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

Rogério Jerônimo Barbosa

19 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:

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

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

## $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

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()

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()

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 = "linhas\_problematicas\_pessoas.csv",  
 row.names = F)

#### 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")

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

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())

Identifica 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(.)

Identificando as variaveis problematicas nos casos problematicos

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")

Substituindo valores invalidos por missing

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   
 }   
}

Compilando o banco de pessoas (quase) final

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()