

Haydée Svab

**Evolução dos padrões de deslocamento por gênero: um
estudo da Região Metropolitana de São Paulo**

São Paulo
Fevereiro/2016

Haydée Svab

Evolução dos padrões de deslocamento por gênero: um estudo da Região Metropolitana de São Paulo

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para Exame
de Qualificação ao Mestrado.

Área de Concentração:
Planejamento de Transportes

Orientador: Orlando Strambi

São Paulo
Fevereiro/2016

Haydée Svab

Evolução dos padrões de deslocamento por gênero: um estudo da Região Metropolitana de São Paulo/ Haydée Svab. – São Paulo, Fevereiro/2016-
426 p. : il. (algumas color.) ; 30 cm.

Orientador: Orlando Strambi

Dissertação de Mestrado – Universidade de São Paulo – USP
Escola Politécnica, Fevereiro/2016.

1. Planejamento de Transportes. 2. Análise de Demanda. 3. Gênero. I.
Orientador: Orlando Strambi II. Universidade de São Paulo. III. Escola Politécnica.
IV. Título

CDU XX:XXX:XXX.X

Haydée Svab

Evolução dos padrões de deslocamento por gênero: um estudo da Região Metropolitana de São Paulo

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para Exame de Qualificação ao Mestrado.

Trabalho aprovado.
São Paulo, 18 de dezembro de 2014:

Orlando Strambi
Orientador:

Professor
Profa. Dra. Heloísa Buarque de Almeida

Professor
Prof. Dr. Luiz Paulo Lopes Fávero

São Paulo
Fevereiro/2016

Este trabalho é dedicado a todas as pessoas que acreditam que o planejamento de transportes só será bem-sucedido se houver conexão com as dimensões sociais e urbanas que influencia e pelas quais é influenciado.

Agradecimentos

Um trabalho de pesquisa sempre é fruto direta ou indiretamente da colaboração de muitas pessoas. Agradecê-las nominalmente é um risco por ser possível deixar inadvertidamente alguém de fora. Assim, deixo registrado um sincero agradecimento a todas e todos que contribuíram para monografia tornar-se realidade: minha família de nascença e também aquela que escolhemos e vamos compondo ao longo da vida.

Ao Diego Rabatone Oliveira, agradeço pela parceria incondicional, pela tranquilidade e amor fundamentais nos momentos de tensão e pelas horas a fio dispendidas ombro a ombro sobre a mesa de trabalho.

Ao Orlando Strambi, agradeço pela dedicação e orientação, indispensáveis para o andamento deste trabalho.

À Cia. do Metropolitano de São Paulo, na pessoa de Emilia Mayumi Hiroi, pela paciência, prestatividade e abertura de portas (e dados), que materializou o conceito de informação pública e dado aberto não apenas para mim, mas para a sociedade.

Ao Felipe Dias e à Glauzia Pereira sou grata pela amizade, pela disposição em ajudar em momentos cruciais - sem vocês a realização deste trabalho seria impossível.

À Bianca Alves, ao Fábio Lofrano e à Patricia Cornils pelas conversas encorajadoras sobre o tema e pela admiração intelectual que me provocam.

Ao Giuliano Salcas Olguin, um dia chefe, hoje, amigo e pai, que guiou minhas primeiras frases produzidas dentro que hoje se chama formalmente de pesquisa.

Às professoras Maria Eugenia Boscov, Cintia Borges Margi e Liedi Bernucci, pelo apoio e pelo exemplo que são para mim como profissionais e como mulheres.

À Angela Buscema, à Enaege Santana e à Patrícia Santana por me auxiliarem a encontrar meus caminhos, não só burocráticos, dentro da Escola Politécnica da USP.

Ao Guilherme Carmelo Ungar, agradeço pela amizade, bom humor e espirituosidade cotidianas que desmancham qualquer desânimo.

Finalmente, agradeço a Villa-Lobos e *The Piano Guys* que acalmaram minhas inquietações com suas composições.

“No atlas do seu império, ó Grande Khan, devem constar tanto a grande Fedora de pedra quanto as pequenas Fedoras das esferas de vidro. Não porque sejam igualmente reais, mas porque são todas supostas. Uma reúne o que é considerado necessário, mas ainda não o é; as outras, o que se imagina possível e um minuto mais tarde deixa de sé-lo.”

(Calvino, Ítalo; In: Cidades Invisíveis)

“Sustainable development is not a goal, but a change in direction.”

(Banister, David; In: Unsustainable Transport)

Resumo

O presente trabalho aborda questões relativas a análises de comportamento da demanda por transportes e joga luz sobre as questões de gênero, tendo como perspectiva o caminho do desenvolvimento sustentável. A revisão de literatura explora os conceitos de gênero, mobilidade, acessibilidade e sustentabilidade, bem como busca apresentar resultados de estudos que analisaram o que ocorre nas intersecções destes conceitos. Para investigar a sobreposição dessas áreas na RMSP adotam-se os dados secundários advindos das Pesquisas Origem Destino da Cia. do Metropolitano de São Paulo. Toma-se por hipótese que existem diferenças entre os padrões feminino e masculino de atividades e mobilidade, sendo que o problema de pesquisa que se deseja investigar é o porquê dessas diferenças existirem. Por isso, usam-se dados de 1977, 1987, 1997 e 2007 buscando observar possíveis tendências ao longo do tempo. Desta maneira, foi primeiro necessário compreender criticamente os dados para, na sequência, proceder algumas análises que confirmam alguns fenômenos ligados às características individuais (como envelhecimento da população, por exemplo) e outros ligados às características das viagens (como aumento da mobilidade feminina). Na próxima etapa pretende-se definir o(s) método(s) a ser(em) adotado(s) que melhor descreva(m) e explique(m) os comportamentos da demanda de transportes, considerando o gênero como uma variável explicativa.

Palavras-chaves: gênero; mobilidade; planejamento; sustentabilidade; transportes.

Lista de ilustrações

Figura 1 – Mapa dos municípios que compõem a região metropolitana em 2014, divididos por sub-regiões	29
Figura 2 – Prisma Espaço Tempo	43
Figura 3 – Quantidade de espaço viário requerido para transportar 60 pessoas por ônibus, bicicleta e carro.	44
Figura 4 – Frances Willard aprendendo a andar de bicicleta	46
Figura 5 – Simulação de probabilidade de uso do carro em função do número de crianças na família, na Alemanha	50
Figura 6 – Esquema indicativo da formação dos IDs	71
Figura 7 – Box plot da variável “TOT_FAM”, por ano	94
Figura 8 – Box plot da variável “TOT_PESS”, por ano	103
Figura 9 – Etapas para a realização de análise de conglomerados (<i>clusters</i>)	129
Figura 10 – Dendrograma resultante da análise de conglomerados hierárquico, para o conjunto de atributos de viagens relativas às famílias	133
Figura 11 – Dendrograma resultante da análise de conglomerados hierárquico, para o conjunto de atributos de viagens relativas às pessoas	134

Lista de quadros

Quadro 1 – Critérios para Avaliação de Dados Secundários	58
Quadro 2 – Dados para atribuição de renda familiar	64
Quadro 3 – Layout	73

Lista de gráficos

Gráfico 1 – Percentual de indivíduos que fazem parte da PEA, por sexo, no Brasil, entre 1950 e 2010	37
Gráfico 2 – Jornadas Médias para o Mercado de Trabalho e para Reprodução Social, por sexo, raça/cor e região geográfica, no Brasil, em 2003	38
Gráfico 3 – Distribuição da variável “FE_DOM”, por ano	92
Gráfico 4 – Distribuição da variável “REN_FAM”, por ano	98
Gráfico 5 – Distribuição da variável “FAIXA_REN_FAM”, por ano	99
Gráfico 6 – Proporção de famílias com pessoa responsável do sexo feminino e do sexo masculino, segundo posse de automóveis, por ano	101
Gráfico 7 – Proporção de famílias com presença de dependentes, por ano	104
Gráfico 8 – Distribuição da variável “IDADE” de respondentes, por ano e por sexo	105
Gráfico 9 – Distribuição da variável “SIT_FAM”, por ano e por sexo	107
Gráfico 10 – Distribuição da variável “GRAU_INSTR”, por ano e por sexo	109
Gráfico 11 – Distribuição da variável “FAIXA_REN_IND”, por ano	111
Gráfico 12 – Distribuição da variável “TOT_VIAG” por ano e por sexo	115
Gráfico 13 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o primeiro modo da viagem, por ano	119
Gráfico 14 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o segundo modo da viagem, por ano	119
Gráfico 15 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o modo da viagem (agregado), por ano	121
Gráfico 16 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o motivo de destino, por ano	123
Gráfico 17 – Comparação entre as durações médias de viagem por ano e por sexo, segundo os modos (agregados)	125
Gráfico 18 – Comparação entre as durações médias de viagem por ano e por sexo, segundo os modos coletivos	126
Gráfico 19 – Comparação entre as durações médias de viagem por ano e por sexo, segundo o motivo da viagem	128
Gráfico 20 – Avaliação do número de <i>clusters</i> para o conjunto de atributos de viagens relativas às famílias	134
Gráfico 21 – Avaliação do número de <i>clusters</i> para o conjunto de atributos de viagens relativas às pessoas	135

Lista de tabelas

Tabela 1 – Consumo dinâmico de espaço por modo e renda na RMSP	45
Tabela 2 – Características Amostrais das Pesquisas OD	62
Tabela 3 – Características Gerais das Pesquisas OD	63
Tabela 4 – Quantidade de viagens e de domicílios, por ano, antes e depois da preparação do BDU	81
Tabela 5 – Estatísticas da variável “ANO”	89
Tabela 6 – Estatísticas da variável “CD_ENTRE”	89
Tabela 7 – Estatísticas da variável “DIA_SEM”	90
Tabela 8 – Estatísticas da variável “F_DOM”	91
Tabela 9 – Estatísticas da variável “F_FAM”	91
Tabela 10 – Estatísticas da variável “F_PESS”	91
Tabela 11 – Estatísticas da variável “FE_DOM”	92
Tabela 12 – Estatísticas da variável “TIPO_DOM”	93
Tabela 13 – Estatísticas da variável “TOT_FAM”	93
Tabela 14 – Estatísticas da variável “COND_MORA”	94
Tabela 15 – Deflatores utilizados para correção dos valores monetários para outubro/2007	96
Tabela 16 – Estatísticas da variável “REN_FAM”	97
Tabela 17 – Evolução da frota de automóveis na RMSP segundo as Pesquisas OD .	97
Tabela 18 – Venda interna de veículos no Brasil entre 1960 e 2009	100
Tabela 19 – Estatísticas da variável “QT_AUTO”	100
Tabela 20 – Proporção das famílias segundo posse de automóveis	101
Tabela 21 – Estatísticas das variáveis “QT_MOTO” e “QT_BICI”	102
Tabela 22 – Estatísticas da variável “TOT_PESS”	102
Tabela 23 – Frequências absoluta e relativa da variável SEXO, por ano	103
Tabela 24 – Evolução das taxas de fecundidade no Brasil, de 1970 a 2010	106
Tabela 25 – Frequências da variável TRABALHA, por ano e sexo	108
Tabela 26 – Frequências da variável ESTUDA, por ano e sexo	108
Tabela 27 – Estatísticas da variável “REN_IND”	110
Tabela 28 – Estatísticas da variável “TOT_VIAG”, por ano e por sexo	113
Tabela 29 – Estatísticas da variável “TOT_VIAG”, por ano e por sexo, considerando apenas quem fez viagem	114
Tabela 30 – Estatísticas da variável “FAM_VIAG_TOT”, por ano	116
Tabela 31 – Frequência relativa das variáveis “MODO1” e “MODO2”, por ano . . .	117
Tabela 32 – Frequência relativa da variável “TIPO_VIAG”, por ano	120
Tabela 33 – Frequência relativa da variável “MOTIVO_DEST”, por ano	122

Tabela 34 – Estatísticas da variável “DURACAO”, por ano	124
Tabela 35 – Medidas de dissimilaridade utilizadas em análise de <i>cluster</i>	132
Tabela 36 – Resultado do agrupamento de 4 <i>clusters</i> , por atributos de viagens de família - método Ward	135
Tabela 37 – Resultado do agrupamento de 4 <i>clusters</i> , por atributos de viagens de família - método centroide	136
Tabela 38 – Resultado do agrupamento de 3 <i>clusters</i> , por atributos de viagens de família - método Ward	136
Tabela 39 – Resultado do agrupamento de 3 <i>clusters</i> , por atributos de viagens de família - método centroide	136
Tabela 40 – Resultado do agrupamento de 3 <i>clusters</i> , por atributos de viagens de pessoas - métodos Ward e centroide	137
Tabela 41 – Ordenamento das variáveis pelas maiores diferenças percentuais - para agrupamento FAM_CLUSTER_WARD4	138
Tabela 42 – Ordenamento das variáveis pelas maiores diferenças percentuais - para agrupamento FAM_CLUSTER_CENTROIDE4	139
Tabela 43 – Resultado da regressão logística multinomial com categoria de referência Ano=4 (correspondente a 2007)	141

Lista de abreviaturas e siglas

CO ₂	dióxido de carbono
Metrô-SP	Companhia do Metropolitano de São Paulo
OD-1977	Pesquisa Origem Destino do Metrô-SP de 1977
OD-1987	Pesquisa Origem Destino do Metrô-SP de 1987
OD-1997	Pesquisa Origem Destino do Metrô-SP de 1997
OD-2007	Pesquisa Origem Destino do Metrô-SP de 2007
ONU	Organização das Nações Unidas
PIB	produto interno bruto
RMSp	Região Metropolitana de São Paulo
WCED	Comissão Mundial sobre Meio Ambiente e Desenvolvimento

Sumário

1	INTRODUÇÃO	25
1.1	Objeto e Objetivos	26
1.2	Justificativa	26
1.3	Método	27
1.4	Estrutura do trabalho	27
2	REVISÃO DE LITERATURA	29
2.1	Gênero	30
2.2	Mobilidade Urbana	39
2.3	Intersecções e Sobreposições	46
3	CONSIDERAÇÕES METODOLÓGICAS	57
4	PESQUISA ORIGEM DESTINO (OD)	61
4.1	Periodicidade e Objetivos da Pesquisa OD	61
4.2	Descrição Sucinta	61
4.3	Dados Coletados	62
4.4	Conceitos Adotados	65
4.5	Limitações	66
5	TRATAMENTO DOS DADOS	67
5.1	Preparação da base de dados	67
5.2	Critérios de Validação	80
5.3	Formulação de Novas Variáveis	81
5.3.1	Variáveis relativas às viagens	81
5.3.2	Variáveis relativas às pessoas	83
5.3.3	Variáveis relativas às famílias	86
6	ANÁLISES REALIZADAS	89
6.1	Estatísticas Descritivas	89
6.2	Análises Preliminares	112
6.3	Análise de conglomerados para identificação de grupos	129
6.4	Regressão logística para investigar a formação de grupos	140
	Referências	143

ANEXOS

155

ANEXO A – CORRESPONDÊNCIA ENTRE ZONAS DAS PESQUISAS ORIGEM DESTINO POR MEIO DAS UNIDADES DE CORRESPONDÊNCIA ENTRE ZONAS (UCOD)	157
ANEXO B – MAPAS DE ZONAS E SUBZONAS DAS PESQUISAS ORIGEM DESTINO (1977, 1987, 1997 E 2007)	161
ANEXO C – LAYOUTS DOS BANCOS DE DADOS DAS PESQUISAS ORIGEM-DESTINO DO METRÔ-SP (1977, 1987, 1997 E 2007)	163
ANEXO D – ROTINAS DE PREPARAÇÃO DAS BASES DE DADOS DAS PESQUISAS ORIGEM-DESTINO DO METRÔ-SP (1977, 1987, 1997 E 2007)	181
ANEXO E – SÍNTESE DE RESULTADOS PARA 4 CONGLOMERADOS	425

1 Introdução

A propagação de megacidades¹, especialmente nos países em desenvolvimento, tem sido acompanhada de fenômenos como aumento da urbanização e da motorização. O Brasil já conta com duas megacidades: Rio de Janeiro e São Paulo, esta última é objeto deste estudo. O que pouco tem acompanhado esses crescimentos têm sido as políticas de planejamento urbano e de transportes, de forma integrada, o que tem incorrido em problemas como congestionamentos (KINGHAM; DICKINSON; COPSEY, 2001; STEG, 2005; METZ, 2012), piora das características ambientais (TERTOOLEN; KREVELD; VERSTRATEN, 1998; RICHARDSON, 2005; BANISTER, 2011), e aprofundamento das desigualdades (HODGE, 1995; AHMED; LU; YE, 2008; LEWIS, 2011). A inequidade não se dá apenas no acesso aos recursos e às oportunidades, mas também na distribuição dos espaços públicos (ALVA, 1997) e, em específico, o espaço destinado à circulação nas cidades (VASCONCELLOS, 2012).

