0 0  
## 5 28 0 0 0 0

* Para domicilios particulares improvisados (V102 == 6), as variaveis V103 a V113 devem permanecer em branco. Como podemos observar, essa condição também é satisfeita:

censo1960\_hh %>%  
 filter(!(num\_linha\_dom %in% hh\_linhas\_problemas)) %>%  
 filter(V102 == 6) %>%  
 select(starts\_with("V")) %>%  
 select(V103:V113) %>%  
 map(., function(x) freq(x, plot=F))

## $V103  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V104  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V105  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V106  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V107  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V108  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V109  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V110  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V111  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V112  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100  
##   
## $V113  
## x   
## Frequência Percentual  
## 49 100  
## Total 49 100

Observamos, deste modo, que com excessao dos 29 registros problematicos identificados, o arquivo de dados de domicilios parece consistente.

#### Corrigindo os registros de domicilios

Marcamos então os casos, de acordo com as avaliações feitas anteriormente:

1. Problema corrigido: O arquivo de dados original (txt) apresentava caracteres deslocados.
2. Problema não corrigido, mas ignorável: uma ou algumas variáveis apresentavam valores inválidos (não listados no dicionário)
3. Registro não problemático
4. Registro completamente corrompido.

A rotina abaixo executa os procedimentos necessários para marcar os casos segundo esses rótulos:

source("https://raw.githubusercontent.com/antrologos/ConsistenciaCenso1960Br/master/Code/Family\_diagnostics\_procedure.R")

Abriremos novamente o arquivo de dados em formato fixo aplicando o layout, e, desta vez, adicionando-lhe o resultados dos diagnósticos:

censo1960\_hh <- aplicaTesta\_layout(dados\_string = c60\_string\_hh,  
 input = input\_hh) %>%  
   
 mutate(  
 # adicionamos uma coluna que indica a linha onde o registro se encontra no arquivo de dados  
 num\_linha\_dom = num\_linha\_dom,  
   
 # Adicionamos uma coluna com os resultados dos procedimentos de diagnóstico  
 cem\_diagnosis\_dom = diagnosis) %>%   
 select(num\_linha\_dom, everything())

Marcamos então as linhas que contêm ao menos um erro por meio de uma variável específica:

censo1960\_hh$cem\_vars\_still\_problematic\_dom <- censo1960\_hh %>%   
 select(starts\_with("test\_")) %>%   
 mutate\_all(function(x) as.numeric(!(x))) %>%  
 mutate\_all(function(x) ifelse(is.na(x), 0, x)) %>%   
 rowSums(.)

Criamos então uma outra variável nova, que lista o nome das variaveis corrompidas em cada registro:

casos\_problematicos <- censo1960\_hh %>%  
 filter(cem\_vars\_still\_problematic\_dom > 0)   
  
teste\_problematicos <- casos\_problematicos %>%  
 select(starts\_with("test\_"), -test\_ID, -test\_barra)  
  
casos\_problematicos$cem\_problematic\_vars\_list\_dom = ""  
vars <- names(teste\_problematicos)  
for(var in vars){  
 problematic\_cases <- which(!teste\_problematicos[[var]])  
 var\_name = gsub(pattern = "test\_", replacement = "" , x = var)  
 casos\_problematicos[problematic\_cases, "cem\_problematic\_vars\_list\_dom"] =   
 paste(casos\_problematicos[problematic\_cases, ]$cem\_problematic\_vars\_list\_dom, var\_name)  
}  
  
casos\_problematicos$cem\_problematic\_vars\_list\_dom <- str\_trim(casos\_problematicos$cem\_problematic\_vars\_list\_dom)  
  
censo1960\_hh <- left\_join(x = censo1960\_hh,  
 y = casos\_problematicos %>%  
 select(num\_linha\_dom, cem\_problematic\_vars\_list\_dom),  
 by = "num\_linha\_dom")

Substituimos então os valores invalidos por missing:

num\_linha\_dom\_prob <- censo1960\_hh %>%  
 filter(cem\_vars\_still\_problematic\_dom >= 1) %>%  
 .$num\_linha\_dom  
  
for(i in num\_linha\_dom\_prob){  
 vars\_prob <- censo1960\_hh[censo1960\_hh$num\_linha\_dom == i, "cem\_problematic\_vars\_list\_dom"] %>%   
 unlist() %>%  
 strsplit(split = " ") %>%  
 unlist() %>%  
 str\_trim()  
 var = vars\_prob[2]  
 for(var in vars\_prob){  
 censo1960\_hh[censo1960\_hh$num\_linha\_dom == i,][[var]] <- NA   
 }  
}

Por fim, compilando o banco de dados consistido das famílias, em seu formato preliminar. Removemos as colunas de teste, Transformamos as variáveis originais em numéricas e acoplamos as variáveis sintéticas de diagnóstico que criamos.

censo1960\_hh\_numeric <- censo1960\_hh%>%  
 select(-starts\_with("test\_"), - barra) %>%  
 select(-cem\_diagnosis\_dom, -cem\_problematic\_vars\_list\_dom,  
 -starts\_with("place\_")) %>%  
 mutate\_all(as.numeric)  
  
censo1960\_hh\_character <- censo1960\_hh %>%  
 select(cem\_diagnosis\_dom, cem\_problematic\_vars\_list\_dom,  
 starts\_with("place\_"))  
  
censo1960\_hh\_preliminar <- cbind(censo1960\_hh\_numeric, censo1960\_hh\_character) %>% as\_tibble()

### Registros de pessoas

A consistência dos registros de pessoas seguirá a mesma ordem e estrutura daquela aplicada para os registros de famílias. Primeiramente, carregamos o arquivo de layout, criado a partir das sintaxes:

input\_person <- read\_xlsx(input\_file, sheet = "Person\_open")

No arquivo de dados original, ainda em formato de texto com caracteres de posições fixas, selecionamos apenas os dados de pessoas (valores diferentes de 1 na coluna 17):

c60\_string\_pess <- c60\_string[which(teste\_char17\_dom\_pess != 1)]  
num\_linha\_pess <- (1:n\_linhas)[which(teste\_char17\_dom\_pess != 1)]

Aplicamos então o arquivo de layout para separar as colunas e executamos testes para averiguar se os valores encontrados no arquivo de dados estão contemplados no dicionário de códigos:

censo1960\_pess <- aplicaTesta\_layout(dados\_string = c60\_string\_pess,  
 input = input\_person) %>%  
 mutate(num\_linha\_pess = num\_linha\_pess) %>%   
 select(num\_linha\_pess, everything())

Identificamos então as linhas que contêm ao menos um erro:

censo1960\_pess$contains\_invalid <- censo1960\_pess %>%   
 select(starts\_with("test\_")) %>%   
 mutate\_all(function(x) as.numeric(!(x))) %>%  
 mutate\_all(function(x) ifelse(is.na(x), 0, x)) %>%   
 rowSums(.)