Estudos constatam (VASCONCELLOS, 2001; RUEDA et al., 2007) que os automóveis particulares ocupam maior quantidade de espaço de circulação para transportar a mesma quantidade de pessoas do que outros modos de transporte (motorizados ou não). Tendo em vista esse cenário, recentemente, foi aprovada no Brasil a Política Nacional de Mobilidade Urbana (BRASIL, 2012) que implicitamente indica nos seus “princípios, diretrizes e objetivos” que não é mais aceitável a priorização do automóvel particular no meio urbano, já que deve promover a “equidade no uso do espaço público de circulação, vias e logradouros”.

Neste cenário, justifica-se estudar alternativas que contribuam para uma mobilidade mais sustentável nos grandes centros urbanos, considerando que entre os vários modos de transporte, em geral, o automóvel particular é a forma de maior atratividade. Observa-se, entretanto, que apesar de seus apelos, um grupo particular de pessoas chama a atenção em relação ao uso diferenciado que fazem do automóvel: as mulheres. Historicamente, estas usam menos o automóvel em relação aos homens (FOX, 1983; HJORTHOL, 2000; POLK, 2003; BEST; LANZENDORF, 2005).

Esta dissertação, portanto, aborda diretamente questões relativas a análises de comportamento da demanda por transportes, jogando luz sobre as questões de gênero e suas articulações, principalmente com políticas de transporte e, em menor medida, com

¹ “Megacidade” é um termo, cunhado em 1990 pela ONU, para designar cidades com mais de dez milhões de habitantes. Segundo dados da Divisão de População da ONU, em 2014 existem 33 megacidades no planeta e São Paulo é a sétima megacidade no ranking. Fonte: <<http://esa.un.org/unpd/ppp/>> Acesso em 30 de outubro de 2014. Segundo Freitag (2007), São Paulo é também uma “megalópole”, isto é, uma megacidade (município com mais de 10 milhões de habitantes) que sofreu um crescimento muito acelerado nas três ou quatro últimas décadas do século XX.

políticas de planejamento urbano. Entender melhor o comportamento da demanda pode colaborar na formulação de políticas de incentivo à troca de modos de transporte (de menos para mais sustentáveis) e à redução da necessidade de viajar.

1.1 Objeto e Objetivos

O objeto desta dissertação de mestrado é a relação entre o gênero e os deslocamentos de indivíduos. Assim, por objetivo geral tem-se investigar como o padrão de viagens de indivíduos é afetado pela categoria de análise gênero, no período de 1977 a 2007 na Região Metropolitana de São Paulo (RMSP). Toma-se por hipótese a ser explorada que pessoas com identidades de gênero masculina e feminina tem se deslocado de forma diferente no espaço.

Como objetivos específicos estão: (i) analisar padrões de viagens, por gênero, de cada *cross-section*²; (ii) analisar padrões de viagens, por modo, de cada *cross-section*; (iii) comparar os padrões encontrados e verificar se existe alguma tendência ao longo do tempo; (iv) elaborar hipóteses sobre as motivações das mulheres e dos homens para realizar viagens, com base na teoria subjacente; (v) analisar criticamente se hipóteses elencadas são validadas pelos dados e resultados obtidos.

1.2 Justificativa

A questão de gênero nos transportes tem atraído atenção crescente da comunidade científica (ROSENBLUM, 1978; HANSON; JOHNSTON, 1985; ROSENBLUM, 2006; CRESSWELL; UTENG, 2008; HANSON, 2010). Pesquisadores(as) começaram a examinar os padrões de mobilidade com o recorte de gênero considerando que há acesso desigual a recursos materiais e imateriais (HOWE; O'CONNOR, 1982; HANSON, 1995a; SILVEY; ELMHIRST, 2003; RAJU, 2005) que levam a diferenças nos padrões de atividades e de viagem (FAGNANI, 1983; MCLAFFERTY; PRESTON, 1991; MCLAFFERTY; PRESTON, 1992; JOHNSTON-ANUMONWO, 1992; ROOT; SCHINTLER, 1999; SCHWANEN; DIJST; DILEMAN, 2002; LEE; MCDONALD, 2003; MCGUCKIN; ZMUD; NAKAMOTO, 2005; CRANE, 2007; VASCONCELLOS, 2012), em particular na escolha de modo e no uso do automóvel particular (FOX, 1983; HJORTHOL, 2000; POLK, 2003; BEST; LANZENDORF, 2005).

Com uma maior participação de mulheres no mercado de trabalho a diferença entre os rendimentos de homens e mulheres vem (lentamente) diminuindo, o que levaria a crer que o padrão de viagens das mulheres passasse a se assemelhar aos dos homens, pois teriam

² Dados em *cross-section* são dados em seções transversais, ou seja, revelam características por meio das variáveis para um dado momento.

mais recursos financeiros para dispor de um automóvel. Pela flexibilidade de horário e de trajeto que o carro pode oferecer, seria esperado um aumento no uso do carro pelas mulheres devido à pressão exercida por uma dupla jornada (trabalho formal e doméstico) ainda mais se houver presença de criança na família. Entretanto, segundo estudo de Best e Lanzendorf (2005) na Alemanha, a presença de criança na família gera o seguinte efeito: a maternidade reduz a probabilidade de uso do carro por mulheres enquanto a paternidade a aumenta para homens.

Resultados como esse, aparentemente contraintuitivo, demonstram a necessidade de olhar mais atento e que considere complexidades sociais na análise do comportamento das demandas de transportes. Ademais, justifica a necessidade de compreender as estratégias adotadas pelas mulheres para cumprir com seu amplo conjunto de compromissos sem, muitas vezes e por motivos diversos, contar com o acesso ao automóvel, ou com acesso restrito a este modo de transporte. Embora este tipo de análise seja cada vez mais frequente no cenário internacional, ainda é incipiente no Brasil. Assim, as lições extraídas deste estudo podem identificar oportunidades de formulação de políticas públicas que visem reduzir a dependência do uso automóvel particular, promovendo sistemas de transporte e estimulando comportamentos mais sustentáveis nas megacidades brasileiras.

1.3 Método

Em primeiro lugar, será conduzida uma revisão bibliográfica sobre gênero, sustentabilidade, transportes e suas intersecções conceituais, tendo como foco o estabelecimento de padrões de viagens e as escolhas modais. A segunda etapa consistirá da caracterização da evolução do padrão de atividades e mobilidade das mulheres por meio de análise das Pesquisas Origem e Destino de 1977, 1987, 1997 e 2007 para a Região Metropolitana de São Paulo (RMSP). A última etapa prevê uma pesquisa qualitativa com o propósito de buscar compreender as motivações comportamentais e estratégias de deslocamento de mulheres e homens na consecução de seu padrão de atividades.

1.4 Estrutura do trabalho

No primeiro capítulo é feita uma breve introdução ao tema, assim como são apresentados objetivos e justificativa deste trabalho. No segundo capítulo é apresentada a revisão de literatura no que tange aos principais conceitos que são explorados, como mobilidade, gênero e sustentabilidade. Na sequência, são apresentadas algumas reflexões acerca das intersecções e sobreposições desses aspectos. No terceiro capítulo são apresentados os métodos que pretendem ser usados no desenvolvimento desta dissertação, a saber: análise das pesquisas Origem Destino realizadas pela Companhia do Metropolitano de São Paulo (Metrô-SP) por meio de um estudo de coorte em pseudo-painel. No quarto capítulo

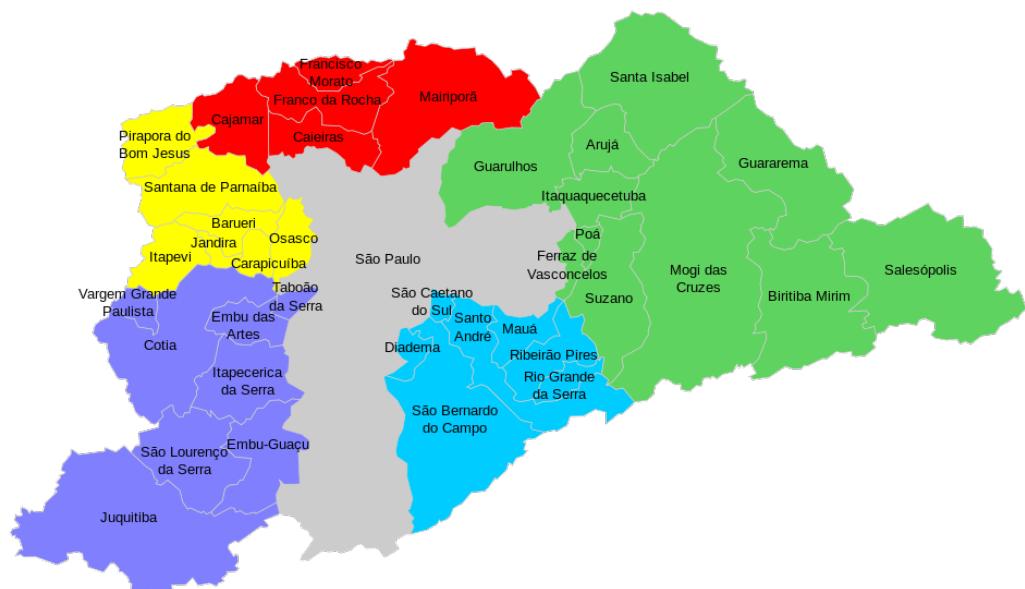
são apresentadas algumas análises feitas a partir da pesquisa Origem Destino 2007 do Metrô-SP (OD-2007), à luz dos caminhos apontados pela revisão de literatura. Ao final, são apresentadas algumas conclusões preliminares e também quais são os próximos passos planejados para a continuação deste trabalho.

2 Revisão de Literatura

“Todo enunciado - desde a breve réplica (monolexemática) até o romance ou o tratado científico - comporta um começo absoluto e um fim absoluto: antes de seu início, há os enunciados dos outros, depois de seu fim, há os enunciados-respostas dos outros [...]. O locutor termina seu enunciado para passar a palavra ao outro ou para dar lugar à compreensão responsiva ativa do outro. O enunciado não é uma unidade convencional, mas uma unidade real, estritamente delimitada pela alternância dos sujeitos falantes” (Bakhtin)

Este capítulo tem por objetivo clarificar conceitos considerando a evolução e as intersecções entre as concepções utilizadas, bem como dar um panorama geral de como as questões de gênero, de mobilidade e de sustentabilidade vêm sendo tratadas sob a perspectiva do planejamento de transportes. Buscou-se, sempre que possível, apresentar aspectos ligados à realidade brasileira e quiçá paulistana, pois o escopo espacial de análise do presente trabalho é a Região Metropolitana de São Paulo (RMSP) (ver Figura 1), área coberta pela Pesquisas Origem e Destino (Pesquisas OD) do Metrô-SP.

Figura 1 – Mapa dos municípios que compõem a região metropolitana em 2014, divididos por sub-regiões



Fonte: Mapa elaborado por Marcos Elias Oliveira Júnior, segundo a Lei 1.139/2011 ([SÃO PAULO \(ESTADO\), 2011](#)). Disponível em: <http://pt.wikipedia.org/wiki/Regi%C3%A3o_Metropolitana_de_S%C3%A3o_Paulo#mediaviewer/File:Mapa-RMSP-subregions.svg> Acesso em 10 de novembro de 2014

2.1 Gênero

Ao nascer, umas das primeiras atividades do ser humano é comunicar-se, o que inclui nominar para si o mundo que o cerca. Esse processo não se dá de maneira solitária, a nominação advém de uma interação social que visa compartilhar signos afim de efetivar a comunicação. Por isso, este capítulo visa estabelecer o que se deseja exprimir através de palavras-chave deste trabalho (gênero, mobilidade e sustentabilidade), considerando a visão de Bakthin, cujo trabalho, segundo [Stella \(2005\)](#), indicava ser necessário

não somente a palavra, mas também a linguagem em geral, ser concebida e tratada de uma outra forma, levando-se em conta sua história, sua historicidade, ou seja, especialmente a linguagem em uso. Isso significa que, no pensamento bakthiniano, a palavra reposiciona-se em relação às concepções tradicionais, passando a ser encarada como um elemento concreto de feitura lógica.

Há um senso comum que confunde e funde, não por acaso, os conceitos de sexo e gênero, muito embora sejam distintos - distinção esta encontrada em maior ou menor grau de acordo com o idioma. Em inglês a palavra *sex* tem sentido mais limitado, ligado à anatomia, e a palavra *gender* tem sentido mais amplo, ligado à construção cultural da identidade. Em francês, a palavra *séxe* e, em alemão, a palavra *Geschlecht*, designam tanto diferenças físicas como psicológicas, sociais e culturais ([FRAISSE, 2001](#)). [Moraes \(1998\)](#) reporta que, em francês, frequentemente utiliza-se *rapports sociaux de séxe* ao invés de *gendre* para se designar *gênero*. Comparado com o termo inglês *gender*, “a palavra gênero, em português, é um substantivo masculino que designa uma classe que se divide em outras, que são chamadas espécies”, definição então retirada do Novo Dicionário Aurélio por [Moraes \(1998, p.101\)](#). Já hoje, em 2014, o Dicionário [Aurélio \(2014\)](#) já comporta entre suas definições, aquela em que *gênero* pode ser entendido como o “conjunto de propriedades atribuídas social e culturalmente em relação ao sexo dos indivíduos”. Porém, entre as definições mais gerais apresentadas pelo Dicionário ?? (??), *gênero* é definido da seguinte forma:

s.m. (lat **generu*, por *genus*) 1 Grupo de seres que têm iguais caracteres essenciais. 2 *Lóg.* A classe que tem mais extensão e portanto menor compreensão que a espécie. 3 *Biol.* Grupo morfológico intermediário entre a família e a espécie. 4 *Gram.* Flexão pela qual se exprime o sexo real ou imaginário dos seres. 5 *Gram.* Forma do adjetivo ou pronome com relação ao gênero dos nomes a que se refere. 6 Agrupamento de indivíduos que possuem caracteres comuns. 7 Espécie, casta, raça, variedade, sorte, categoria, estilo etc. 8 Qualidade, espécie, modo.

Percebe-se que o termo gênero designa um conceito em construção e consolidação, não apenas no Brasil, sendo necessário defini-lo sempre que o utilizarmos como denominação de categoria de análise ([MORAES, 1998](#)). Para isso, será feito um breve apanhado do

surgimento e trajetória da palavra *gender* ou *gênero* nas pesquisas acadêmicas, inclusive no Brasil, bem como a evolução do conceito.

O conceito fundido e identitário de sexo e gênero, como se fossem sinônimos, pertence a uma visão binária de mundo que define as mulheres mais próximas das natureza, do trabalho reprodutivo, da passividade e do irracional e, em oposição, define os homens mas próximos à cultura, ao trabalho produtivo, à ação e à racionalidade (HARAWAY, 2004). Estudiosas feministas, rejeitando o determinismo bio-sexual para a situação social das mulheres, precisavam desmontar a naturalização das diferenças entre homens e mulheres que vinculava suas relações sociais, políticas e econômicas a seu aparelho reprodutor (PISCITELLI, 2009). Para Haraway (2004, p.218), as feministas lutaram “para remover as mulheres da categoria da natureza e colocá-las na cultura como sujeitos sociais na história, construídas e auto-construtoras”. Dessa forma, a evolução do conceito de gênero mescla-se à história do feminismo.

A chamada “primeira onda feminista” ocorreu entre o final do século XIX e o início do século XX nos países hoje considerados desenvolvidos da Europa e da América do Norte. A principal bandeira reivindicava direitos iguais, compondo uma ideia de que deveria haver uma igualdade entre os sexos. Em decorrência dessa primeira movimentação, em diversos países, as mulheres conquistaram alguns direitos equivalentes aos dos homens, como o voto. Essa conquista do voto como um direito político caracterizou o movimento sufragista, que não pode ser confundido com o movimento feminista, embora seja parte dele. A Finlândia foi o primeiro país a garantir direito a votar e ser votado(a) igualmente a mulheres e homens, em 1906, quando ainda era um Principado do Império Russo (RAY, 1918). Na Inglaterra, em 1865, John Stuart Mill apresenta ao Parlamento um projeto de lei dando o voto às mulheres, que não foi aprovado. Somente em 1928, o voto feminino é autorizado nas mesmas condições às dos homens (NELSON, 2002). Nos Estados Unidos, em 1920, foi aprovada a 19^a Emenda¹, que proibia o estabelecimento de qualquer restrição ao voto (estadual e federal) baseada no sexo do(a) votante.

Na década de 1930, Mead, uma antropóloga estadunidense, problematiza a fixitude dos conceitos *feminilidade* e *masculinidade* a partir de uma pesquisa comparativa entre três sociedades tribais na Nova Guiné (MEAD, 2000). A pesquisadora conclui não haver um temperamento inato, universal que tenha origem biológica, ligada ao aparelho reprodutor. Ela observa que traços de caráter são aprendidos em sociedade, podendo, portanto, ser modificados e até desaprendidos. Ela deixa legado teórico que suporta a ideia de que existe uma construção cultural da diferença sexual.

¹ Fonte: <<http://www.archives.gov/historical-docs/document.html?doc=13&title.raw=19th+Amendment+to+the+U.S.+Constitution:+Women%27s+Right+to+Vote>> Acesso em 02 de novembro de 2014.

Em 1949, a filósofa francesa Beauvoir lança a obra *O Segundo Sexo*, considerado precursor da “segunda onda feminista”(PISCITELLI, 2009). Ainda que Beauvoir (1967) não cite o conceito de “papel social” ou mesmo “papel sexual”, ela enfatiza logo de início que o “ser uma mulher” é uma construção social:

nenhum destino biológico, psíquico, econômico define a forma que a fêmea humana assume no seio da sociedade; é o conjunto da civilização que elabora esse produto intermediário entre o macho e o castrado que qualificam de feminino. (BEAUVOIR, 1967, p.09)

Em sua obra, Beauvoir (1967) tem por foco questionar a dominação masculina, sem deixar de questionar também a eficácia do movimento feminista forjado até então no combate a essa dominação. Ela julgava ser possível esse combate ser bem sucedido se fossem combatidos elementos como: forma com que mulheres eram educadas; instituição de casamentos opressores; maternidade compulsória; vigência de um duplo padrão de moralidade sexual que permitiam maior liberdade sexual somente aos homens; e falta de trabalhos dignos e bem remunerados que possibilitassem independência econômica às mulheres.

Quase que concomitantemente, nos Estados Unidos, nasce um novo par de categorias de estudos, o sexo-gênero (FRAISSE, 2001; STOLKE, 2004; HARAWAY, 2004). A distinção entre as características biológicas e as características sociais torna-se mais difundida, ou seja, na academia e na sociedade passa a ser considerada a noção de que posturas sociais de identidade masculina ou feminina não estabelecem relação biunívoca com o sexo anatômico. A denominação dessa construção cultural pela palavra gênero ocorre em 1958, na Califórnia, quando foi empreendida uma pesquisa acerca da identidade de gênero no *California Gender Identity Center*. Os resultados foram apresentados pelo psicanalista Robert Stoller em 1963, no Congresso de Psicanálise de Estocolmo. Essa mesma pesquisa embasou a elaboração do primeiro volume de *Sex and Gender* de ?? (??). Essa obra expôs o quanto a relação sexo e gênero não é automática, nem estrita, discorrendo ainda sobre casos em que a anatomia da genitália não seria compatível com a identidade masculina ou feminina da pessoa. Assim, Stoller formula um conceito de gênero ligado à cultura, enquanto o conceito de sexo permanece ligado à morfologia corporal.

Em 1970 e 1980, o debate sobre esse par de categorias (sexo-gênero) toma espaço na comunidade acadêmica estadunidense. A antropóloga Rubin (1975) introduz a categoria gênero no debate sobre opressões sociais sofridas pelas mulheres por meio do seu ensaio *The Traffic in Women: Notes on the 'Political Economy' of Sex*. Nessa obra, Rubin faz uma análise marxista sobreposta ao sistema sexo-gênero da qual depreende que no sistema de trocas capitalista, os homens estabelecem-se como vendedores e as mulheres são estabelecidas como mercadorias para serem trocadas. Rubin dialoga com Lévi-Strauss que aponta ser o casamento o dispositivo mais importante de aliança entre as famílias,

inexistente se não fosse pelo *tabu do incesto* (LÉVI-STRAUSS, 2010). Para Rubin (1975 apud PISCITELLI, 2009) esse tabu é precedido por outro, o da *homossexualidade*. Isso porque, mediante a divisão sexual do trabalho² e ao tomar como a menor unidade de sobrevivência econômica a família, tem-se necessariamente um homem e uma mulher, numa relação heterossexual de dependência mútua. Rubin discute também o trabalho doméstico, dando visibilidade a um trabalho que muitas vezes viabiliza o sustento do trabalhador (geralmente homem) sem que seja remunerada (a mulher). Por fim, ela consegue articular teoricamente gênero e sexualidade de forma que o conceito de gênero constituído até então não reside apenas em identificação com um determinado sexo, mas pressupõe que o desejo sexual seja por indivíduo do sexo oposto.

A distinção entre sexo e gênero foi extremamente útil às feministas acadêmicas, pois sinalizava um lastro teórico para embasar os estudos sobre a condição da mulher, muitas vezes inferiorizada por sua condição biológica inerente. Com isso, o questionamento à lógica binária de interpretação do mundo passou a ser menos frequente e incisivo (HARAWAY, 2004, p.218) e, porque não, superada em alguma medida. Conforme pode-se ver no trabalho de Rubin, o conceito de gênero foi além de separar dimensões culturais e biológicas de mulheres e homens. Cada vez mais o conceito de gênero passa a significar também a superação da leitura binária de mundo que só permite feminilidade ou masculinidade. Para Heilborn (1992, p.41):

A categoria de gênero não deve ser açãoada como um substituto de referência para homem ou mulher. Seu uso designa, ou deveria fazê-lo, a dimensão inerente de uma escolha cultural e de conteúdo relacional. Por outro lado, traz embutida a articulação desse código, que se apropria da articulação da diferença sexual tematizando-a em masculino e feminino, com outros níveis de significação dos universos.

Se a primeira onda do feminismo reivindicou direitos iguais, a segunda onda avançou e lutou pelo exercício igual dos direitos. Na primeira onda buscava-se provar que as diferenças entre o feminino e o masculino eram de origem social e não biológica. Tal afirmação não é abandonada na segunda onda, mas aprofundada, passando-se a buscar as origens de tais diferenças sócio-culturais. Nessa construção, segundo Piscitelli (2009, p.133-134):

A categoria “mulher” foi desenvolvida pelo feminismo da segunda onda em leituras segundo as quais a opressão das mulheres está além de questões de classe e raça, atingindo todas mulheres, inclusive as mulheres das classes altas e brancas. [...] O reconhecimento político das mulheres como coletividade ancora-se na ideia de que o que une as mulheres ultrapassa

² A expressão *divisão sexual do trabalho* foi inicialmente utilizada por etnólogos para se referir à repartição das atividades entre homens e mulheres nas sociedades que estudavam (KERGOAT, 2004). Esta autora afirma ainda que “a divisão sexual do trabalho é aquela decorrente das relações sociais de sexo”, o que será explorado mais adiante neste capítulo.

em muito as diferenças entre elas. Isso criava uma “identidade” entre elas.

Se essa uniformização entre as mulheres foi útil para forjar uma união na conquista por direitos, em meados da década de 1970 e início dos anos 1980, já era questionada. Feministas negras e mulheres de países subdesenvolvidos (FURTADO, 2009) cada vez menos identificavam-se com o arcabouço teórico hegemônico e homogêneo apresentado por feministas dos países do “norte rico”, inclusive por Rubin. Assim, a “terceira onda feminista” desdobra-se em feminismos diversos. Afinal, as mulheres negras contam com trajetória histórica diferente das mulheres brancas, grande parte das vezes tendo a escravidão e suas consequências como parte determinante da vida de sua ancestralidade (HOOKS, 1990; CRENSHAW, 2002). No caso de países subdesenvolvidos, como o Brasil, não cabe comparar *ipsis literis* a trajetória das mulheres (mesmo brancas) brasileiras com as europeias. A título de ilustração, o estudo de Pinto (2004) apresenta como as mulheres brasileiras são vistas como mais maternais, com vocação para a domesticidade e muito mais “racializadas” do que as portuguesas.

Oferecendo alguma resposta a essas demandas por interseccionalidade³ em 1986, a historiadora pós-estruturalista Joan Scott publica seu artigo *Gender: A Useful Category of Historical Analysis* em que faz uma leitura crítica da utilização do termo *gênero* como categoria de análise e relaciona necessariamente esta categoria a outras como classe e raça, pois demonstra ser o gênero necessariamente imbricado a relações hierarquizadas de poder:

a oposição binária e o processo social das relações de gênero tornam-se, ambos, partes do sentido do próprio poder. Colocar em questão ou mudar um aspecto ameaça o sistema por inteiro. Se as significações de gênero e de poder se constroem reciprocamente, como é que as coisas mudam? [...] o gênero tem que ser redefinido e reestruturado em conjunção com uma visão de igualdade política e social que inclui não só o sexo, mas também, a classe e a raça. (SCOTT, 1986, p.1073,1075)

É então necessário olhar a construção das identidades de gênero à luz das relações de poder e olhar brevemente como se deu a evolução dos direitos, especialmente na sociedade brasileira. As mulheres no Brasil escravocrata dispunham de uma grande imobilidade geográfica e mesmo as mulheres das classes dominantes raramente saíam às ruas e, quando o faziam, nunca estavam desacompanhadas (SAFFIOTI, 2013). Mulheres e homens de então desfrutavam de maneira assimétrica do direito de ir e vir.

Na campo dos direitos políticos, o movimento sufragista das brasileiras não teve tanta capilaridade nem foi um movimento de massas como nos Estados Unidos, Inglaterra

³ Interseccionalidade ou abordagem interseccional, segundo Crenshaw (2002, p.177) “trata especificamente da forma pela qual o racismo, o patriarcalismo, a opressão de classe e outros sistemas discriminatórios criam desigualdades básicas que estruturam as posições relativas de mulheres, raças, etnias, classes e outras”.

ou Rússia. Ele teve início na década de 1910, quando o Partido Republicano Feminino é fundado no Rio de Janeiro com o objetivo de instaurar o debate acerca do voto feminino⁴. A igualdade de condições de voto entre homens e mulheres se concretiza em 1932, pelo Decreto nº 21.076 que autoriza o voto a qualquer cidadã ou cidadão com idade superior a 21 anos. A eleição de 1933 foi a primeira em que mulheres puderam participar do pleito, votando e sendo votadas, como Carlota Pereira Queiroz, a primeira deputada brasileira, que participou da Assembleia Nacional Constituinte entre 1934 e 1935 (TABAK, 1989).

Embora o direito ao voto tenha sido emblemático, a ideia de desfrutar de *direitos iguais* na sociedade, mulheres e homens, tratava também de outros direitos como o acesso à educação e poder ter posse de bens - por muito tempo, de acordo com a lei, só homens podiam ser proprietários de casas, por exemplo (PISCITELLI, 2009). Subjacente a esses questionamentos das mulheres tecia-se o conceito de “papel social”, bastante difundido a partir da década de 1930. Para Piscitelli (2009, p.127), a teoria dos papéis sociais buscava:

compreender os fatores que influenciam o comportamento humano. A ideia é que os indivíduos ocupam posições na sociedade, desempenhando papéis de filho, de estudante, de avô. [...] A ideia de posições ocupadas no desempenho dos papéis faz referência a categorias de pessoas que são reconhecidas coletivamente. Um dos atributos que podem servir de base para a definição dessas categorias é a idade. [...] Outro desses atributos pode ser o sexo. Nesse caso, homens e mulheres desempenham papéis culturalmente construídos: os papéis sexuais.

Essa busca por um leque de direitos não foi um movimento só das mulheres, mas um movimento de luta por cidadania⁵. Sob o ponto de vista de gênero e cidadania, Brito (2001) relembra que o conceito clássico de cidadania, o grego, excluía mulheres e escravos. Ela pontua que ao longo da história as identidades de homens e mulheres foram construídas pressupondo uma dicotomia entre o âmbito público e o privado. Blay (2001) relata que até os anos 1960/1970 era um fator negativo para a mulher participar da vida pública. A partir de 1970, com o movimento feminista passa a haver críticas e questionamentos quanto à natureza, à separação e à natural atribuição dessas esfera a um determinado sexo.

⁴ Bertha Lutz, filha do cientista Adolfo Lutz, licenciou-se em Ciências Naturais na Sorbonne de Paris e, ao retornar ao Brasil, funda a Federação Brasileira pelo Progresso Feminino, em 1919, que leva adiante a luta pelo sufrágio feminino (PINSKY; PEDRO, 2003). A primeira cidade a autorizar o voto feminino em eleições foi Mossoró (RN), em 1928. Em nível nacional, Getúlio Vargas autoriza em 1931 o voto feminino apenas às mulheres solteiras, viúvas com renda própria ou casadas com a autorização do marido.

⁵ A cidadania, para Carvalho (2002), é entendida como o exercício pleno de três direitos: direitos civis, direitos sociais e direitos políticos. Os civis são aqueles considerados direitos fundamentais, como o direito à vida, à liberdade, à propriedade, à igualdade perante a lei. Eles garantem a vida em sociedade e dependem da existência de uma justiça independente, eficiente, barata e acessível a todos. Os políticos se referem à participação do cidadão no governo da sociedade. Seu exercício é limitado a uma parcela da população definida por idade, por exemplo, e consiste na capacidade de fazer demonstrações políticas, de organizar partidos, de votar, de ser votado. Por fim, os sociais são aqueles que garantem a participação na riqueza coletiva e se baseia na ideia de justiça social. Incluem os direitos à educação, ao trabalho, ao salário justo, à saúde, à aposentadoria.

Assim, elabora-se uma perspectiva de análise a partir do gênero, e não do sexo biológico, pois o conceito de gênero também comprehende as dimensões social e política do termo.