Gravamos então, num objeto à parte, os identificadores únicos (números das linhas) dos registros problematicos:

# Problema 1: presenca de caracteres nao listados como validos pelo dicionario de codigo  
pess\_linhas\_problemas\_1 <- censo1960\_pess %>%   
 filter(contains\_invalid > 0) %>%  
 .$num\_linha\_pess

As seguintes linhas do arquivo de dados original (89 no total) apresentaram problemas desta natureza. Alguns exemplos desse desse tipo de ocorrência são:

writeLines(c60\_string[pess\_linhas\_problemas\_1[1:10]])

## 02024301021102572117128571"912183110085404048352425156\0001403  
## 17180603174940273329105532392000311 \0024733  
## \212135072103211531 291 08542092000052 0029831  
## 2121190121536143352812554209142031 100953000034 0030734  
## 2121190121536245352911 857 209200014200000000034 0030823  
## 212135072103211531 2813554209200016440648060503191281550032581  
## 212138052189801725171255720R20003110095702025332321176\0035191  
## 212144012199206635281605420 200031100625000034 \0035911  
## 212166X72236615425171505520 20003110040010055332321207\0037520  
## 31 M 1BA33HIA \0046690

Observe que no último caso do exemplo acima, não há apenas dígitos, mas também caracteres do alfabeto, formando, aproximadamente, a palavra “BAHIA”. Trate-se muito provavelmente de um erro de gravação em consequência das mudanças de mídia, formato e esquemas de codificação. O dicionário de códigos que acompanhava o conjunto de dados recebidos como doação pelo Centro de Estudos da Metrópole menciona que ao menos um processo do tipo teria sido realizado para a amostra de 25%:

O ARQUIVO ORIGINAL ESTAVA GRAVADO EM FORMATO EBCDIC (FORMATO PARA MÁQINAS IBM) E ALGUMAS VARIÁVEIS EM CÓDIGO BINÁRIO. OS DADOS FORAM RECUPERADOS E GRAVADOS EM FORMATO ASCII. ESTE TRABALHO FOI REALIZADO PELO POPULATION RESEARCH CENTER DA UNIVERSIDADE DO TEXAS–AUSTIN, EM COMUM ACORDO COM O IBGE.

Há razões para crer que procedimentos de leitura, gravação e conversão do mesmo gênero teriam ocorrido também para a amostra de 1,27%.

Passamos à identificação do segundo tipo de problema: a presença de caracteres inválidos:

# Problema 2: presenca de outros tipos de caracteres invalidos  
pess\_linhas\_problemas\_2 = grep(pattern = "[[:alpha:]]|[[:cntrl:]]|[[:punct:]]" ,   
 x = gsub(pattern = "[\\]",   
 replacement = "",   
 x = c60\_string\_pess))  
pess\_linhas\_problemas\_2 = censo1960\_pess[pess\_linhas\_problemas\_2,"num\_linha\_pess"] %>% unlist()

No todo, foram encontrados 36 desse tipo. São exemplos desta ocorrência:

writeLines(c60\_string[pess\_linhas\_problemas\_2[1:10]])

## 02024301021102572117128571"912183110085404048352425156\0001403  
## 212138052189801725171255720R20003110095702025332321176\0035191  
## 212166X72236615425171505520 20003110040010055332321207\0037520  
## 31 M 1BA33HIA \0046690  
## 54 M 4GU35ANABARA \0097157  
## 54553230552620473319011'41292000 \0104113  
## 5455413155302026211712'5423908231620095702029385125176\0104200  
## 54554131553020862117144'708909081620064006069361328145\0104260  
## 54554131553021433129102'41292000 \0104317  
## 54554131553021443119116'4129200008210000000035 \0104318

Repare que novamente nos deparamos com o registro que contém caracteres formando, de modo aproximado, a palavra “BAHIA” (UF da qual o registro de fato faria parte, de acordo com os dois dígitos iniciais). Ou seja, os dois métodos de identificação de erros capturam os problemas existentes nessa linha. Além disso, observamos que um caso bastante semelhante ocorre logo em seguida, no qual se forma, também de forma aproximada, a palavra “GUANABARA”. Esses são dois registros completamente corrompidos, que, por precaução, deverão ser ignorados. **Casos dessa natureza terão suas variáveis marcadas como missing e serão assinalados, em variável apropriada, criada pelo Centro de Estudos da Metrópole, como registros inutilizáveis para a maioria das análises**.

Por fim, buscamos pela presença de caracteres adicionais, separando variáveis e valores que deveriam ser adjacentes nos dados:

# Problema 3: presenca de espacos (até 3) entre caracteres  
pess\_linhas\_problemas\_3 = grep(pattern = "[[:digit:]][[:blank:]]{1,3}[[:digit:]]",  
 x = c60\_string\_pess)  
pess\_linhas\_problemas\_3 = censo1960\_pess[pess\_linhas\_problemas\_3,"num\_linha\_pess"] %>% unlist()

São exemplos desse tipo de ocorrência:

writeLines(c60\_string[pess\_linhas\_problemas\_3[1:10]])

## 14164101152 00673519104570792000 \0021501  
## \212135072103211531 291 08542092000052 0029831  
## 2121190121536143352812554209142031 100953000034 0030734  
## 2121190121536245352911 857 209200014200000000034 0030823  
## 212135072103211531 2813554209200016440648060503191281550032581  
## 212144012199206635281605420 200031100625000034 \0035911  
## 212166X72236615425171505520 20003110040010055332321207\0037520  
## 2122620722 34069352911354209200015200000000034 \0040955  
## 31 1 80731608050352911557059200031100000000034 \0046690  
## 4040940140880001311 1 954169200017210400130776 \0064744

Compilamos num objeto as linhas detectadas como problemáticas segundo as três definições acima e salvamos num arquivo separado os dados e diagnósticos sobre os registros de pessoas, que serão avaliados manualmente:

pess\_linhas\_problemas = c(pess\_linhas\_problemas\_1, pess\_linhas\_problemas\_2, pess\_linhas\_problemas\_3) %>%  
 unique() %>%  
 sort()  
  
write.csv2(x = censo1960\_pess %>%   
 mutate(string = c60\_string\_pess) %>%  
 filter(num\_linha\_pess %in% pess\_linhas\_problemas) %>%  
 select(num\_linha\_pess, string, starts\_with("test\_"), contains\_invalid),  
 file = "Auxiliary Files/linhas\_problematicas\_pessoas.csv",  
 row.names = F)

Trata-se, assim, de um banco de dados consistido dos registros de família. Resta ainda construir o banco de domicílios e realizar procedimentos de consistência, comparando-o com os dados de pessoas (a serem analisados na seção seguinte).

#### Corrigindo os registros de pessoas

Uma vez realizado o diagnóstico manual, carregamos os seus resultados:

checks\_pess <- read\_xlsx("Auxiliary Files/check\_line\_by\_line.xlsx", sheet = "persons")

E, tal como fizemos anteriormente, executamos uma breve rotina que assinala os casos problemáticos:

source("https://raw.githubusercontent.com/antrologos/ConsistenciaCenso1960Br/master/Code/Persons\_diagnostics\_procedure.R")

Aplicamos então o arquivo de layout sobre os registros de pessoas para gerar novamente um banco de dados com separação de colunas – adicionando a variável de diagnóstico, recém-criada.

censo1960\_pess <- aplicaTesta\_layout(dados\_string = c60\_string\_pess,  
 input = input\_person) %>%  
 mutate(num\_linha\_pess = num\_linha\_pess,  
 cem\_diagnosis\_pess = diagnosis) %>%   
 select(num\_linha\_pess, everything())

Marcamos as linhas que contem ao menos um erro:

censo1960\_pess$cem\_vars\_still\_problematic\_pess <- censo1960\_pess %>%   
 select(starts\_with("test\_")) %>%   
 mutate\_all(function(x) as.numeric(!(x))) %>%  
 mutate\_all(function(x) ifelse(is.na(x), 0, x)) %>%   
 rowSums(.)