A partir dos anos 1990 o uso da categoria gênero tornou-se mais frequentemente utilizada no Brasil e, cada vez mais influenciada pelas diversas escolas de psicanálise para explicar a produção e a reprodução da identidade de gênero do sujeito. A psicanalista brasileira [Kehl \(1998\)](#) embora não tenha como central esse debate, participa dele e em sua obra *Deslocamentos do Feminino*, ao invés de apartar sexo de gênero, assinala que gênero é um conceito que inclui a dimensão biológica do sexo, não sem somar-lhe atributos que a cultura provê. A partir de Scott é cada vez mais corrente incorporar a dimensão da política e do poder na composição do conceito de gênero, conforme explicita [Moraes \(1998, p.100\)](#):

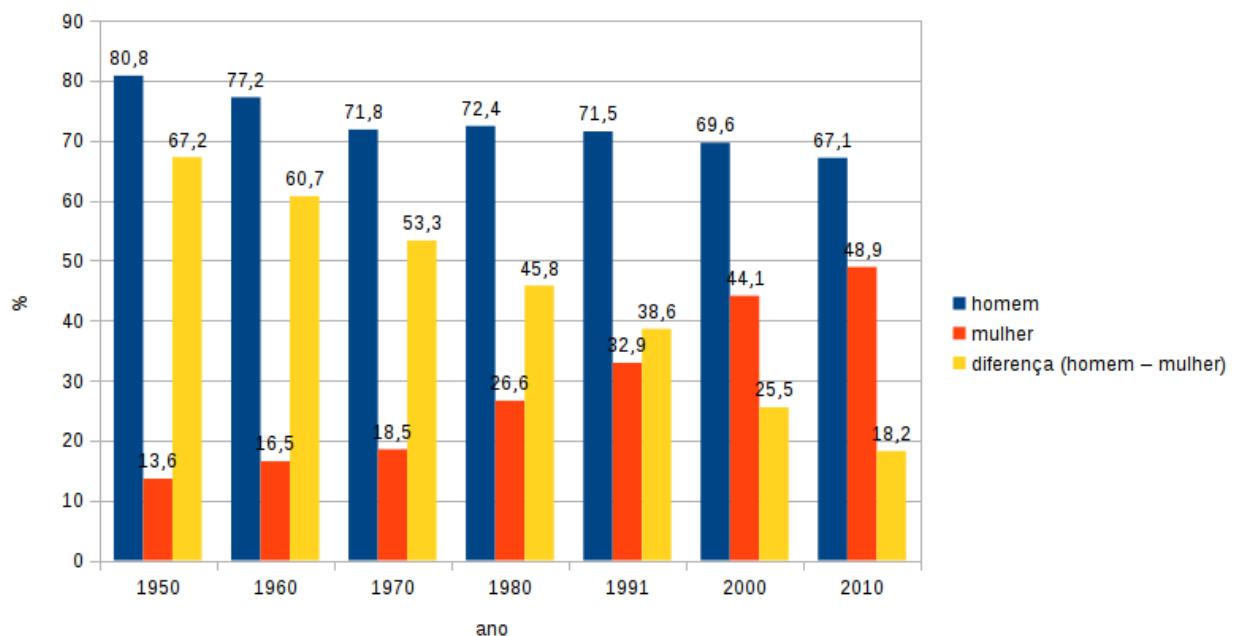
A expressão relações de gênero, tal como vem sendo utilizada no campo das ciências sociais, designa, primordialmente, a perspectiva culturalista em que as categorias diferenciais de sexo não implicam o reconhecimento de uma essência masculina ou feminina, de caráter abstrato e universal, mas, diferentemente, apontam para a ordem cultural como modeladora de homens e mulheres. Em outras palavras, o que chamamos de homem e mulher não é o produto da sexualidade biológica, mas sim de relações sociais baseadas em distintas estruturas de poder.

Essas relações de poder incidem tanto sobre as relações que se desdobram no espaço público, quanto as do espaço privado. No espaço público, ou não-doméstico, tem relevância para o presente estudo o mercado de trabalho. Ao longo do tempo, a urbanização e a industrialização levaram à ampliação da classe média e ao crescimento do consumo no Brasil. As mulheres entraram também neste processo, embora a maior parte das trabalhadoras tenha sido absorvida, ao menos inicialmente, no setor de serviços e com enorme concentração nos empregos domésticos, de menor rendimento. Constatase assim que existe aqui uma divisão sexual do trabalho ([KERGOAT, 2004](#)), que tem por características a destinação prioritária dos homens à esfera produtiva e das mulheres à esfera reprodutiva. Essa forma de divisão pauta-se em dois princípios: o da separação e o da hierarquização. O princípio da separação explicita a ideia de que há “trabalhos de mulheres” e “trabalhos de homens” enquanto o princípio da hierarquização indica existir uma diferença de valoração entre o trabalho do homem (produtivo, mais valioso) e o da mulher (reprodutivo, menos valioso). [Blay \(2001, p.84\)](#) fala sobre a mulher brasileira:

Até a década de 1960 a história, quando focalizava a mulher, atinha-se às supostas atividades femininas fundamentais, isto é, às de um ser apêndice da família. A historiografia simplesmente ignorava a participação feminina no mercado de trabalho, a enorme freqüência com que sustentavam economicamente a si e aos seus.

Ao longo do século XX, observou-se aumento de mulheres na população economicamente ativa brasileira⁶ (ver Gráfico 1). O fenômeno permanece no século XXI de acordo com estudo da Fundação Perseu Abramo (2010): (i) observa-se que de 2001 a 2010 manteve-se a preponderância feminina em ocupações que demandam de 20 a 40 horas semanais; (ii) para homens, manteve-se o predomínio histórico de jornada superior a 40 horas semanais; (iii) o ingresso das mulheres no mercado de trabalho não alterou drasticamente o papel delas na família e, portanto, nas atividades ligadas às tarefas domésticas. Isto é, apesar de muitas mulheres terem entrado no mercado de trabalho algumas décadas atrás, elas ainda são responsáveis pela maior parte do trabalho doméstico. No Gráfico 2 é possível constatar não apenas essa divisão sexual do trabalho - produtivo, no mercado, e reprodutivo, no lar - mas também que as jornadas totais que acabam ficando a cargo da mulher são maiores. Os homens acumulam uma jornada de cerca de 50 horas por semana, as mulheres, 57 horas semanais.

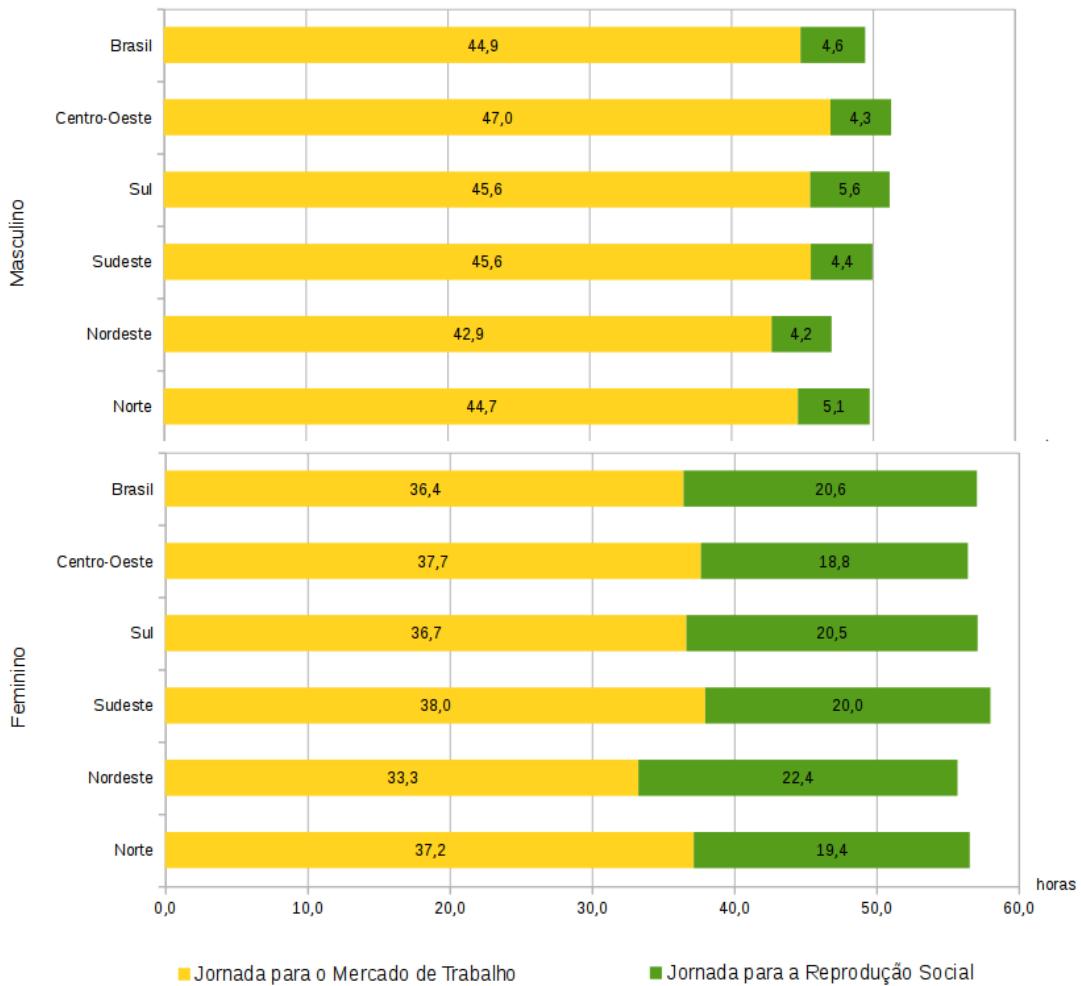
Gráfico 1 – Percentual de indivíduos que fazem parte da PEA, por sexo, no Brasil, entre 1950 e 2010



Fonte: Adaptado de (ALVES, 2013)

⁶ A população economicamente ativa (PEA) é obtida pela soma da população ocupada e desocupada com 16 anos ou mais de idade. “População ocupada” compreende as pessoas que, num determinado período de referência, trabalharam ou tinham trabalho mas não trabalharam (por exemplo, pessoas em férias). “População desocupada” compreende as pessoas que não tinham trabalho, num determinado período de referência, mas estavam dispostas a trabalhar, e que, para isso, tomaram alguma providência efetiva nos últimos 30 dias (consultando pessoas, jornais, etc.). Fonte: IBGE - disponível em <<http://www.ibge.gov.br/apps/snig/v1/?loc=0,355030&cat=118,119,1,2,-2,-3&ind=87>> Acesso em 21 de novembro de 2014

Gráfico 2 – Jornadas Médias para o Mercado de Trabalho e para Reprodução Social, por sexo, raça/cor e região geográfica, no Brasil, em 2003



Fonte: PNAD (2003 apud [Soares e Pinheiro, 2003](#))

Nota: Segundo Dear e Scott (1981), o espaço de reprodução é onde a recuperação da força de trabalho ocorre sendo a residência o local principal a ser considerado; e o espaço de produção é onde o processo de acumulação do capital ocorre, ou seja, no que se denomina mercado de trabalho (indústria, comércio e serviços no geral).

Esse é o ponto em que a atuação no espaço público e a no espaço privado vincula-se. A mulher passa a poder desempenhar atividades antes tidas como “masculinas”, porém sem ser desonerada de desempenhar as atividades tidas como “femininas”, pois ainda “persistem nichos onde vigora uma imagem feminina vinculada à maternidade e ao cuidado da família, à saúde da prole” (BLAY, 2001, p.94). Assim, a ampliação do leque de papéis sociais que a mulher desempenha impacta as relações de poder dentro do ambiente doméstico, dentro da família. Isso molda as necessidades, interesses, atividades e padrão de viagens dos integrantes da família, a partir das identidades de gênero constituídas, forjadas pelos comportamentos de indivíduos e da relação de poder estabelecida entre eles.

2.2 Mobilidade Urbana

A palavra mobilidade, de acordo com o Dicionário Michaelis, significa “(i) propriedade do que é móvel ou do que obedece às leis do movimento; (ii) deslocamento de indivíduos, grupos ou elementos culturais no espaço social; (iii) movimento comunicado por uma força qualquer; (iv) falta de estabilidade, de firmeza ou inconstância”. Tal definição reflete toda uma gama de conceitos relacionados a movimento e/ou deslocamento, o que na área de transportes relaciona-se imediatamente a viagens. No Brasil, há cerca de 100 anos, a maior parte das viagens de pessoas era feita a pé⁷ e, quando muito, usava-se tração animal (cavalo ou boi), especialmente para cargas. Isso incorria em baixas velocidades de deslocamento e assim, grande parte das pessoas acabavam por desenvolver suas atividades, por toda vida, nas proximidades de onde nasceram. Neste último século o cenário mudou bastante, as mais diversas tecnologias se desenvolveram, os rendimentos aumentaram, a mobilidade aumentou (METZ, 2012, p.06), e como exemplos icônicos dessa maior mobilidade figuram a utilização do carro e do avião.

O conceito de mobilidade pode englobar muitos outros e se desdobrar em uma grande diversidade de temas. Há quem o aborde ligando-o ao turismo (ENLOE, 1989; FROHLICK, 2008), enquanto outros (CHANT, 1992; SILVEY, 2000) abordam-no sob a perspectiva dos movimentos migratórios entre países ou dentro de uma mesma nação. Outro uso do termo é ligado ao intercâmbio de estudantes e pesquisadores de diferentes instituições de origem – o que dá origem à expressão *mobilidade acadêmica* (ENDERS, 1998; TREMBLAY, 2005; HOFFMAN, 2008). Ademais, há um olhar sobre a mobilidade em que as condições geodemográficas são elementos de contorno, delimitando assim as áreas da mobilidade rural e urbana.

Embora pareçam óbvias à primeira vista – porque em alguma medida vividas – as diferenças entre rural e urbano são bem menos claras quando olhadas mais de perto. No Brasil as distinções nascem de critérios político-administrativos, originados em decreto de 1938⁸ de Getúlio Vargas e, até hoje, no Brasil, baseia-se em critérios políticos administrativos. Segundo o IBGE, “como situação urbana consideram-se as áreas correspondentes às cidades (sedes municipais), às vilas (sedes distritais) ou às áreas urbanas isoladas”⁹. Como rural, classifica-se tudo o que não se configure urbano. Trata-se de definição legal (jurídica) ocorrida frequentemente no campo da política e passível de críticas, como a de (GRABOIS et al., 2001) que aponta que tal classificação não considera

⁷ Os primeiros carros foram montados em São Paulo pela Ford na década de 1910. Fonte: <http://www.carroantigo.com/portugues/conteudo/curio_hist_carro_brasileiro.htm> Acesso em 25 de outubro de 2014

⁸ Decreto-Lei nº 311, de 2 de Março de 1938 disponível em: <<http://www2.camara.leg.br/legin/fed/decrei/1930-1939/decreto-lei-311-2-marco-1938-351501-publicacaooriginal-1-pe.html>> Acesso em 06 de novembro de 2014

⁹ Conceitos adotados no censo do IBGE disponíveis em: <<http://www.ibge.gov.br/home/estatistica/populacao/contagem/conceitos.shtm>>

as diferentes funções dos aglomerados como critério. A questão da definição do que é rural vai além da abordagem teórica e tem como pano de fundo as diferenças de tributação entre as áreas rural e urbana. Como saída a essa arbitrariedade que fica a cargo dos poderes municipais, Veiga (2002) elenca três critérios que entende importantes a se considerar nesse tipo de classificação: (i) população total do município, (ii) densidade demográfica e (iii) localização.

O objeto deste trabalho é a RMSP, composta por 39 municípios¹⁰; todos, atualmente, contam com áreas consideradas urbanas. Muito embora se vá seguir as classificações oficiais do IBGE e do Metrô-SP, entende-se como salutar esta breve discussão sobre o significado do que é ser área urbana ou rural, pois “o espaço rural tem passado recentemente por um conjunto de mudanças com significativo impacto sobre suas funções e conteúdo social” (MARQUES, 2002, p.96). Deixa-se a ressalva de que mesmo na área de enfoque, urbana, encontram-se atividades agrícolas (comumente tidas como rurais), afinal, são cada vez mais imprecisos os limites entre um e outro (MINGIONE; PUGLIESE, 1987). Um outro fenômeno que merece alguma atenção, dada a natureza deste trabalho, é o êxodo rural seletivo que vem sendo constatado por alguns pesquisadores. (FROELICH et al., 2011) constata que no Rio Grande do Sul a emigração do campo é desigual em gênero e em idade: mulheres e jovens migram mais, homens e idosos são os que permanecem no campo, nas atividades rurais. Fenômeno semelhante é constatado em Santa Catarina¹¹, e em alguns países europeus¹². Ou seja, existe diferença relacional entre a mobilidade feminina e a masculina expressa, por exemplo, nos deslocamentos campo-cidade. O recorte deste trabalho, entretanto, concentra-se majoritariamente nos deslocamentos intra-urbanos, expressos amplamente pelo conceito de mobilidade urbana.

Se “o estudo dos problemas urbanos é indissociável da relação campo-cidade” (FREITAG, 2007, p.154) e por isso, independente da época estudada, é preciso tê-la em mente; especificamente a mobilidade urbana é um elemento fundamental para que seja possível garantir aos habitantes de uma cidade acesso aos bens que lhes oferece (IEMA,

¹⁰ Os 39 municípios que compõem a RMSP são agrupados em 6 regiões de acordo com Lei Complementar estadual nº 1.139, de 16 de junho de 2011. Na região central está São Paulo. Na região Sudoeste encontram-se 8 municípios, a saber, Juquitiba, São Lourenço da Serra, Embu-Guaçu, Itapecerica da Serra, Embu, Tabão da Serra, Cotia e Vargem Grande Paulista. Na Região Oeste encontram-se 7 municípios, a saber, Pirapora do Bom Jesus, Santa de Parnaíba, Barueri, Jandira, Itapevi, Carapicuíba, Osasco. Na região Norte encontram-se 5 municípios, a saber, Cajamar, Caieira, Franco da Rocha, Francisco Morato, Mairiporã. Na região Leste encontram-se 11 municípios, a saber, Santa Isabel, Arujá, Guarulhos, Itaquaquecetuba, Guararema, Poá, Suzano, Ferraz de Vasconcelos, Mogi das Cruzes, Biritiba Mirim, Salesópolis. Na região Sudeste encontram-se 7 municípios, a saber, Santo André, São Bernardo do Campo, São Caetano do Sul, Diadema, Mauá, Ribeirão Pires, Rio Grande da Serra.