E criamos a coluna adicional que identifica e nomeia todas as variaveis problematicas em cada registro:

casos\_problematicos <- censo1960\_pess %>%  
 filter(cem\_vars\_still\_problematic\_pess > 0)   
  
teste\_problematicos <- casos\_problematicos %>%  
 select(starts\_with("test\_"), -test\_ID, -test\_barra)  
  
casos\_problematicos$cem\_problematic\_vars\_list\_pess = ""  
vars <- names(teste\_problematicos)  
for(var in vars){  
 problematic\_cases <- which(!teste\_problematicos[[var]])  
   
 var\_name = gsub(pattern = "test\_", replacement = "" , x = var)  
   
 casos\_problematicos[problematic\_cases, "cem\_problematic\_vars\_list\_pess"] =   
 paste(casos\_problematicos[problematic\_cases, ]$cem\_problematic\_vars\_list\_pess, var\_name)  
}  
  
casos\_problematicos$cem\_problematic\_vars\_list\_pess <- str\_trim(casos\_problematicos$cem\_problematic\_vars\_list\_pess)  
  
censo1960\_pess <- left\_join(x = censo1960\_pess,  
 y = casos\_problematicos %>%  
 select(num\_linha\_pess, cem\_problematic\_vars\_list\_pess),  
 by = "num\_linha\_pess")

Ao fim, os casos corrompidos terão marcações como as do exemplo abaixo:

censo1960\_pess %>%  
 filter(cem\_problematic\_vars\_list\_pess != "") %>%  
 select(num\_linha\_pess, cem\_problematic\_vars\_list\_pess)

## # A tibble: 81 x 2  
## num\_linha\_pess cem\_problematic\_vars\_list\_pess   
## <int> <chr>   
## 1 9817 V207   
## 2 162537 V206   
## 3 194609 UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207~  
## 4 224345 V208   
## 5 228629 V208   
## 6 238423 V208   
## 7 293555 UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207~  
## 8 293556 V116   
## 9 388894 UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207~  
## 10 388895 UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207~  
## # ... with 71 more rows

Substituiremos então os valores invalidos por missing. Uma estratégia alternativa a esse passo seria lançar mão de métodos probabilísticos de imputação de dados faltantes (como a Imputação Múltipla, por exemplo). No entanto, decidimos por deixar a cargo dos futuros usuários a avaliação da necessidade desse procedimento.

num\_linha\_pess\_prob <- censo1960\_pess %>%  
 filter(cem\_vars\_still\_problematic\_pess >= 1) %>%  
 .$num\_linha\_pess  
  
for(i in num\_linha\_pess\_prob){  
 vars\_prob <- censo1960\_pess[censo1960\_pess$num\_linha\_pess == i, "cem\_problematic\_vars\_list\_pess"] %>%   
 unlist() %>%  
 strsplit(split = " ") %>%  
 unlist()  
   
 for(var in vars\_prob){  
 censo1960\_pess[censo1960\_pess$num\_linha\_pess == i,][[var]] <- NA   
 }   
}

Compilamos, por fim, uma versão “semi-final” do banco de pessoas, após os procedimentos de consistência. Eles deverão ainda ser combinados com os dados de domicílios e as informações que se repetem nos dois tipos de registro (como UF, Município e área de residência) deverão ainda ser contrastadas.

censo1960\_pess\_numeric <- censo1960\_pess%>%  
 select(-starts\_with("test\_"), - barra) %>%   
 select(-cem\_diagnosis\_pess, -cem\_problematic\_vars\_list\_pess,  
 -starts\_with("place\_")) %>%  
 mutate\_all(as.numeric)  
  
  
censo1960\_pess\_character <- censo1960\_pess %>%  
 select(cem\_diagnosis\_pess, cem\_problematic\_vars\_list\_pess,  
 starts\_with("place\_"))  
  
censo1960\_pess\_semifinal <- cbind(censo1960\_pess\_numeric, censo1960\_pess\_character) %>% as\_tibble()

## Identificando domicílios

O primeiro passo para a criação de um banco de domicílios a partir dos dados de famílias e pessoas é a elaboração de um ID único para cada domícilio. Como vimos anterioremente, as variáveis V101 e V102 são essenciais para esse passo, por identificarem os tipos de família e composição dos moradores.

Observamos que dois registros do banco de famílias, contudo, não possuem informação para a V101:

censo1960\_hh\_preliminar %>%   
 filter(is.na(V101 == 0)) %>%  
 select(ID)

## # A tibble: 2 x 1  
## ID  
## <dbl>  
## 1 155539  
## 2 165232

O registro com ID 155539 será tratado adiante.

A família com ID 165232, por sua vez, tinha informação inválida na V101 (valor 0). Em passos anteriores, esse valor foi substituído por missing. Alteraremos esse campo para 1 (código referente aos “Domicílios Particulares Únicos”). Avaliando os casos que precedem e sucedem esse registro, ele não parece estar ligado a outros domicílios (i.e. não parece ser uma família principal, secundária ou terciária). Além disso, a estrutura familiar de seus moradores não é indicativa de um domicílio coletivo.

censo1960\_hh\_preliminar <- censo1960\_hh\_preliminar %>%   
 mutate(V101 = ifelse(ID == 165232, 1, V101))

Outra inconsistência que devemos resolver neste momento refere-se à presença de um caso de missing para a variavel sobre tipo de registro (que diferenciava pessoas e domicílios). Esse valor missing era anteriormente inexistente – foi atribuído em meio aos procedimentos de consistência, pelo fato de que o registro ao qual se refere estava majoritariamente corrompido. Assumiremos que se trata mesmo de um domicílio, atribuindo-lhe o valor para a variável em questão:

censo1960\_hh\_preliminar$REC\_TYPE = 1

Criamos então variaveis separadas para os IDs de domicilio e de familia, assinalando ainda os tipos de família encontrados:

censo1960\_hh\_preliminar <- censo1960\_hh\_preliminar %>%  
 arrange(ID, V101) %>%  
 mutate(cem\_IDdomicilio = ifelse(V101 %in% c(1,2,3), ID, NA),  
 cem\_tipofamilia = ifelse(V101 %in% c(1,2), 1, ifelse(V101 %in% c(4,5), 2, NA)),  
 n\_linha = 1:n())

A codificação da variável “cem\_tipofamilia” será a seguinte:

* **1** - Familia principal ou unica
* **2** - Outras familias
* **NA** - Nao se aplica (domicilios coletivos)

As linhas sem valores válidos para a variável cem\_IDdomicilio devem, deste modo, dizer respeito a famílias conviventes. Por definição, esses registros devem reecber o mesmo ID de domicílios das famílias principais com as quais co-habitam. Para isso, primeiramente, selecionamos esses casos:

linhas\_familias <- censo1960\_hh\_preliminar %>%  
 filter(is.na(cem\_IDdomicilio)) %>%   
 .$n\_linha

Então, para as familias secundarias, copiamos a informação sobre ID de Domicílio do caso anterior (que deve ser, necessariamente, uma família principal):

for(linha\_familia in linhas\_familias){  
 caso\_anterior <- censo1960\_hh\_preliminar[(linha\_familia - 1) , "cem\_tipofamilia"] == 1   
 dim(caso\_anterior) <- NULL  
   
 if( caso\_anterior & !is.na(caso\_anterior)){  
 censo1960\_hh\_preliminar[linha\_familia , "cem\_IDdomicilio"] <- censo1960\_hh\_preliminar[(linha\_familia-1) , "cem\_IDdomicilio"]  
 }  
}

Repetimos o procedimento para as familias terciarias, copiando a informação do caso anterior – mas agora, apenas se o caso anterior for o de uma família secundária:

linhas\_familias = which(is.na(censo1960\_hh\_preliminar$cem\_IDdomicilio))  
for(linha\_familia in linhas\_familias){  
 caso\_anterior <- censo1960\_hh\_preliminar[(linha\_familia - 1) , "V101"] == 4  
 proprio\_caso <- censo1960\_hh\_preliminar[(linha\_familia ) , "V101"] == 5  
   
 dim(caso\_anterior) <- NULL  
 dim(proprio\_caso) <- NULL  
   
 if( caso\_anterior & proprio\_caso & !is.na(caso\_anterior) & !is.na(proprio\_caso)){  
 censo1960\_hh\_preliminar[linha\_familia , "cem\_IDdomicilio"] <- censo1960\_hh\_preliminar[(linha\_familia-1) , "cem\_IDdomicilio"]  
 }  
}

Por fim, a antiga variável ID converte-se agora em ID apenas das famílias. E já podemos apagar a variável auxiliar “n\_linha”, criada para este passo:

censo1960\_hh\_preliminar <- censo1960\_hh\_preliminar %>%  
 rename(cem\_IDfamilia = ID)  
censo1960\_hh\_preliminar$n\_linha = NULL

Examinado agora o registro com ID de família igual a 155539, o que observamos é que ele parece constituir um domicilio autonomo, não faz parte do domicilio 155537, que o precede. Trata-se, na realidade, de um registro de domicilio com dados corrompidos. Mais adiante, quando constrastarmos suas informações com aquelas existentes nos registros das pessoas que nele habitam, detectaremos a existência de discrepâncias. Ainda assim, devemos mantê-lo no banco de dados. Neste passo, é preciso separá-lo do domicílio anterior:

censo1960\_hh\_preliminar <- censo1960\_hh\_preliminar %>%  
 mutate(cem\_IDdomicilio = ifelse(cem\_IDfamilia == 155539, 155539, cem\_IDdomicilio),  
 V101 = ifelse(cem\_IDfamilia == 155539, 1, V101))

Se identificamos corretamente os domicílios, não deve haver variação de suas características estruturais entre as famílias conviventes. Ou seja: famílias que moram juntas não devem apresentar informações discrepantes com respeito ao número de cômodos, presença de energia elétrica, água etc. Testamos isso por meio do procedimento abaixo. A variância dessas características deve ser igual a zero:

test = censo1960\_hh\_preliminar %>%  
 select(-num\_linha\_dom, -V101, -cem\_IDfamilia, -cem\_vars\_still\_problematic\_dom,  
 -cem\_diagnosis\_dom, -cem\_problematic\_vars\_list\_dom, -cem\_tipofamilia) %>%  
 select\_if(function(x) !is.character(x)) %>%  
 group\_by(cem\_IDdomicilio) %>%  
 summarise\_all(stats::var, na.rm= T)  
   
domicilios\_problema = NULL  
for(i in 2:ncol(test)){  
 domicilios\_problema <- c(domicilios\_problema, test[which(test[,i] > 0 & !is.na(test[,2])),"cem\_IDdomicilio"] %>% unlist())  
}  
  
domicilios\_problema <- domicilios\_problema %>% unique()

Nenhum problema foi encontrado: não há domicilios que apresentem variação de caracteristicas entre familias:

censo1960\_hh\_preliminar %>%  
 filter(cem\_IDdomicilio %in% domicilios\_problema)

Temos então um banco de famílias, com marcações dos IDs dos domicílios aos quais pertencem.

## Importando IDs de domicílio para o banco de pessoas

No banco de famílias, devemos então checar se o número de linhas é identico ao número de IDs:

length(censo1960\_hh\_preliminar$cem\_IDfamilia)  
## [1] 174467  
length(censo1960\_hh\_preliminar$cem\_IDfamilia %>% unique()) #ok  
## [1] 174467

No banco de pessoas, checamos se o número de IDs de familia é identico ao número de linhas no banco de famílias:

censo1960\_pess\_semifinal <- censo1960\_pess\_semifinal %>%  
 rename(cem\_IDfamilia = ID)  
length(censo1960\_pess\_semifinal$cem\_IDfamilia %>% unique()) # ok  
## [1] 174467

Por fim, verificamos se todos os valores listados como IDs de familia no arquivo de pessoas estão de fato contemplados no arquivo de famílias. E não encontramos nenhuma discrepância:

sum( !(censo1960\_pess\_semifinal$cem\_IDfamilia %in% censo1960\_hh\_preliminar$cem\_IDfamilia) ) # ok!

## [1] 0

Como o pareamento existe, podemos trazer a informação recém-criada sobre ID de domicílio para os registros de indivíduos:

censo1960\_pess\_semifinal <- left\_join(x = censo1960\_pess\_semifinal,  
 y = censo1960\_hh\_preliminar %>%  
 select(cem\_IDdomicilio,cem\_IDfamilia, V101),  
 by = "cem\_IDfamilia")

## Construindo um banco de domicílios

Para construir um banco de domicílios daquele sobre famílias, devemos remover os registros de familias secundarias e terciarias, mantendo apenas as unicas, principais e os domicilios coletivos. Podemos também remover algumas variáveis desnecessárias:

censo1960\_hh\_semifinal = censo1960\_hh\_preliminar %>%  
 filter(!(V101 %in% c(4,5))) %>%   
 select(-cem\_tipofamilia, -cem\_IDfamilia, -REC\_TYPE)

Problemas de gravação em dois registros adjacentes voltam agora a ser relevantes. Trata-se das linhas 293555 e 293556 do arquivo original:

1. 31 M 1BA33HIA \0046690
2. 31 1 80731608050352911557059200031100000000034 \0046690

Visualisando esses dois registros dentro de um contexto maior, observamos que estão justamente na fronteira entre dois estados: Sergipe (UF=30) e Bahia (UF=31). Então, no entanto, sendo atribuidos ao domicilio de Sergipe dos casos que os antecedem: de ID=0046690. Há, deste modo, uma constradição entre a informação do ID e a da UF.

writeLines(c60\_string[293552:293560])

## 3030950130378219151478389680203001 \0046690  
## 303095013037821925271195427920003110010001015393325137\0046690  
## 30309501303782193529105542792000311 \0046690  
## 31 M 1BA33HIA \0046690  
## 31 1 80731608050352911557059200031100000000034 \0046690  
## 3131110131344001131578380680208004 \0046691  
## 313111013134400123171605705920003110062312096366423157\0046691  
## 3131110131344001332815654059200031100623120934 \0046691  
## 3131110131344001331912657059200015200000000031 \0046691

A palavra “BAHIA”, presente no registro 293555, é forte evidência de que se trataria, de fato, de um caso daquela UF. Devemos então estabelecer uma regras de desambiguação desses dois registros – **isso afetará o número de casos únicos de domicílios e famílias**. Mas uma observação pode facilitar a decisão. Os IDs de família (caracteres nas posições 56-62) parecem ter sido criados a posteriori, não sendo dados originais. Se isso for verdade, podem conter erros. A numeracao daqueles IDs parece seguir sempre uma mesma lógica: seguindo a ordem dos casos no banco de dados, inicia-se um novo valor que o caracteres da posição 17 assume o valor 1 (que seria indicativo de família/domicílio). Como, nesse caso, o registro da linha 293555 está corrompido, não teria havido uma nova contagem do ID. Provavelmente, por esta razão, o numero do domicilio continuou o mesmo dos registros anteriores.

Para iniciar um novo ID temos então duas saídas:

1. Assumir que o registro da linha 293555 é ele mesmo um domicílio (o que poderia ser feito com substr(c60\_string[293555], 17, 17) <- "1").
2. Criar uma nova linha no banco de dados, representando o domicílio faltante. Com isso, os registros corrompidos seriam entendidos como “pessoas”.