¹¹ Éxodo seletivo é retratado em Santa Catarina pelo documentário “Celibato no Campo” de Cassemiro Vitorino e Ilka Goldschmidt, 2013, disponível em <<http://www2.camara.leg.br/camaranoticias/tv/materias/OLHARES/440520-CELIBATO-NO-CAMPO.html>> Acesso em 15 de outubro de 2014.

¹² Relatório do Parlamento Europeu em 2003 apontava que somente 37% da mão-de-obra rural da União Europeia era de mulheres. Disponível em <<http://www.europarl.europa.eu/sides/getDoc.do?type=REPORT&reference=A5-2003-0230&format=XML&language=PT>> Acesso em 16 de outubro de 2014

2010). Cabe aqui diferenciar o que seja mobilidade do que seja acessibilidade. A mobilidade exprime a capacidade de se deslocar no espaço, “refere-se à habilidade de mover-se entre dois diferentes locais de atividade” (HANSON, 1995b, p.04). Sobre a acessibilidade e sua ligação com a mobilidade, Hanson afirma ainda:

A acessibilidade refere-se ao número de oportunidades [...] disponíveis dentro de uma determinada distância ou tempo de viagem. [...] Conforme as distâncias entre os locais de atividades se tornam maiores [...] a acessibilidade passa a depender cada vez mais da mobilidade, particularmente daquela relacionada aos veículos particulares. (HANSON, 1995b, p.04)

Dessa maneira, vinculando a mobilidade à motorização, Hanson afirma que é possível promover acessibilidade sem incrementar a mobilidade (1995, p.05), afinal, ter toda uma sorte de serviços próximos à residência daria a possibilidade de ir à padaria, ao mercado, à igreja, à escola, à livraria, etc. a pé. A autora ainda se aprofunda na questão da acessibilidade ao classificá-la como (i) de pessoas ou (ii) de lugares. Trata-se apenas de diferentes referenciais, a acessibilidade de uma pessoa indica o quanto fácil ou difícil é para ela chegar a determinado local; a acessibilidade de um lugar mostra o quanto fácil ou difícil pode ser alcançá-lo. Essas expressões “fácil” e “difícil”, entretanto, são formas muita genéricas para caracterizar o que se deseja exprimir. Problema para o qual ela apresenta como resposta uma medida de acessibilidade A_i , onde A_i é o conjunto das oportunidades O_i ponderadas pelas distâncias $d_{i,j}$ da residência da pessoa i :

$$A_i = \sum_i O_i d_{i,j}^{-b} \quad (2.1)$$

Caso considere-se ao invés de i indivíduos, i zonas, a Equação 2.1 referir-se-á à zona i de análise. Como este é um modelo bastante simplificado, indica apenas um potencial de acessibilidade e tem suas limitações, como por exemplo, desconsiderando a dimensão temporal dos deslocamentos. Vasconcellos (2012), por sua vez, pontua a questão temporal em sua definição de acessibilidade:

medida pela quantidade e/ou diversidade de destinos que a pessoa consegue alcançar, por certa forma de transporte, em determinado tempo. Quanto maior for esta quantidade, maior é a acessibilidade, ou seja, mais oportunidades as pessoas terão para realizar atividades desejadas ou necessárias. (VASCONCELLOS, 2012, p.42)

O mesmo autor, em obra anterior, indica como forma de mensurar a acessibilidade, a soma dos tempos: (i) de deslocamento até o meio de transporte; (ii) tempo de espera, caso exista; (iii) tempos(s) dentro do(s) meio(s) de transporte; (iv) tempo de transferência entre diferentes meios de transportes, caso exista; (v) tempo após saída do meio de transporte até atingir o destino final. Destes tempos, ele classifica os itens (i) e (v) como

microacessibilidade, ou seja, itens que referem-se “à facilidade relativa de ter acesso aos veículos ou destinos desejados (por exemplo, condições de estacionamento ou acesso ao ponto de ônibus)” (2001, p.91). A **macroacessibilidade** é definida por ele como a:

facilidade relativa de atravessar o espaço e atingir construções e equipamentos urbanos desejados. Ela reflete a variedade de destinos que podem ser alcançados e, consequentemente, o arco de possibilidades de relações sociais, econômicas, políticas e culturais, dos habitantes do local. (VASCONCELLOS, 2001, p.91)

Isto posto, a acessibilidade pode ser entendida como a capacidade de se chegar onde se deseja e, para sua mensuração pode-se usar a distância e/ou o tempo. Uma das formas de reunir essas duas dimensões é através do prisma espaço-tempo. Na Figura 2 pode-se observar o diagrama do tempo em função da distância¹³ de um casal hipotético com filho pequeno. Supõe-se que ambos trabalhem das 8 às 18 horas, que seja a mulher a levar e buscar a criança na escola e que haja um carro na família. A Figura 2 descreve uma situação em que o pai fica com o carro da família. Assim, a mulher leva a criança na escola a pé¹⁴ e segue para o trabalho de ônibus enquanto seu marido segue para o trabalho de carro, ambos no sentido bairro-centro¹⁵. Na volta, ele retorna de carro diretamente para a residência e ela sai do trabalho de ônibus, apanha a criança na escola, para depois irem a pé para casa; todos no sentido centro-bairro.¹⁶ São acessíveis à mulher as oportunidades contidas na área de hachura vermelha. A área azul indica o diferencial de acessibilidade que o homem tem neste caso. Ou seja, a cidade possível no que diz respeito a oportunidades de trabalho, escola, lazer, etc. ao alcance dessa mulher é quase a metade daquela desse homem.

A situação narrada é hipotética, mas é ilustrativa de alguns fatores que restringem as liberdades de movimento, segundo Hagerstrand (1970 apud HANSON, 1995b):

- (i) limitações devido ao fato que não se pode estar em dois lugares ao mesmo tempo e que certas tarefas precisam ser feitas usando um determinado modo de transporte (por algum motivo pode não haver outras opções);
- (ii) necessidade de encaixar os compromissos de uma pessoa com os de outra(s) pessoa(s), como por exemplo, levar filho(a)(s) à escola, acompanhar idoso(a)(s) ao médico ou almoçar com amigo(a)(s);

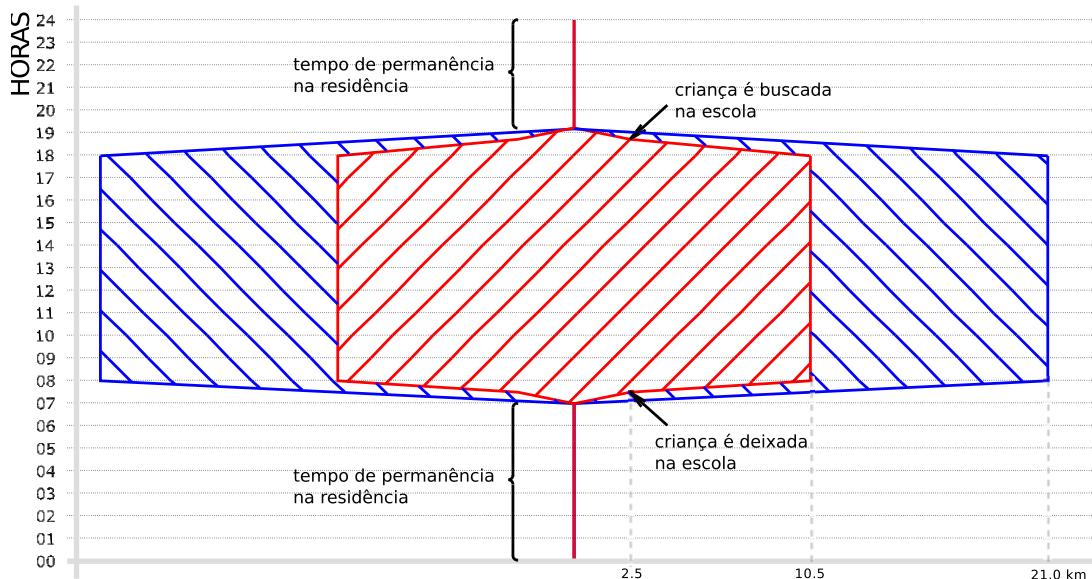
¹³ Para ser efetivamente um prisma, tal gráfico deveria ser em três dimensões: na base x e y representariam os deslocamentos no plano, e na altura, z, teríamos o tempo.

¹⁴ Considerou-se que a velocidade média ao caminhar seja de uma velocidade média de 5km/h. Fonte: <<http://www.anpet.org.br/xxviiianpet/anais/documents/AC301.pdf>> Acesso em 03 de dezembro de 2014.

¹⁵ Para esta simulação ilustrativa foram usadas velocidades do pico da manhã de 16km/h para ônibus e de 21km/h para carros, de acordo com dados da CET. Fonte: <<http://www.cetsp.com.br/media/228073/2007%20%20volumes%20e%20velocidades.pdf>> Acesso em 06 de dezembro de 2014.

¹⁶ Para esta simulação ilustrativa foram usadas velocidades do pico da tarde de 11km/h para ônibus e de 19km/h para carros, de acordo com dados da CET. Fonte: <<http://www.cetsp.com.br/media/228073/2007%20%20volumes%20e%20velocidades.pdf>> Acesso em 03 de dezembro de 2014.

Figura 2 – Prisma Espaço Tempo

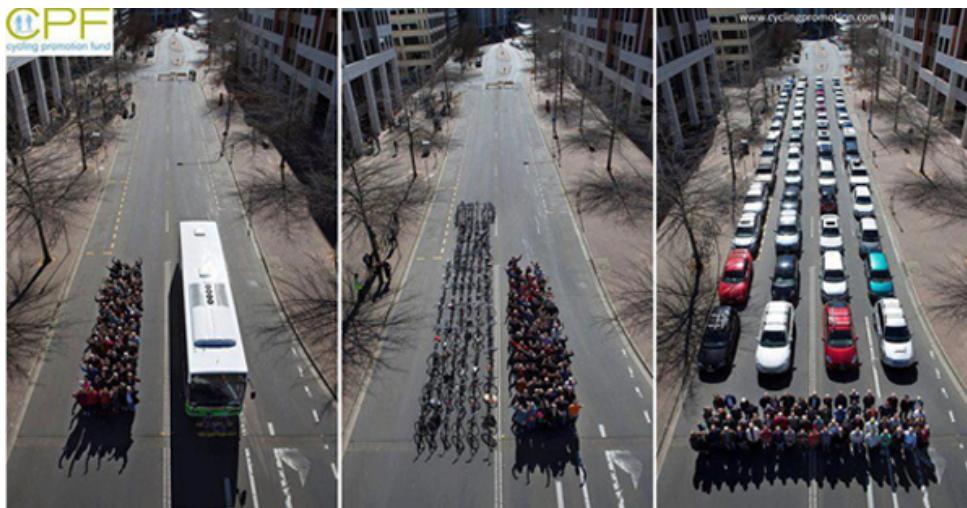


Fonte: Elaboração própria

- (iii) restrições devido à autoridade social, política e/ou legal no acesso a algum lugar
 - pode haver regras explícitas (ou implícitas), por exemplo, que impeçam as pessoas de andarem à noite sozinhas num determinado local.

Só que as pessoas geralmente não dispõem de várias alternativas modais ou porque não têm recursos (ou para ter um carro ou para morar em área bem servida de transporte público) ou porque se sentem inseguras utilizando algum modo específico (medo de ser assaltado(a) no carro, de ser atropelado(a) de bicicleta, de ser assediada(a) no transporte público, entre outros). Não são todos(as) que podem arranjar horários flexíveis de trabalho e/ou de estudos para conciliar as outras atividades da forma mais eficiente, por exemplo, evitando os deslocamentos nos horários de pico. Enfim, as restrições incidem diferentemente nos grupos sociais criando assimetrias no acesso às oportunidades. Essa iniquidade desdobra-se no espaço: na distribuição desigual de empregos nas cidades, na variação de preço do solo urbano, na densidade heterogênea de serviços públicos oferecidos nos diversos bairros e mesmo na ocupação do espaço de circulação (ver Figura 3).

Figura 3 – Quantidade de espaço viário requerido para transportar 60 pessoas por ônibus, bicicleta e carro.



Fonte: Foto de *Cycling Promotion Fund*, disponível em <<http://www.bhtrans.pbh.gov.br/portal/page/portal/portalpublico/Temas/ObservatorioMobilidade/FiquePorDentro/ObsMobBH%20A%20cidade%20com%20menos%20carros>> - acesso em 22 de novembro de 2014.

Em 1991, o estrato com os 20% de menor renda da população perfazia 9% das milhas viajadas nos Estados Unidos (por carro e ônibus), ao passo que o estrato com os 20% de maior renda concentrava 32% do total de milhas (CAMERON, 1994 apud HANSON, 1995b). Vasconcellos (2001) apresenta o consumo de espaço, por modo de transporte e renda da RMSP em 1987 e 1997 (ver Tabela 1). Observa-se que na RMSP os grupos de maior renda tendem a consumir mais espaço de circulação, o que levanta a questão do quão (in)justo é esse cenário, principalmente no Brasil, onde o sistema de tributação que custeia a infra-estrutura pública urbana é regressivo¹⁷.

Segundo Urry (2004 apud CRESSWELL; UTENG, 2008), tendo em vista as iniquidades urbanas de acesso, estruturadas socialmente, há cinco “mobilidades” bastante interdependentes:

- (i) viagem corpórea das pessoas por motivo de trabalho, lazer, etc.;
- (ii) movimento físico de objetos (cargas);
- (iii) viagem imaginativa a lugares por meio de imagens (fotos ou televisão);
- (iv) viagem virtual mediante uso da internet;
- (v) viagem comunicativa através de mensagens trocadas entre pessoas (cartas, mensagens de celular, telefone).

¹⁷ O Brasil conta com um sistema de tributação regressivo, ou seja, aquele em que a retirada é proporcionalmente maior das pessoas com menor capacidade de contribuir (GRECO; GODOI, 2005).

Tabela 1 – Consumo dinâmico de espaço por modo e renda na RMSP

Renda familiar mensal (1987)	Espaço dinâmico (km*m ² /dia/pessoa) (1987)	Renda familiar mensal (1997)	Espaço dinâmico (km*m ² /dia/pessoa) (1997)
0 a 240	7,6	0 a 250	9,2
241 a 480	13,4	251 a 500	14,6
481 a 900	25,1	501 a 1000	23,7
901 a 1800	42,2	1001 a 1800	36,7
1801 ou mais	74,8	de 1801 a 3600	56,2
-	-	3601 ou mais	81,9

Fonte: Adaptado de ([VASCONCELLOS, 2001](#), p.181;196)

Nota: Considerando consumo médio de $1,0m^2/pessoa$ em transporte público ($30m^2$ de área de ônibus para 30 passageiros em média) e $6,6m^2/pessoa$ em transporte privado ($10m^2$ de área de carro para 1,5 passageiros em média).