Ambas as soluções envolvem modificação drástica dos dados originais – o que não parece ser recomendado. No primeiro caso, além disso, o registro da linha 293556 seria compreendido como um indivíduo morando sozinho. Se as demais informações da linha forem consideradas, isso introduziria outros problemas: trataria-se de uma pessoa na posição de “filho” (v203=9, caractere 20), com apenas 15 anos de idade(caracteres 22 e 23) – o que introduziria uma nova inconsistência. Essas duas soluções parecem ser por demais radicais. **Por esta razão, decidimos não proceder nenhuma alteração, mantendo a inconsistência original**.

## Imputação de valores faltantes

Há variáveis que se repetem tanto nos registros de pessoas e domicílios: UF, V116 (Município) e V118 (área de residência, se rural ou urbana). Esse fato permite que, nos casos de informação faltante ou corrompida, seja possível copiar o dado do outro tipo de registro. Iniciamos então pela importação dos dados de domicílios para os registros de pessoas:

censo1960\_pess\_semifinal <- left\_join(x = censo1960\_pess\_semifinal,   
 y = censo1960\_hh\_semifinal %>%  
 select(-V101),   
 by = "cem\_IDdomicilio") %>%  
 rename(UF\_pess = UF.x,  
 UF\_dom = UF.y,  
 V116\_pess = V116.x,  
 V116\_dom = V116.y,  
 V118\_pess = V118.x,  
 V118\_dom = V118.y)

Identificamos agora os casos em que há missing apenas numa das fontes de informacao (ou pessoas ou domicilios):

censo1960\_pess\_semifinal <- censo1960\_pess\_semifinal %>%  
 mutate(cem\_UF\_missingDom\_notmissingPerson = as.numeric(is.na(UF\_dom) & !is.na(UF\_pess) ),  
 cem\_UF\_missingPerson\_notmissingDom = as.numeric(is.na(UF\_pess) & !is.na(UF\_dom) ),  
 cem\_V116\_missingDom\_notmissingPerson = as.numeric(is.na(V116\_dom) & !is.na(V116\_pess)),  
 cem\_V116\_missingPerson\_notmissingDom = as.numeric(is.na(V116\_pess) & !is.na(V116\_dom) ),  
 cem\_V118\_missingDom\_notmissingPerson = as.numeric(is.na(V118\_dom) & !is.na(V118\_pess)),  
 cem\_V118\_missingPerson\_notmissingDom = as.numeric(is.na(V118\_pess) & !is.na(V118\_dom))  
 )

Iniciamos o procedimento de imputação pela cópia de informações: quando há missing na variável de domicílio, lhe atribuímos o valor observado no registro de pessoa e vice-versa:

censo1960\_pess\_semifinal <- censo1960\_pess\_semifinal %>%   
 mutate(UF\_dom = ifelse(cem\_UF\_missingDom\_notmissingPerson == 1, UF\_pess, UF\_dom),  
 UF\_pess = ifelse(cem\_UF\_missingPerson\_notmissingDom == 1, UF\_dom, UF\_pess),  
   
 V116\_dom = ifelse(cem\_V116\_missingDom\_notmissingPerson == 1, V116\_pess, V116\_dom),  
 V116\_pess = ifelse(cem\_V116\_missingPerson\_notmissingDom == 1, V116\_dom, V116\_pess),  
   
 V118\_dom = ifelse(cem\_V118\_missingDom\_notmissingPerson == 1, V118\_pess, V118\_dom),  
 V118\_pess = ifelse(cem\_V118\_missingPerson\_notmissingDom == 1, V118\_dom, V118\_pess)  
 )

Deletando então as variaveis temporarias utilizadas nesse procedimento:

censo1960\_pess\_semifinal$cem\_UF\_missingDom\_notmissingPerson = NULL  
censo1960\_pess\_semifinal$cem\_UF\_missingPerson\_notmissingDom = NULL  
censo1960\_pess\_semifinal$cem\_V116\_missingDom\_notmissingPerson = NULL  
censo1960\_pess\_semifinal$cem\_V116\_missingPerson\_notmissingDom = NULL  
censo1960\_pess\_semifinal$cem\_V118\_missingDom\_notmissingPerson = NULL  
censo1960\_pess\_semifinal$cem\_V118\_missingPerson\_notmissingDom = NULL

Podemos ainda atribuir valores para pessoas com missing, tomando como doadores as informações de outroso indivíduos residentes no mesmo domicílio.

#### Identificando casos de missing na variavel de UF

censo1960\_pess\_semifinal = data.table(censo1960\_pess\_semifinal)  
censo1960\_pess\_semifinal[ , num\_NA\_UF\_dom := sum(is.na(UF\_dom)), by = cem\_IDfamilia]  
censo1960\_pess\_semifinal[ , num\_NA\_UF\_pess := sum(is.na(UF\_pess)), by = cem\_IDfamilia]  
censo1960\_pess\_semifinal[ num\_NA\_UF\_dom > 0 | num\_NA\_UF\_pess > 0 , ]

## num\_linha\_pess UF\_pess V116\_pess REC\_TYPE V118\_pess V202 V203 V204 AGE  
## 1: 951431 NA NA NA NA NA NA NA NA  
## 2: 951432 NA NA NA NA NA NA NA NA  
## 3: 951433 NA NA NA NA NA NA NA NA  
## 4: 951434 81 8006 3 5 1 9 1 3  
## 5: 951435 81 8006 3 5 2 9 1 6  
## V205 V206 V207 V208 V209 V299 V210 V211 V212 V213 V214 V215 V216 V217  
## 1: NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## 2: NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## 3: NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## 4: 5 4 24 9 2 0 0 NA NA NA NA NA NA NA  
## 5: 5 7 24 9 2 0 0 3 1 1 NA NA NA NA  
## V218 V219 V220 V221 V223 V223B V224 cem\_IDfamilia  
## 1: NA NA NA NA NA NA NA 155539  
## 2: NA NA NA NA NA NA NA 155539  
## 3: NA NA NA NA NA NA NA 155539  
## 4: NA NA NA NA NA NA NA 155539  
## 5: NA NA NA NA NA NA NA 155539  
## cem\_vars\_still\_problematic\_pess cem\_diagnosis\_pess  
## 1: 15 4  
## 2: 15 4  
## 3: 15 4  
## 4: 0 3  
## 5: 0 3  
## cem\_problematic\_vars\_list\_pess  
## 1: UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207 V208 V209 V299 V210  
## 2: UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207 V208 V209 V299 V210  
## 3: UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207 V208 V209 V299 V210  
## 4: <NA>  
## 5: <NA>  
## place\_holder\_07\_08.x place\_holder\_09\_16.x cem\_IDdomicilio V101  
## 1: 155539 1  
## 2: 155539 1  
## 3: 155539 1  
## 4: 05 82402101 155539 1  
## 5: 05 82402103 155539 1  
## num\_linha\_dom UF\_dom V116\_dom V118\_dom V102 V103 V104 V105 V106 V107  
## 1: 951430 NA NA NA NA NA NA NA NA NA  
## 2: 951430 NA NA NA NA NA NA NA NA NA  
## 3: 951430 NA NA NA NA NA NA NA NA NA  
## 4: 951430 81 8006 5 NA NA NA NA NA NA  
## 5: 951430 81 8006 5 NA NA NA NA NA NA  
## V108 V109 V110 V111 V112 V113 cem\_vars\_still\_problematic\_dom  
## 1: NA NA NA NA NA NA 5  
## 2: NA NA NA NA NA NA 5  
## 3: NA NA NA NA NA NA 5  
## 4: NA NA NA NA NA NA 5  
## 5: NA NA NA NA NA NA 5  
## cem\_diagnosis\_dom cem\_problematic\_vars\_list\_dom place\_holder\_07\_08.y  
## 1: 4 UF V116 REC\_TYPE V118 V101   
## 2: 4 UF V116 REC\_TYPE V118 V101   
## 3: 4 UF V116 REC\_TYPE V118 V101   
## 4: 4 UF V116 REC\_TYPE V118 V101   
## 5: 4 UF V116 REC\_TYPE V118 V101   
## place\_holder\_09\_16.y place\_holder\_32\_32 place\_holder\_35\_54  
## 1:   
## 2:   
## 3:   
## 4:   
## 5:   
## num\_NA\_UF\_dom num\_NA\_UF\_pess  
## 1: 3 3  
## 2: 3 3  
## 3: 3 3  
## 4: 3 3  
## 5: 3 3

O domicílio/família 155539 era um caso de registro corrompido para a variavel UF. Lançamos mão, então de informações advindas de casos doadores:

censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, UF\_pess := 81]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, UF\_dom := 81]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, V116\_pess := 8006]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, V116\_dom := 8006]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, V118\_pess := 5]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, V118\_dom := 5]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539, V101 := 1]   
censo1960\_pess\_semifinal[cem\_IDfamilia == 155539]

## num\_linha\_pess UF\_pess V116\_pess REC\_TYPE V118\_pess V202 V203 V204 AGE  
## 1: 951431 81 8006 NA 5 NA NA NA NA  
## 2: 951432 81 8006 NA 5 NA NA NA NA  
## 3: 951433 81 8006 NA 5 NA NA NA NA  
## 4: 951434 81 8006 3 5 1 9 1 3  
## 5: 951435 81 8006 3 5 2 9 1 6  
## V205 V206 V207 V208 V209 V299 V210 V211 V212 V213 V214 V215 V216 V217  
## 1: NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## 2: NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## 3: NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
## 4: 5 4 24 9 2 0 0 NA NA NA NA NA NA NA  
## 5: 5 7 24 9 2 0 0 3 1 1 NA NA NA NA  
## V218 V219 V220 V221 V223 V223B V224 cem\_IDfamilia  
## 1: NA NA NA NA NA NA NA 155539  
## 2: NA NA NA NA NA NA NA 155539  
## 3: NA NA NA NA NA NA NA 155539  
## 4: NA NA NA NA NA NA NA 155539  
## 5: NA NA NA NA NA NA NA 155539  
## cem\_vars\_still\_problematic\_pess cem\_diagnosis\_pess  
## 1: 15 4  
## 2: 15 4  
## 3: 15 4  
## 4: 0 3  
## 5: 0 3  
## cem\_problematic\_vars\_list\_pess  
## 1: UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207 V208 V209 V299 V210  
## 2: UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207 V208 V209 V299 V210  
## 3: UF V116 REC\_TYPE V118 V202 V203 V204 AGE V205 V206 V207 V208 V209 V299 V210  
## 4: <NA>  
## 5: <NA>  
## place\_holder\_07\_08.x place\_holder\_09\_16.x cem\_IDdomicilio V101  
## 1: 155539 1  
## 2: 155539 1  
## 3: 155539 1  
## 4: 05 82402101 155539 1  
## 5: 05 82402103 155539 1  
## num\_linha\_dom UF\_dom V116\_dom V118\_dom V102 V103 V104 V105 V106 V107  
## 1: 951430 81 8006 5 NA NA NA NA NA NA  
## 2: 951430 81 8006 5 NA NA NA NA NA NA  
## 3: 951430 81 8006 5 NA NA NA NA NA NA  
## 4: 951430 81 8006 5 NA NA NA NA NA NA  
## 5: 951430 81 8006 5 NA NA NA NA NA NA  
## V108 V109 V110 V111 V112 V113 cem\_vars\_still\_problematic\_dom  
## 1: NA NA NA NA NA NA 5  
## 2: NA NA NA NA NA NA 5  
## 3: NA NA NA NA NA NA 5  
## 4: NA NA NA NA NA NA 5  
## 5: NA NA NA NA NA NA 5  
## cem\_diagnosis\_dom cem\_problematic\_vars\_list\_dom place\_holder\_07\_08.y  
## 1: 4 UF V116 REC\_TYPE V118 V101   
## 2: 4 UF V116 REC\_TYPE V118 V101   
## 3: 4 UF V116 REC\_TYPE V118 V101   
## 4: 4 UF V116 REC\_TYPE V118 V101   
## 5: 4 UF V116 REC\_TYPE V118 V101   
## place\_holder\_09\_16.y place\_holder\_32\_32 place\_holder\_35\_54  
## 1:   
## 2:   
## 3:   
## 4:   
## 5:   
## num\_NA\_UF\_dom num\_NA\_UF\_pess  
## 1: 3 3  
## 2: 3 3  
## 3: 3 3  
## 4: 3 3  
## 5: 3 3

#### Identificando casos de missing na variavel v116 \*\*

O domicílio/família 145606, com informação faltante na variável v116 não poderá ser corrigido. Trata-se de uma residencia com um único morados. Logo, não é possível encontrar doadores.

censo1960\_pess\_semifinal[ , num\_NA\_V116\_dom := sum(is.na(V116\_dom)), by = cem\_IDfamilia]  
censo1960\_pess\_semifinal[ , num\_NA\_V116\_pess := sum(is.na(V116\_pess)), by = cem\_IDfamilia]  
censo1960\_pess\_semifinal[ num\_NA\_V116\_dom > 0 | num\_NA\_V116\_pess > 0 ]

## num\_linha\_pess UF\_pess V116\_pess REC\_TYPE V118\_pess V202 V203 V204 AGE  
## 1: 887256 71 NA 2 5 1 7 1 21  
## V205 V206 V207 V208 V209 V299 V210 V211 V212 V213 V214 V215 V216 V217  
## 1: 5 4 19 9 2 0 0 3 1 1 0 9 60 0  
## V218 V219 V220 V221 V223 V223B V224 cem\_IDfamilia  
## 1: 0 5 3 323 2 120 7 145606  
## cem\_vars\_still\_problematic\_pess cem\_diagnosis\_pess  
## 1: 1 2  
## cem\_problematic\_vars\_list\_pess place\_holder\_07\_08.x  
## 1: V116 07  
## place\_holder\_09\_16.x cem\_IDdomicilio V101 num\_linha\_dom UF\_dom V116\_dom  
## 1: 71038216 145606 1 887255 71 NA  
## V118\_dom V102 V103 V104 V105 V106 V107 V108 V109 V110 V111 V112 V113  
## 1: 5 5 9 8 2 7 9 6 8 0 2 2 1  
## cem\_vars\_still\_problematic\_dom cem\_diagnosis\_dom  
## 1: 1 2  
## cem\_problematic\_vars\_list\_dom place\_holder\_07\_08.y place\_holder\_09\_16.y  
## 1: V116 07 71038216  
## place\_holder\_32\_32 place\_holder\_35\_54 num\_NA\_UF\_dom num\_NA\_UF\_pess  
## 1: 0 0 0  
## num\_NA\_V116\_dom num\_NA\_V116\_pess  
## 1: 1 1