Não é possível considerar então a mobilidade do indivíduo, isolando-o do seu contexto social, econômico, político e cultural; muito pelo contrário, só é possível entendê-la se considerarmos os ambientes em que o indivíduo se anora: doméstico, familiar e social ([HANSON, 2010](#)). Portanto, acrescido do significado de urbano definido pelo IBGE, trabalha-se aqui com o conceito de mobilidade relacionado ao item (i) de [Urry \(2004 apud CRESSWELL; UTENG, 2008\)](#), mais detalhado no artigo *Gender and mobility: new approaches for informing sustainability* de [Hanson](#), que emprega o termo **mobilidade** para designar:

o movimento de pessoas de um lugar para outro lugar no decorrer da vida cotidiana [...] [sendo a] principal preocupação com as viagens pessoais que compõem a rotina diária de atividades como o trabalho (remunerado e não remunerado), lazer, socialização e compras. ([HANSON, 2010](#), p.7)

2.3 Intersecções e Sobreposições

Não apenas o movimento feminista e o embrião da concepção de gênero datam do final do século XIX; a intersecção entre gênero, mobilidade e sustentabilidade também. Em 1895, Willard publicou seu livro *A Wheel whithin a Wheel* em que narra como ela, mulher, aos 53 anos, aprendeu a andar de bicicleta (ver Figura 4). Ela não escreveu um livro sobre mobilidade, nem sobre gênero, muito menos sobre sustentabilidade. Porém, ela aborda essas questões a partir dessa sua experiência. Ela toca na questão de gênero, por exemplo, ao falar do vestuário de uma ciclista:

Se as mulheres pedalarem, ao fazê-lo elas devem vestir-se mais racionalmente do que foram acostumadas. E se elas fizerem isso, muitos preconceitos concernentes ao que elas estariam autorizadas a vestir cairão por terra. (Livre tradução de Willard, 1895, p.39)

Figura 4 – Frances Willard aprendendo a andar de bicicleta



Fonte: (WILLARD, 1895, p.56)

O que se esperava que uma mulher vestisse àquela época? Como pedalar com vestidos tão longos que sempre cobriam os pés e contavam com muitos babados, plissados, franjas e passamanarias¹⁸? Andar de bicicleta não foi para ela apenas um desafio por conta da habilidade manual requerida e/ou idade que possuía. Ao longo da obra ela retrata não apenas que ganhou mobilidade, ela adquiriu auto-confiança e vislumbrou possibilidades que

¹⁸ Essas características da indumentária feminina utilizada no final do século XIX podem ser percebidas ora numa pintura de Cézanne (*Madam Cézanne num vestido vermelho*, de 1888/1890), ora nas peças expostas no Museu da Moda em Canela (<http://www.museudamodadecanela.com.br/> - acesso em 15 de novembro de 2014) ou no *The Metropolitan Museum of Art* (http://www.metmuseum.org/toah/hd/wrth/hd_wrth.htm - acesso em 15 de novembro de 2014).

antes não reconhecia, como aspirações relativas a seu crescimento pessoal. Ela constatou que a imobilidade física feminina atava-se a outras imobilidades, sociais. Assim, transformar a forma de se mover no espaço era (e é) também uma forma de transformar as relações de gênero (HANSON, 2010). E aqui o termo gênero é pertinente na análise da obra, e não anacrônico, porque tem no cerne as relações de poder estabelecidas entre indivíduos (SCOTT, 1986), em função do que se entende por “feminino” e “masculino”. E embora a preocupação a respeito de sustentabilidade seja recente, extemporânea a Willard, vale notar que a liberdade adquirida por ela dá-se ao aprender a andar de bicicleta, um modo de transporte não motorizado, com emissões nulas de gases do efeito estufa, de manutenção pouco custosa e, cada vez mais, símbolo de sustentabilidade.

Os primeiros artigos que abordam explícita e articuladamente questões de gênero e de transporte datam do fim da década de 1970, como o editorial de Rosenbloom (1978), em que ela problematiza como serão distribuídas as atividades e, por conseguinte, as viagens da população frente ao fato de que a proporção de mulheres na força de trabalho vinha aumentando rapidamente¹⁹. À época havia quem afirmasse que conforme as desigualdades salariais diminuíssem e os papéis sociais se alterassem, as diferenças nos padrões de viagens sumiriam. Rosenbloom discorda e, entre seus argumentos figura o de que as variáveis sócio-econômicas tradicionais pouco explicam as relações de poder e os processos de decisão circunscritos ao ambiente doméstico. Outrossim, identifica-se que nas diversas classes socio-econômicas as mulheres que trabalham ainda continuam sendo as principais responsáveis pelas tarefas domésticas e pelo cuidado com as crianças. Um fato retratado pela autora ilustra o desconhecimento completo do fenômeno por parte dos órgãos oficiais de planejamento de transportes: cientistas contratados pelo U. S. Department of Transportation chegaram a declarar que estimular os homens a usar o transporte público e deixar o carro em casa poderia aumentar o consumo de energia e os níveis de poluição, pois as mulheres usariam o carro fazendo viagens mais curtas, em baixas velocidades e com maior consumo de combustível.

Assim como foi para Willard, o ganho de mobilidade pode refletir melhores condições de vida para as mulheres. Não se pode esquecer que na maioria das vezes a viagem é atividade meio e não fim, ou seja, as pessoas precisam de um motivo para fazer uma viagem, querem chegar a algum lugar por um propósito. Caso uma mulher saísse de casa sem propósito, por exemplo no Brasil da década de 1970, seria questionada moralmente. Um homem também poderia ser considerado vagabundo na mesma situação - embora, há de se frisar, o julgamento moral sobre ele seria mais condescendente do que sobre ela. D’Incao (2012, p.235) relata que:

cronistas, viajantes e historiadores [...] exibem um quadro em que a

¹⁹ Em 1978, 54% das mulheres casadas dos Estados Unidos eram assalariadas, mais do que o dobro do que se constatou logo após o fim da Segunda Guerra Mundial (ROSENBLOOM, 1978).

menina ou a mulher [burguesa] candidata ao casamento é extremamente bem cuidada, é trancafiada nas casas, etc.

Desta forma, uma das maneiras de adquirir liberdade de movimento foi poder ter motivos que não domésticos para sair de casa, o que não se deu - nem se dá - sem resistência, como o editorial de Rosenbloom já dava pistas. Hanson (1995a apud HANSON, 2010) constatara que as mulheres de Massachusetts, Estados Unidos, tinham dificuldade de encontrar trabalho no final dos anos 1980. Mais recentemente, Silvey e Elmhirst (2003) apontam que mulheres sequer são consideradas para certos postos de trabalho na Indonésia porque não se supõem que possam estar fora de casa após escurecer. Em algumas vilas indianas, Raju (2005) ao caracterizar um projeto que visa o empoderamento feminino constata que uma das mudanças mais significativas detectadas foi o fato das mulheres poderem sair sozinhas de casa, isto é, poderem existir *per si* no espaço público. O “ganhar a rua” feminino é fundamentalmente ligado ao acesso ao trabalho, ao aumento da participação feminina na população economicamente ativa (KÜNZLER, 1994 apud BEST; LANZENDORF, 2005) e ao crescimento da renda individual. Mandel (2004) mostrou que mulheres que têm mais liberdade para fazer viagens têm maior renda em Porto Novo, Benin. Mesmo ao considerar países com menor desigualdade²⁰, como a Noruega, observar-se-á que as mulheres casadas trabalham em localidades mais próximas das residências e têm menos poder de escolha geográfico do que seus maridos no que tange às oportunidades de trabalho (HJORTHOL, 2000).

Não obstante, é preciso ter cuidado e não fazer uma relação identitária automática entre **mobilidade** e **empoderamento**. Alterar padrões de mobilidade pode significar alterar relações de poder já que é de alguma forma requisito da acessibilidade à escola, ao trabalho, a hospitais, às lojas, às áreas de lazer, etc. Então, maior mobilidade pode significar mais equidade em algumas situações, mas não necessariamente em todas. Isto é, não há uma relação biunívoca em que maior mobilidade leve, sempre, a mais acesso e oportunidades iguais a todas pessoas. Essas nuances podem ser observadas quando se perfaz uma abordagem interseccional, como fizeram McLafferty e Preston (1991), McLafferty e Preston (1992) e Crane (2007). Gilbert (1998) faz essa ponderação e encara como demasiado simplista entender a mobilidade como empoderamento e a imobilidade como sinal de falta de poder; afinal, é preciso considerar a espacialização dessa mobilidade, considerando suas características sociais, culturais e econômicas. Talvez uma pessoa se desloque menos por ter mais acesso a oportunidades num raio próximo do seu lar e este lar assim seja localizado porque essa pessoa desfruta de melhor posição social e econômica.

²⁰ Para medir a desigualdade é comum utilizar o Índice de Gini, cujos valores são tão mais altos quanto maior for a desigualdade da renda familiar. Estados Unidos apresentam um Índice Gini de 34 (2005); Indonésia, de 39,4 (2005); Benin, de 36,5 (2003); Noruega, de 25 (2008); e Brasil, de 55,3 (2001). Fonte: <<https://www.cia.gov/library/publications/the-world-factbook/fields/2172.html>> Acesso em 29 de novembro de 2014.

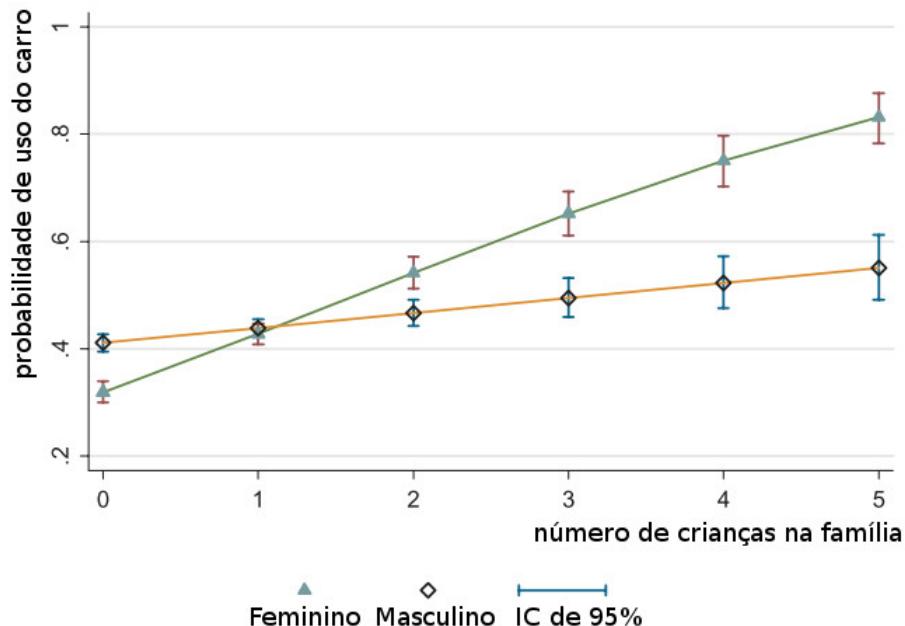
[McLafferty e Preston \(1991\)](#) indicam que maior mobilidade não significa mais poder, visto as condições socioeconômicas de quem precisa fazer longas viagens para trabalhar em postos de baixa remuneração. Em artigo de [1992](#), com base em dados referentes ao final dos anos 1980 do norte de Nova Jersey (Estados Unidos), [McLafferty e Preston](#) sustentam a hipótese que as diferenças de gênero na segmentação do mercado de trabalho têm consequências sobre a distribuição espacial (desigual) das minorias. As autoras conseguem atingir uma **abordagem interseccional** entre raça, gênero e classe neste trabalho. As mulheres afro-descendentes, latinas e brancas distribuem-se de forma diversa no espaço urbano. As negras contam com tempos de viagem maiores e menor grau de confinamento espacial que as latinas e as brancas. A “cidade da mulher negra” é maior, mas nem por isso, melhor: elas têm menos acesso a oportunidades de trabalho em regiões próximas das suas residências e geralmente são empregadas em postos de baixa remuneração. A maioria desses postos, ligados ao setor de serviços e não ao de processos de manufatura, são menos vulneráveis ao desemprego. Os homens negros e latinos e as mulheres latinas são contratados mais frequentemente pelo setor manufatureiro e, portanto, são mais suscetíveis ao desemprego estrutural. Essa menor vulnerabilidade ao desemprego da mulher negra, contudo, não implica que as oportunidades de emprego lhes sejam mais acessíveis - elas precisam ir mais longe para conseguir-las. Entre as mulheres, as latinas apresentam viagens relativamente curtas, próximo ao padrão das brancas. Porém, contam com alto grau de confinamento e a pior remuneração de todos os grupos analisados. Destarte, [McLafferty e Preston](#) mostram que analisar somente a classe (pela renda, poder de compra, posse de bens, grau de instrução associado), ou somente o gênero, ou somente a raça/etnia pode levar a conclusões muito triviais para fenômenos que são mais complexos. Em estudo mais recente, também nos Estados Unidos, que parte de bases de dados nacionais de 1985, 1995 e 2005, [Crane \(2007\)](#) observa que os tempos de viagens vêm convergindo quando consideradas as várias raças/etnias do mesmo gênero, em especial para mulheres. Ou seja, mulheres negras, brancas, asiáticas e latinas tendem a apresentar menos diferenças entre si ao longo do tempo. Analogamente, esse fenômeno também ocorre dentro do grupo masculino.

Embora a divisão de trabalho por gênero seja identificada como um fator que influencia a mobilidade, costuma-se ver o trabalho doméstico como uma restrição na participação do mercado de trabalho. Subestima-se o efeito do arranjo familiar no padrão de atividades e de viagens geradas a partir de demandas domésticas. Se no mercado de trabalho vem sendo traçado um caminho que tende a diminuir o desequilíbrio de gênero, no trabalho doméstico ainda é a mulher a grande responsável pela sua execução. Cabe então, analisar as **viagens cujo motivo não seja o trabalho** e outros aspectos relacionados ao arranjo familiar: como a presença de criança interfere na rotina familiar e como se distribui o uso do automóvel entre os membros, quando este está presente.

Em 1997, Root e Schintler (1999) indicam que cerca de 50% das viagens feitas por mulheres, nos Estados Unidos, por motivos pessoais (não trabalho) eram, na realidade, para a família. Vance e Iovanna (2007) trabalham econometricamente com dados em painel da Alemanha, referente ao período de 1996 a 2003, indagando se o gênero desempenha papel relevante na determinação da probabilidade de utilização do automóvel ou da distância percorrida; e se assim o for, se seria esse papel influenciado por outros atributos socieconômicos do indivíduo ou de seu núcleo familiar. Os autores constataram que as mulheres realizam mais viagens do que os homens quando o motivo não é trabalho. E embora o volume de viagens delas seja maior, a relação de dependência do carro para este tipo de viagem é menor - elas utilizam mais outros modos.

Vance e Iovanna (2007) verificaram que situação ocupacional (empregado(a) ou não), número de crianças na família (ver Figura 5), facilidade de acesso ao transporte público tiveram influência significativa nas viagens feitas por carro (que não para o trabalho), tanto para homens como para mulheres. Essas mesmas variáveis não influenciaram, porém, a distância média dirigida; apenas o acesso (ou não) ao automóvel incidiu sobre esse efeito.

Figura 5 – Simulação de probabilidade de uso do carro em função do número de crianças na família, na Alemanha



Fonte: Adaptado de (VANCE; IOVANNA, 2007, p.59)

A **presença de criança** é um fator que impacta bastante no padrão de atividades da família, e pode incidir diferentemente sobre pais e para mães. Ainda olhando para a Alemanha, o estudo de Best e Lanzendorf (2005) para a região de Cologne indica que mães usam menos frequentemente o carro do que mulheres sem filhos, ao passo que pais usam

mais o carro do que homens sem filhos. Este resultado contradiz o encontrado por [Vance e Iovanna](#), indicando que é preciso considerar a diversidade dos contextos socioeconômicos e culturais neste tipo de análise e que não é possível simplesmente transpor conclusões de outros locais para a RMSP.

[Goddard et al. \(2006\)](#) identificaram que a presença de criança na família causa diferença significativa no comportamento de viagens entre homens e mulheres no norte da Califórnia (Estados Unidos). A presença de criança na família tem grande impacto na quantidade de tempo/distância que a mulher dirige automóvel, mas tal efeito não se observa no comportamento masculino. Esse comportamento pode se dever ao fato de que os homens, antes da paternidade, já estão mais familiarizados com o uso do carro e tendem a não trocar sua escolha modal. As mulheres, mães, tendem a ser mais pressionadas pelas atividades a serem cumpridas, que não o trabalho. Elas passam a absorver com mais facilidade a viagem de servir passageiro (a criança). Nesta conjuntura vale ressaltar que as escolas, em especial que atendem as crianças nas primeiras idades, geralmente ficam próximas à residência o que estimula que as viagens para levar crianças à escola sejam feitas a pé, pelas mães. Porém, as metodologias utilizadas nos grandes *surveys*, tanto no Brasil como fora, pecam em detectar viagens curtas a pé. Nas Pesquisas OD da RMSP as viagens feitas integralmente a pé, se estiverem num raio inferior a 500 metros da residência, são desconsideradas se não forem com destino ao trabalho ou à escola.