\*\* Identificando casos de missing na variavel v118\*\*

Não há necessidade de registros doadores para a variável V118:

censo1960\_pess\_semifinal[ , num\_NA\_V118\_dom := sum(is.na(V118\_dom)), by = cem\_IDfamilia]  
censo1960\_pess\_semifinal[ , num\_NA\_V118\_pess := sum(is.na(V118\_pess)), by = cem\_IDfamilia]  
censo1960\_pess\_semifinal[ num\_NA\_V118\_dom > 0 | num\_NA\_V118\_pess > 0 ]

## Empty data.table (0 rows) of 67 cols: num\_linha\_pess,UF\_pess,V116\_pess,REC\_TYPE,V118\_pess,V202...

## Inconsistências entre pessoas e domicílios

Há inconsistências incontornáveis entre o restante dos dados de domicílios e pessoas: informações discrepantes sobre UF de residência, município e local de moradia (rural/urbano). Assinalaremos esses registros com váriáveis adicionais, contruídas para esse fim:

# Transformado o objeto de volta em tibble  
censo1960\_pess\_semifinal = as\_tibble(censo1960\_pess\_semifinal) %>%  
 select(-starts\_with("num\_NA\_"))  
  
censo1960\_pess\_semifinal$cem\_dissonant\_UF = as.numeric(!(censo1960\_pess\_semifinal$UF\_pess == censo1960\_pess\_semifinal$UF\_dom))  
censo1960\_pess\_semifinal$cem\_dissonant\_V116 = as.numeric(!(censo1960\_pess\_semifinal$V116\_pess == censo1960\_pess\_semifinal$V116\_dom))  
censo1960\_pess\_semifinal$cem\_dissonant\_V118 = as.numeric(!(censo1960\_pess\_semifinal$V118\_pess == censo1960\_pess\_semifinal$V118\_dom))

Casos que permanecem problemáticos na variável UF (3 registros):

censo1960\_pess\_semifinal %>%  
 filter((cem\_dissonant\_UF %in% 1 | is.na(cem\_dissonant\_UF) )) %>%  
 nrow()

## [1] 3

Casos que permanecem problemáticos na variável v116 (44 registros):

censo1960\_pess\_semifinal %>%  
 filter( (cem\_dissonant\_V116 == 1 | is.na(cem\_dissonant\_V116)) ) %>%   
 nrow()

## [1] 44

Casos que permanecem problemáticos na variável v118 (280 registros):

censo1960\_pess\_semifinal %>%  
 filter((cem\_dissonant\_V118 == 1 | is.na(cem\_dissonant\_V118))) %>%  
 nrow()

## [1] 280

Total de registros problemáticos que permanecem no banco: 299.

censo1960\_pess\_semifinal %>%  
 filter((cem\_dissonant\_UF == 1 | is.na(cem\_dissonant\_UF) ) |  
 (cem\_dissonant\_V116 == 1 | is.na(cem\_dissonant\_V116))|  
 (cem\_dissonant\_V118 == 1 | is.na(cem\_dissonant\_V118))) %>%  
 nrow()

## [1] 299

## Re-criando o banco de domicílios (após as imputações) a partir dos dados de pessoas

No banco de pessoas, criamos algumas variáveis adicionais: um identificador de moradores (que exclui pessoas listadas como “não moradores presentes”) e o peso dos indivíduos (lançando mão das informações sobre a fração amostral). Essas informações serão usadas para produzir variáveis sobre domicílios: número de pessoas em cada domicílio e o próprio peso dos domicílios. Calculamos também o número de famílias por domicílio (lembrando que, por definição, o Censo de 1960 considera como domicílios coletivos aqueles com 4 ou mais famílias conviventes).

censo1960\_pess\_semifinal <- censo1960\_pess\_semifinal %>%  
 mutate(ind\_morador = as.numeric(V203 %in% c(0:3, 7:9)),  
 ind\_morador = ifelse(is.na(ind\_morador), 0, ind\_morador),  
 cem\_wgt = 1/0.0127) %>%  
   
 group\_by(cem\_IDdomicilio) %>%  
 mutate(cem\_num\_moradores = n(),  
 cem\_num\_familias = length(unique(cem\_IDfamilia)),   
 cem\_num\_familias = ifelse(V101 == 3, NA, cem\_num\_familias),   
 cem\_num\_pessoas = sum(ind\_morador),   
 ind\_morador = NULL) %>%  
 ungroup()

Procedemos então a agregação das informações das pessoas por IDs de domicílios. Com essa operação, re-criamos os registros de domicílios a partir da sumarização das informações individuais. Obviamente, apenas utilizamos as variáveis cabíveis, que não apresentam variações entre moradores de uma mesma residência – aquelas iniciadas por “V1”, UF e aquelas criadas acima, nos procedimentos anteriores.

Como procedimento para agregação dos casos, calcularemos o valor máximo de cada variável dentro do domicílio. Trata-se de uma operação arbitrária: como todos os indivíduos que co-habitam apresentam necessariamente valores idênticos, poderíamos ter calculado a média, o mínimo, a mediana ou qualquer outra medida. No entanto, todas essas se aplicam apenas aos dados numéricos. Aplicaremos um procedimento de sumarização para as variaveis de tipo character separadamente.

censo1960\_hh\_final <- censo1960\_pess\_semifinal %>%  
 select(UF\_dom, starts\_with("V1"),  
 ends\_with("\_dom"), -ends\_with("pess"),   
 cem\_IDdomicilio,  
 cem\_num\_pessoas, cem\_num\_moradores, cem\_num\_familias, cem\_wgt,  
 cem\_dissonant\_UF,cem\_dissonant\_V116, cem\_dissonant\_V118,   
 -cem\_vars\_still\_problematic\_dom, -num\_linha\_dom, -cem\_problematic\_vars\_list\_dom) %>%  
 group\_by(cem\_IDdomicilio) %>%  
 summarise\_all(max\_sem\_na) %>%  
 ungroup() %>%  
 mutate\_all(replace\_infinite) %>%  
 rename(UF = UF\_dom,  
 V116 = V116\_dom,   
 V118 = V118\_dom)

No passo a seguir, sumarizamos apenas a variável character faltante:

censo1960\_hh\_tmp <- censo1960\_pess\_semifinal %>%   
 select(cem\_IDdomicilio, cem\_problematic\_vars\_list\_dom) %>%  
 group\_by(cem\_IDdomicilio) %>%  
 summarise(cem\_problematic\_vars\_list\_dom = first(cem\_problematic\_vars\_list\_dom))

A partir dos IDs de domicílio, fundimos os dois bancos-sumário:

censo1960\_hh\_final <- censo1960\_hh\_final %>%  
 left\_join(x = .,  
 y = censo1960\_hh\_tmp,  
 by = "cem\_IDdomicilio")  
names(censo1960\_hh\_final) = tolower(names(censo1960\_hh\_final))

Por fim, re-ordenamos as variáveis e eliminamos da lista da variáveis problemáticas a coluna “REC\_TYPE” (que já não consta mais no banco de dados):

censo1960\_hh\_final <- censo1960\_hh\_final %>%  
 select(uf, v116, v118, cem\_iddomicilio,   
 v101, v102, cem\_wgt,  
 v103, v104, v105, v106, v107, v108,  
 v109, v110, v111, v112, v113,   
 cem\_num\_pessoas, cem\_num\_moradores,  
 cem\_num\_familias, cem\_diagnosis\_dom,  
 cem\_problematic\_vars\_list\_dom,  
 cem\_dissonant\_uf, cem\_dissonant\_v116,  
 cem\_dissonant\_v118)  
  
censo1960\_hh\_final <- censo1960\_hh\_final %>%  
 mutate(cem\_problematic\_vars\_list\_dom = str\_replace\_all(string = cem\_problematic\_vars\_list\_dom,  
 pattern = "REC\_TYPE",  
 replacement = "") %>%  
 str\_replace\_all(pattern = " ",  
 replacement = " ") %>%   
 tolower()  
 )