[Fox](#) já apontava que em 1983 as mulheres nos Estados Unidos usavam menos o automóvel e mais o transporte público. Essa priorização masculina no **uso do automóvel** vem persistindo nos Estados Unidos e também na Europa. [Hjorthol \(2000\)](#) ao investigar mulheres e homens casados da região de Oslo, Noruega, observou que em famílias que dispunham de um carro, o marido detinha a prioridade do uso. [Polk \(2003\)](#) indica que os homens usam mais o carro, acumulam mais quilômetros percorridos por ano e fazem mais viagens como ocupantes únicos do que mulheres na Suécia. Embora ter à disposição um carro para uso privado é o fator que mais influencia o seu uso em viagens motivo “manutenção do lar”, segundo [Best e Lanzendorf \(2005\)](#), na Alemanha, isso ainda não é suficiente para que as mulheres usem mais o carro do que os homens no geral. Estes autores elencam ainda outros fatores que estimulam o uso do carro: renda familiar e participação no mercado de trabalho. Assim, se há mais renda familiar, há mais condições financeiras de adquirir um carro para tê-lo à disposição. E se há mais pessoas na família que fazem parte do mercado de trabalho, há mais interesse em comprar um carro, seja por aumento da renda familiar, seja pela necessidade de contar com um modo com flexibilidade de rota.

Portanto, as pesquisas apontam para o fato de que, *ceteris paribus*, quando existe a posse do automóvel na família, este fica mais frequentemente à disposição dos homens do que das mulheres. Dessa forma, as mulheres andam mais de transporte público e, ao andarem de automóvel: (i) são com maior frequência passageiras, remetendo à ideia de “não andar

desacompanhada fora de casa”, e (ii) servem passageiro mais frequentemente, remetendo às tarefas do cuidado (HIRATA; GUIMARÃES, 2012) com crianças e idosos (ROSENBLOOM, 2000; ROSENBLOOM, 2003). Se isso por um lado reflete menor autonomia e independência das mulheres por construção histórica, por outro lado, percebe-se também que elas conscientemente indicam estar mais dispostas a uma migração modal para obter um padrão mais sustentável de deslocamentos do que eles. Mulheres e homens demonstram atitudes diferentes em relação à proteção do meio ambiente e à sustentabilidade, apesar da taxa de motorização feminina vir crescendo nos países mais industrializados (ROOT; SCHINTLER, 1999). Polk (2003) se propõe a estudar se, na Suécia, as mulheres são potencialmente mais adaptáveis a um sistema de transportes mais **sustentável** do que os homens. A autora conclui que sim, as mulheres tendem a expressar mais preocupação em relação as questões ambientais e declararam maior vontade de reduzir o uso do carro.

Como foi possível depreender das seções anteriores deste capítulo, o carro é modo que mais ocupa espaço urbano e com maior consumo energético por pessoa; desta maneira, buscar um olhar de sustentabilidade sobre a análise de gênero e transportes é imprescindível. Schwanen, Dijst e Dieleman (2002) indicam que são variáveis importantes de análise do ponto de vista da sustentabilidade a **distância** viajada, o **modo** utilizado e o **tempo** de viagem - este último, embora às vezes não analisado, importa por ser um fator que pesa bastante nas decisões associadas à viagem (fazê-la ou não, que modo utilizar e que rota percorrer).

Fox indicava os padrões de viagens das mulheres nos Estados Unidos em 1983: elas faziam menos viagens, viagens mais curtas e rápidas. Quase dez anos depois, Johnston-Anumonwo (1992) revisita a hipótese de que mulheres fazem viagens mais curtas que homens em função de suas socialmente construídas atribuições domésticas. O autor foca-se no tipo de família e toma para essa análise a variável número de trabalhadores(as) da família. Assim, constitui-se o grupo das famílias em que há um(a) trabalhador(a) e daquelas com dois trabalhadores(as). Com dados de Baltimore, Estados Unidos, no grupo em que o arranjo familiar conta com mais de uma pessoa que trabalha, as diferenças entre distâncias de viagens de homens e mulheres tendem a ser maiores - ainda que sejam controlados outros fatores como, por exemplo, presença de criança na família. Nesse mesmo país, já no século XXI, Crane (2007) utiliza dados de 1985, 1995 e 2005 em suas análises e novamente afirma que mulheres fazem viagens mais curtas que homens. Crane vai além e contesta estudos que apontam que o *gender gap* dos tempos de viagem esteja diminuindo e das distâncias de viagem tenham até sumido em algumas áreas. Constatou que ainda persistem diferenças: as distâncias percorridas por homens e por mulheres convergem muito lentamente e os tempos divergem.

Na Europa, o panorama se mantém semelhante ao dos Estados Unidos. Fagnani (1983) estuda os padrões de deslocamento das mulheres em Paris e constata que lá também

elas desenvolvem viagens mais curtas. Schwanen, Dijst e Dieleman (2002), a partir do *National Travel Survey* holandês de 1998, analisam tempo, motivo e modo da viagem. Encontram evidências que fatores sociodemográficos (gênero, número de trabalhadores(as) na família, idade e grau de instrução) e contexto espacial da localização da residência influenciam o tempo médio de viagem diário. Os efeitos da posse de carro e da renda se dão de forma indireta, pois refletirão na escolha modal ou no número de trabalhadores(as) da família, segundo os autores. Do ponto de vista da interação com características urbanas, os tempos de viagem de carro tendem a aumentar quanto maior for o grau de urbanização e quanto mais policêntrica for uma região.

No conjunto Ásia e Oceania, o diagnóstico permanece. Ao analisar as viagens motivo trabalho da região metropolitana de Melbourne, Austrália, Howe e O'Connor (1982) expõem que as taxas de participação feminina na força de trabalho australiana eram baixas à época, levanta a hipótese de que isso se dê por alguma dificuldade de acesso às oportunidades de trabalho. Constata que mulheres fazem viagens mais curtas que homens e conclui que para incrementar a participação feminina no mercado de trabalho é preciso que ofertas de emprego estejam mais distribuídas espacialmente. Cusset (1997 apud VASCONCELLOS, 2001) constata que as mulheres em Hanoi (Vietnam) fazem a maior parte das viagens diárias a pé (54%), para os homens essa taxa cai para cerca de 39%, indicando maior motorização deles. Lee e McDonald (2003) também constata que para Seul, Coreia, tanto os tempos quanto as distâncias de viagens são menores para as trabalhadoras. Nessa cidade, trabalhadores(as) solteiros(as) contam com distâncias de viagem maiores que os(as) casados(as) e, merece destaque o detalhe de que trabalhadores cujas esposas trabalham têm viagens mais curtas do que aqueles cujas esposas não trabalham. Por fim, Lee e McDonald aponta que a responsabilidade por cuidar das crianças é um fator de encurtamento das viagens das mulheres casadas coreanas.

No Brasil, em específico na RMSP, Vasconcellos (2001) indica também que as mulheres fazem menos viagens e andam mais a pé do que os homens. Como os padrões de deslocamentos das mulheres são muito moldados pelas atividades ligadas ao ambiente doméstico, Strambi e Bilt (1998), com base na Pesquisa OD-1987, lançam olhar sobre o núcleo familiar e constatam que, para categorias de menor renda, o número médio de viagens cai conforme cresce o tamanho da família, o que pode indicar insuficiência de renda para arcar com as tarifas de transporte dos “membros adicionais”. Percebe-se que o estudo da mulher, da **família** e da mulher na família são relevantes, pois são fortes fatores sociais e culturais condicionantes de comportamentos. Vasconcellos (2001, p.119) endossa essa abordagem ao dizer que:

O papel das mulheres é especialmente relevante para entender os padrões diários de deslocamento nas famílias dos países em desenvolvimento. A melhor forma de entender estes padrões é começando pela divisão de tarefas domésticas e depois examinar condicionantes culturais e religiosos de sua mobilidade.

[McGuckin e Murakami \(1995\)](#), com base no *Nationwide Personal Transportation Survey* de 1995 estadunidense examinam o **encadeamento de viagens** feitas durante a semana de homens e mulheres em fase adulta. Constatam que mulheres fazem cadeias de viagens com mais segmentos para que possam acomodar suas responsabilidades domésticas. Essa segmentação é ainda mais sentida por elas quando se conta com a presença de criança na família. As autoras ainda apontam uma mudança na dinâmica doméstica em relação às tarefas e responsabilidades. Dez anos depois, [Mcguckin, Zmud e Nakamoto \(2005\)](#) publicaram estudo sobre o encadeamento de viagens com início na residência e término no trabalho de acordo com gênero e ciclo de vida a partir de base de dados nacionais dos Estados Unidos de 2001. Constataram que em lares com pai e mãe, em que ambos trabalham, as viagens cujo propósito de servir passageiro é deixar as crianças na escola têm maior probabilidade de serem acomodadas na cadeia de viagens da mulher do que do homem. Na Europa, situação semelhante é encontrada. Na Espanha e no Reino Unido, em 1997, grande parte das viagens femininas era para servir passageiro, em sua maioria, filhos(as) ([ROOT; SCHINTLER, 1999](#)). Em Oslo, mulheres casadas fazem mais encadeamento de viagens por conta de responsabilidades domésticas do que homens casados ([HJORTHOL, 2000](#)). Logo, se houve algum rearranjo na divisão de tarefas domésticas, essa mudança não foi profunda o suficiente para gerar alterações perceptíveis nos padrões de deslocamentos de homens e mulheres em diversos países.

Como o gênero configura e influencia a mobilidade, e como a mobilidade configura e influencia o gênero é o foco da discussão empreendida por [Hanson \(2010\)](#). Entretanto, o mais comum na pesquisa quando ocorre a abordagem de gênero e mobilidade é um campo de conhecimento olhar o outro, com pouco esforço de buscar literatura ou metodologia de intersecção entre as áreas e, com isso, parte-se de conceitos e hipóteses que conduzem a resultados pouco convergentes ao final. [Hanson \(2010, p.09\)](#) afirma ainda que em geral o debate ocorre em termos muito genéricos em torno da mobilidade/imobilidade com reflexos de uma ideologia dual que identificam a mulher/feminilidade com a casa, o espaço doméstico, movimentos restritos e o homem/masculinidade com a rua, o espaço público, movimentos livres. Enxergar o mundo a partir de um código binário de gênero foi uma construção longa e aboli-la por completo é uma tarefa a que este trabalho não se propõe, embora sempre que possível tente considerar a categoria como não binária.

A maior parte da literatura revisada diz respeito aos países do hemisfério Norte, mais industrializados, concentrando-se na Europa e nos Estados Unidos. Isso deve-se principalmente a dois fatores: (i) são países em que os temas gênero, transporte e sus-

tentabilidade são mais sistematicamente investigados; (ii) alguns países da Ásia também têm alguma produção dentro da temática desta pesquisa, mas muitas vezes em língua que não é de grande difusão no ocidente (como mandarim ou coreano). O que realmente chama a atenção é a parca literatura a respeito da América Latina. A maioria dos estudos encontrados baseiam-se em grandes bases de dados quantitativos que são tratados de forma agregada. Daí, conclui-se que, agregadamente, os comportamentos de homens e de mulheres são diferentes, em diversos países e continentes. Uma série de aspectos relacionados à viagem são analisados para fazer essa constatação: quantidade por pessoa, tempo, distância, encadeamento, modo e motivo. Outros aspectos, ligados às características de indivíduos, são frequentemente envolvidos nas análises: gênero, idade, estágio do ciclo de vida (individual e familiar), estado civil, presença de filhos na família, papel exercido dentro da família, situação ocupacional e renda.

3 Considerações Metodológicas

Esta pesquisa lida fundamentalmente com dados secundários, principalmente do Metrô-SP através de suas Pesquisas OD e, em menor grau, com dados socioeconômicos advindos de pesquisas PNAD¹ e censos do IBGE². Por dados secundários entendem-se aqueles que “já foram coletados para objetivos que não os do problema” (MALHOTRA, 2001, p.127). Trabalhar com dados secundários traz vantagens e desvantagens. Como vantagens tem-se o acesso relativamente fácil, o baixo custo de coleta e a rapidez de obtenção dos dados. Como desvantagem tem-se que o propósito da coleta dos dados difere daquele para que estão sendo utilizados aqui. Isto é, a formulação dos questionários e o desenho dos bancos de dados buscam responder a perguntas diferentes das propostas por esta dissertação. Essa desvantagem não inviabiliza o uso dos dados, mas é uma informação que deve estar em mente ao manipulá-los e analisá-los. Malhotra (2001, p.128) indica que os dados secundários podem auxiliar a:

1. identificar o problema
2. definir melhor o problema
3. desenvolver uma abordagem do problema
4. formular uma concepção de pesquisa adequada
5. responder a certas perguntas de pesquisa e testar algumas hipóteses
6. interpretar dados primários com mais critério

Nesta dissertação os bancos de dados e manuais de referência das respectivas Pequenas OD foram requisitados em 05 de maio de 2014 por meio de formulário *online* do e-SIC³ do governo do Estado de São Paulo e disponibilizados pelo Metrô-SP em mídia digital para retirada em dentro do prazo de 20 dias estabelecido pela lei de acesso à informação pública. Os dados provenientes de PNAD, censos ou outras pesquisas foram obtidos por meio de relatórios públicos, disponibilizados *online* e têm suas fontes indicadas ao longo do texto. Entende-se que os dados advindos das Pesquisas OD auxiliarão na tarefa de identificar se existem diferentes padrões de mobilidade de acordo com o gênero na RMSp, pois contêm informações sobre os deslocamentos de indivíduos, com representatividade em suas respectivas zonas e indicados os respectivos fatores de expansão nos bancos de dados. Para

¹ A PNAD é a Pesquisa Nacional por Amostra de Domicílios, realizada bianualmente pelo IBGE (referência no mês de setembro) com objetivo de investigar características socioeconômicas da população. Fonte:<http://www.previdencia.gov.br/arquivos/office/3_081014-105206-595.pdf> Acesso em 20 de novembro de 2014

² O censo demográfico é uma pesquisa realizada decenalmente pelo IBGE com objetivo de caracterizar sociodemograficamente a população brasileira. O primeiro foi feito em 1872 e o último data de 2010. Fonte:<<http://cod.ibge.gov.br/234lq>> Acesso em 20 de novembro de 2014

³ O e-SIC é o Sistema Eletrônico do Serviço de Informações ao Cidadão que possibilita que qualquer pessoa, física ou jurídica, encaminhe pedidos de acesso à informação, acompanhe o prazo e receba a resposta da solicitação realizada para órgãos e entidades governamentais. Fonte: <<http://www.sic.sp.gov.br/>> Acesso em 20 de abril de 2014

investigar a evolução e os motivos dessas diferenças têm-se como base as hipóteses advindas da revisão da literatura: ao gênero feminino é atribuído um determinado papel social e lhe é inputada a realização de uma maior diversidade de trabalhos (remunerados e não remunerados) que precisam ser acomodados no cotidiano. Os dados socieconômicos podem dar indícios daquelas hipóteses ligadas à influência das características socioeconômicas sobre as ações de indivíduos. Dados como idade e ocupação (se trabalha, se estuda, etc.) podem dar indícios para compreender a influência do ciclo de vida no comportamento dos indivíduos. Dados como situação na família (pessoa responsável, cônjuge, etc.) e presença de filhos podem auxiliar a compreender a dinâmica familiar e seu reflexo nos padrões de atividades das pessoas. Em tempo, dados como posse de automóveis, motocicletas, bicicletas e modos utilizados nas viagens, podem contribuir para compreender a relação que é estabelecida com o transporte público e o transporte privado, mais especificamente o carro.

Os dados secundários das Pesquisas OD foram analisados segundo alguns critérios, observados no Quadro 1. Os objetivos primários da coleta dos dados são apresentados na Seção 4.1. Em relação à natureza dos dados, para que fossem melhor utilizados nesta dissertação, foram feitas compatibilizações entre as zonas (geográficas) de análise e os diversos bancos de dados, processo mais detalhadamente explicado no Capítulo 5. Em relação à confiabilidade, os dados foram adquiridos diretamente do Metrô-SP, responsável pela coordenação da coleta e compilação dos dados, e que goza de boa reputação e pioneirismo no Brasil na realização de pesquisas dessa natureza. Em relação à atualidade dos dados, no Capítulo no Capítulo 4 esclarece-se a periodicidade e datas de referência das OD-1977, OD-1987, OD-1997 e OD-2007. Em relação às especificações e metodologia, no Capítulo 4 são descritos brevemente os métodos de coleta, amostragem, conceitos utilizados, além das decorrentes limitações.