Obtemos então a **versão final do banco de domicílios** e a salvamos em disco:

data\_address <- "Data - After Consistency/Censo.1960.brasil.domicilios.amostra.1.27porcento.csv"  
  
fwrite(x = censo1960\_hh\_final,   
 file = data\_address,  
 na="",  
 row.names = F,  
 quote = TRUE)  
  
zip::zip(zipfile = 'Data - After Consistency/Censo.1960.brasil.domicilios.amostra.1.27porcento.zip',   
 files = data\_address)  
  
file.remove(data\_address)

## [1] TRUE

A lista de variáveis contidas no arquivo de dados, ao fim, é a seguinte:

names(censo1960\_hh\_final)

## [1] "uf" "v116"   
## [3] "v118" "cem\_iddomicilio"   
## [5] "v101" "v102"   
## [7] "cem\_wgt" "v103"   
## [9] "v104" "v105"   
## [11] "v106" "v107"   
## [13] "v108" "v109"   
## [15] "v110" "v111"   
## [17] "v112" "v113"   
## [19] "cem\_num\_pessoas" "cem\_num\_moradores"   
## [21] "cem\_num\_familias" "cem\_diagnosis\_dom"   
## [23] "cem\_problematic\_vars\_list\_dom" "cem\_dissonant\_uf"   
## [25] "cem\_dissonant\_v116" "cem\_dissonant\_v118"

## O banco de pessoas

O banco de pessoas, neste ponto, apenas requer alguns ajustes finais: renomear variáveis, excluir colunas auxiliares e alguma re-ordenação. Iniciamos por fazer com que os nomes de todas as variáveis sejam escritos apenas com letras minúsculas, como fizemos no banco de domicílios:

names(censo1960\_pess\_semifinal) = tolower(names(censo1960\_pess\_semifinal))

Mantemos então no banco de dados apenas as variáveis finais, excluindo aquelas utilizadas nos passos intermediários da consistência. Renomeamos também a variável idade, anteriormente denominada “age”, para v204b. No dicionário de códigos o nome v204 está reservado para a variável “Tipo de Idade” (se em meses ou em anos). O nome “age” era o que constava nas sintaxes recebidas juntamente com o arquivo de dados original.A decisão de denominá-la v204b tem como objetivo manter o padrão de nomeação usual do IBGE e seguir a mesma lógica da variável referente aos setores de atividade econômica, denominada v223b, que também não possuia código próprio (no dicionário, o código v223 é usado para descrever a variável que capta a ocupação na última semana, como o setor de atividade).

censo1960\_pess\_final <- censo1960\_pess\_semifinal %>%  
 rename(cem\_idindividuo = num\_linha\_pess,  
 v204b = age) %>%   
 select(uf\_pess, uf\_dom,   
 v116\_pess, v116\_dom,   
 v118\_pess, v118\_dom,  
 starts\_with("v"),  
 starts\_with("cem\_id"),  
 starts\_with("c"),  
 everything(),  
 -rec\_type,   
 -starts\_with("place"),  
 -num\_linha\_dom,  
 -cem\_vars\_still\_problematic\_dom, -cem\_vars\_still\_problematic\_pess) %>%   
 select(starts\_with("u"), starts\_with("v"), everything())

Reordenamos então as variáveis do banco de dados:

censo1960\_pess\_final <- censo1960\_pess\_final %>%  
 select(uf\_pess,uf\_dom,v116\_pess,v116\_dom,v118\_pess,v118\_dom,  
 cem\_idindividuo,cem\_idfamilia,cem\_iddomicilio,cem\_wgt,  
 v202,v203,v204,v204b,v205,v206,v207,v208,v209,v299,  
 v210,v211,v212,v213,v214,v215,v216,v217,v218,v219,  
 v220,v221,v223,v223b,v224,v101,v102,v103,v104,v105,  
 v106,v107,v108,v109,v110,v111,v112,v113,  
 cem\_num\_pessoas,cem\_num\_moradores,cem\_num\_familias,  
 cem\_diagnosis\_pess,cem\_problematic\_vars\_list\_pess,cem\_diagnosis\_dom,  
 cem\_problematic\_vars\_list\_dom,cem\_dissonant\_uf,cem\_dissonant\_v116,  
 cem\_dissonant\_v118)

Por fim, substituímos o nome “age” por “v204b”, onde houver, na coluna que indica as variáveis problemáticas, dos registros com informações corrompidas.

censo1960\_pess\_final <- censo1960\_pess\_final %>%   
 mutate(cem\_problematic\_vars\_list\_pess = str\_replace\_all(string = cem\_problematic\_vars\_list\_pess,  
 pattern = "AGE",  
 replacement = "v204b") %>%  
 str\_replace\_all(pattern = "REC\_TYPE",  
 replacement = "") %>%  
 str\_replace\_all(pattern = " ",  
 replacement = " ") %>%   
 tolower(),  
 cem\_problematic\_vars\_list\_dom = str\_replace\_all(string = cem\_problematic\_vars\_list\_dom,  
 pattern = "REC\_TYPE",  
 replacement = "") %>%  
 str\_replace\_all(pattern = " ",  
 replacement = " ") %>%   
 tolower()  
 )

Salvamos em disco o **arquivo final de pessoas do Censo de 1960**:

data\_address <- "Data - After Consistency/Censo.1960.brasil.pessoas.amostra.1.27porcento.csv"  
  
fwrite(x = censo1960\_pess\_final,  
 file = data\_address,  
 na = "",  
 row.names = F,  
 quote = TRUE)  
  
zip::zip(zipfile = 'Data - After Consistency/Censo.1960.brasil.pessoas.amostra.1.27porcento.zip',   
 files = data\_address)  
  
file.remove(data\_address)

## [1] TRUE

A lista de variáveis contidas no arquivo de dados, ao fim, é:

names(censo1960\_pess\_final)

## [1] "uf\_pess" "uf\_dom"   
## [3] "v116\_pess" "v116\_dom"   
## [5] "v118\_pess" "v118\_dom"   
## [7] "cem\_idindividuo" "cem\_idfamilia"   
## [9] "cem\_iddomicilio" "cem\_wgt"   
## [11] "v202" "v203"   
## [13] "v204" "v204b"   
## [15] "v205" "v206"   
## [17] "v207" "v208"   
## [19] "v209" "v299"   
## [21] "v210" "v211"   
## [23] "v212" "v213"   
## [25] "v214" "v215"   
## [27] "v216" "v217"   
## [29] "v218" "v219"   
## [31] "v220" "v221"   
## [33] "v223" "v223b"   
## [35] "v224" "v101"   
## [37] "v102" "v103"   
## [39] "v104" "v105"   
## [41] "v106" "v107"   
## [43] "v108" "v109"   
## [45] "v110" "v111"   
## [47] "v112" "v113"   
## [49] "cem\_num\_pessoas" "cem\_num\_moradores"   
## [51] "cem\_num\_familias" "cem\_diagnosis\_pess"   
## [53] "cem\_problematic\_vars\_list\_pess" "cem\_diagnosis\_dom"   
## [55] "cem\_problematic\_vars\_list\_dom" "cem\_dissonant\_uf"   
## [57] "cem\_dissonant\_v116" "cem\_dissonant\_v118"