Quadro 1 – Critérios para Avaliação de Dados Secundários

Critérios	Questões
Objetivo	Por que os dados foram coletados?
Natureza	Definição de variáveis chave; unidades de medição; categorias usadas e relações examinadas
Confiabilidade	Experiência; credibilidade; reputação e integridade da fonte
Atualidade	Prazo entre coleta e publicação; frequência das atualizações
Especificações e Metodologia	Método de coleta de dados; índice de respostas; qualidade dos dados; técnica de amostragem; tamanho da amostra; criação do questionário e trabalho de campo

Fonte: Adaptado de ([MALHOTRA, 2001](#), p.129)

» REESCREVER ESTE PARÁGRAFO

Como análises preliminares dos dados, são apresentadas algumas estatísticas feitas para os bancos de dados das quatro *cross-sections* (1977, 1987, 1997 e 2007). Essas estatísticas podem ser divididas em três grupos: o de características das viagens, o de famílias e o de indivíduos. O primeiro grupo de análises busca compreender como essa amostra se comporta em termo de viagens realizadas, em cada ano e diferencialmente entre os anos, olhando para tanto variáveis como ZZZZZZZZZZZZz duração das viagens e número de viagens realizadas. O segundo grupo de análises procura como são as características sócio-econômicas familiares, observando variáveis como XXXXXXXXXXx O terceiro grupo de análises busca compreender como é essa amostra, em cada ano e diferencialmente entre os anos, olhando variáveis como YYYYYYYYYYYY idade, situação na família, grau de instrução e renda.

» REESCREVER ESTE PARÁGRAFO

Dentro deste segundo grupo de análises foi de interesse buscar verificar se a variável explicativa sexo era relevante para explicar tanto a duração como o número de viagens. Para tanto foram feitas regressões lineares simples e seus resultados são apresentados na Seção 6.2. Também há interesse em verificar se existe alguma diferença estatisticamente significativa nos padrões de deslocamento entre os sexos, para cada *cross-section*. Isso foi feito tomando como hipótese nula que as médias de ambos sexos eram iguais, para as variáveis dependentes analisadas. Essa hipótese foi testada e os resultados também são apresentados na Seção 6.2.

» REESCREVER ESTE PARÁGRAFO

Por fim, esta pesquisa utiliza um banco de dados secundário em que a informação relativa a gênero nasce quase que exclusivamente da variável sexo, codificada binariamente - certamente uma limitação; muito embora seja o tipo de pergunta que o(a) entrevistador não faz diretamente, mas marca no papel a partir de julgamento visual. Isso significa que uma pessoa transsexual cuja performatividade seja feminina, provavelmente será identificada como mulher no questionário. As variáveis de análise (dependentes) geralmente feitas nas pesquisas desta temática giram em torno de tempo e distância viajados, modos utilizados, motivo das viagens e encadeamento das viagens (HANSON, 2010). Estas variáveis começaram a ser exploradas no presente texto e continuarão a ser desenvolvidas; porém, se nesta abordagem não houver elemento metodológico inovador que rompa com a leitura binária de gênero, que a utilidade do presente trabalho seja em função da sua existência - já que não se localizou trabalho dessa natureza feito para a RMSP com o alcance temporal aqui pretendido.

É possível fazer análises agregadas (por zonas) ou desagregadas (por famílias ou indivíduos). As análises desagregadas da demanda, considerando como unidade de

estudo família indivíduo, têm sido indicadas como vantajosas do ponto de vista conceitual (ORTÚZAR; WILLUMSEN, 1994). As análises desagregadas permitem, por exemplo, avaliar o comportamento da demanda e considerar modelos baseados em teorias de atividades humanas, que consideram condicionantes familiares e externos aos deslocamentos ao invés das viagens isoladamente (JONES, 1981). Um aspecto a destacar na modelagem desagregada é a diferenciação dada à utilização da família ou do indivíduo como unidade de observação.

Sendo necessário considerar variáveis e categorias relevantes, bem como as interações entre elas, Strambi e Bilt (1998) utilizaram técnicas de segmentação para identificar grupos razoavelmente homogêneos, viabilizando o estudo do comportamento desses grupos e também dos fatores que diferenciam os grupos. A análise das atividades humanas por segmentos da população possibilita identificar necessidades distintas de transporte e, a partir dos padrões de atividades e viagens, tais informações podem ser aplicadas na formulação ou revisão de modelos de projeção de demanda (MAHMASSANI, 1988).

Essa segmentação pode ser feita segundo dois caminhos (MAGIDSON, 1994):

- (i) modelos não baseados em critérios que, através de técnicas de análises multivariadas, agrupam os indivíduos em aglomerados (ou *clusters*) de características homogêneas e não são preditivos;
- (ii) modelos baseados em critérios que, pela combinação de variáveis independentes, também definem agrupamentos homogêneos (para uma variável-critério de segmentação) e podem ser utilizados para predição.

Foram feitas análises de conglomerados (*clusters*) tendo como entradas um conjunto de variáveis de viagem relativo à família e um outro, relativo às pessoas. Quais foram essas variáveis e como cada uma foi formulada é possível observar nos capítulos subsequentes. Com isto, os grupos serão formados segundo similaridade em seus comportamentos de viagem, ou seja, em seus padrões de deslocamento. A partir da definição dos *clusters*, investigou-se quais características familiares e individuais aproximavam os elementos dentro de cada grupo, mas sobretudo, quais eram importantes para diferenciação entre grupos.

Observou-se que, apesar da variável ANO não entrar nas clusterizações, ela foi muito relevante na formação dos grupos, indicando que o efeito do tempo pode ter grande peso nos padrões de deslocamentos. Numa análise dos padrões de viagem do sudeste de Michigan, entre 1965 e 1980, KOSTYNIUK e KITAMURA (1984) concluíram que o padrão de deslocamentos diários não é estável ao longo do tempo.

4 Pesquisa Origem Destino (OD)

4.1 Periodicidade e Objetivos da Pesquisa OD

A Pesquisa Origem Destino (Pesquisa OD) é realizada a cada dez anos pela Companhia do Metropolitano de São Paulo (Metrô-SP), a partir de 1967. Assim, até hoje foram realizadas cinco Pesquisas OD (1967, 1977, 1987, 1997 e 2007), das quais este trabalho abrangerá as quatro últimas, cobrindo uma janela temporal de 30 anos. O intervalo de dez anos foi considerado pelo Metrô-SP muito longo mediante as rápidas transformações no espaço urbano; assim, em 2002 e em 2012 foram feitas Pesquisas de Aferição, com menor amostragem e zonas mais agregadas. Cabe esclarecer que estas pesquisas de aferição não serão objeto de análise do presente estudo.

A Pesquisa OD nasceu com a missão de compor uma base de dados que servisse de suporte a decisões de planejamento de transporte urbano na Região Metropolitana de São Paulo, que hoje abarca 38 municípios, além de São Paulo. Atualmente, além de cumprir esse papel, também é ferramenta de suporte para o planejamento urbano de maneira mais sistemática, bem como para a formulação de políticas públicas segmentadas, nas áreas de educação, saúde e segurança pública, por exemplo ([METRÔ-SP, 2008](#)).

4.2 Descrição Sucinta

A Pesquisa OD é composta de duas partes complementares, a saber, a Pesquisa Domiciliar e a Pesquisa de Linha de Contorno. A Pesquisa Domiciliar tem como escopo as viagens internas à Região Metropolitana de São Paulo (RMSP); nela são escolhidos os domicílios por amostragem, cujo critério será melhor discutido adiante, em que todos habitantes respondem a um questionário estruturado referente às viagens feitas no dia útil anterior à pesquisa. Já a Pesquisa de Linha de Contorno monitora pontos de entrada e saída (limites) da RMSP a fim de captar as viagens com origem dentro da RMSP e destino fora, vice-versa, ou ainda viagens que a atravessam. O presente trabalho tem como foco as viagens feitas internamente à RMSP, portanto, as bases de dados consideradas serão apenas aquelas advindas das Pesquisas Domiciliares.

A Pesquisa OD considera a dimensão espacial dos deslocamentos considerando as zonas de origem e de destino. Tais zonas tiveram seus limites alterados e área de cobertura expandida desde 1967. Na Tabela 2 é possível observar quantos municípios da RMSP foram envolvidos em cada pesquisa e em quantas zonas eram divididos. A correspondência entre as diversas zonas é feita por uma unidade de compatibilização chamada Unidade de

Correspondência de Zona (UCOD), em relação às quais todas zonas têm referência. Para que seja possível realizar uma análise de evolução temporal conjugando dados de diversas OD é preciso organizar todas as informações de maneira coerente, assim, é apresentado no Anexo A as 67 UCOD com as respectivas zonas correspondentes para 1977, 1987, 1997 e 2007. Para tal consolidação ser feita, parte das informações foi recebida do Metrô-SP e parte foi fruto de compilação própria.

Tabela 2 – Características Amostrais das Pesquisas OD

Ano	Municípios da RMSP	Zona
1967	15	206
1977	27	243
1987	38	254
1997	39	389
2007	39	460

Fonte: Compilação a partir de ([METRÔ-SP, 1977](#); [METRÔ-SP, 1987](#); [METRÔ-SP, 1997](#); [METRÔ-SP, 2007](#))

4.3 Dados Coletados

A Pesquisa OD coleta dados referentes a domicílios, famílias, indivíduos e viagens, o que possibilita buscar relações entre características de deslocamentos e de indivíduos (e respectivas famílias e domicílios), e também características socioeconômicas. Em 1987, 1997 e 2007 a amostra de domicílios é do tipo estratificada por faixas de consumo de energia elétrica¹ - isso se dá por dois fatores: (i) as concessionárias possuem bases cadastrais de registro de domicílios mais confiáveis e representativas; (ii) “o consumo de energia elétrica tem correlação com a renda familiar, que por sua vez tem correlação com o número de viagens da família” ([METRÔ-SP, 2008](#), p.10). Esse esquema de amostragem estratificada buscou, em todos anos obter nível de confiança de 95%. Nas zonas em que não foi possível utilizar esse arranjo, foi feita amostra causal simples, com erros em torno de 7,5%. Na Tabela 3 é possível observar algum dados relativos às amostras.

Definido o tamanho da amostra total, define-se o tamanho de amostra para cada zona e, a partir daí, procede-se um sorteio de endereços por faixa de consumo energético -

¹ Na década de 1970 a companhia telefônica TELESP realizou um cadastro de domicílios, que foram categorizados segundo o padrão arquitetônico percebido externamente à residência. Nessa categorização eram considerados critérios como, por exemplo, se a construção é do tipo geminada ou não, etc. Os domicílios eram então classificados de 1 a 5, em que 1 significa o melhor parâmetro e 5 o pior (favela). O Metrô-SP utilizou esse cadastro da TELESP para fazer a estratificação da amostra de domicílios para a Pesquisa OD-1977.

etapa esta realizada pelas concessionárias, que fornecem ao Metrô-SP apenas os endereços dos domicílios selecionados, além de alguns adicionais para substituição caso necessário. Os selecionados recebem comunicação oficial por carta do Metrô-SP contendo as informações pertinentes à pesquisa. Quando no domicílio, os(as) pesquisadores(as) aplicam o questionário a todas pessoas que moram ali.

Tabela 3 – Características Gerais das Pesquisas OD

Ano	Domicílios	Pessoas entrevistadas do sexo feminino	Pessoas entrevistadas do sexo masculino
1977	26.132	55.868	52.163
1987	26.070	57.637	53.176
1997	23.841	51.454	47.326
2007	29.957	49.116	42.289
Total	106.000	214.075	194.954

Fonte: Compilação a partir de ([METRÔ-SP, 1977](#); [METRÔ-SP, 1987](#); [METRÔ-SP, 1997](#); [METRÔ-SP, 2007](#))

A coleta, consistência e digitação do dados são de responsabilidade de institutos de pesquisa contratados pelo Metrô-SP e que variaram ao longo do tempo. Após a consolidação primeira do banco de dados, é calculado e aplicado um fator de expansão aos resultados amostrais dos domicílios segundo a expressão (4.1). Depois determina-se, por consequência, um fator de correção referente às famílias e às pessoas. As viagens de quem usou o modo metrô são expandidas levando em consideração a entrada de passageiros no sistema Metrô-SP na data de referência da pesquisa. Situação análoga ocorre com o trem metropolitano. As viagens de quem usou outro modo que não metrô e/ou trem teve seu fator de expansão de viagens determinado pelo total de passageiros transportados pelo sistema de ônibus (em 2007 foram utilizados os dados provenientes de Bilhete Único da SPTrans).

$$\text{Fator de expansão de domicílio}_i = \frac{\text{Total de domicílios da zona}_i}{\text{domicílios da amostra da zona}_i} \quad (4.1)$$

Vale fazer algumas considerações acerca da renda familiar. Nem todas as pessoas respondem qual é a renda familiar, e como trata-se de uma das informações mais importantes para descrever o comportamento das pessoas ([SHEARMUR, 2006](#)).

Nos casos em que a renda não foi informado pelo(a) entrevistado(a), ela é atribuída, mas não sem critério. A atribuição da renda familiar baseou-se na pontuação estabelecida

por um critério nacional², que variou ao longo do tempo - tais informações podem ser vistas no Quadro 2.

Quadro 2 – Dados para atribuição de renda familiar

Ano	Mês de Referência	Classificação de Referência para Atribuição da Renda
1977	setembro	Função do Salário Mínimo
1987	setembro	Critério ABA/ABIMEPE (análogo ao Critério Brasil)
1997	outubro	Critério Brasil (ABIMEPE)
2007	outubro	Critério Brasil (ABEP)

Fonte: Compilação de informações obtidas por meio de correspondência eletrônica com Emilia Mayumi Hiroi, Coordenadora de Pesquisa e Avaliação de Transporte do Metrô-SP

Com isso, nos casos em que as pessoas não informaram a renda mas declararam os bens de consumo da família, a variável “renda familiar mensal” foi atribuída por meio das equações de regressão³ (4.2), (4.3) e (4.4), cuja função é estimar o poder de compra da família. Nesses critérios de classificação econômica existe a orientação de que a categoria automóvel não deve considerar táxis, vans, *pickups* usadas para fretes ou qualquer veículo usado para atividades profissionais, nem tampouco devem ser considerados veículos de uso misto (lazer e profissional) (ABEP, 2009). Essa mesma orientação em relação aos automóveis é feita pelos manuais das Pesquisas OD (METRÔ-SP, 1977; METRÔ-SP, 1987; METRÔ-SP, 1997; METRÔ-SP, 2007) tornando o conjunto coerente.

$$RFM_{87} = e^{(9,126 + 0,05051 * PONTUACAO_{ABA})} \quad (4.2)$$

$$RFM_{97} = e^{(5,672 + 0,03259 * PONTUACAO_{ABIMEPE})} \quad (4.3)$$

$$RFM_{07} = e^{(5,864 + 0,084 * PONTUACAO_{BRASIL})} \quad (4.4)$$

² Em 1987 foi usado o Critério ABA, em 1997 foi usado o critério ABIMEPE e, em 2007 foi utilizado o Critério Brasil, todos muitos semelhantes em metodologia que visa a classificação em categorias de capacidade de consumo segundo a posse de bens de consumo e do grau de instrução “do chefe da família” Fonte: <<http://www.abep.org/new/criterioBrasil.aspx>> Acesso em 17 de novembro de 2014.

³ As equações de regressão de 1987, 1997 e 2007 foram obtidas por meio de correspondência eletrônica com Emilia Mayumi Hiroi, Coordenadora de Pesquisa e Avaliação de Transporte do Metrô-SP.

Nas famílias em que não se obteve nem declaração da renda, nem informações suficientes sobre bens de consumo, a renda foi atribuída à família a partir da mediana da zona a que pertencia e com mesmo grau de instrução do(a) “chefe da família”.

4.4 Conceitos Adotados

A seguir, são replicados alguns conceitos utilizados pelo Metrô-SP no desenvolvimento das Pesquisas OD, a saber, *zona, família, respondente qualificado, modo coletivo, modo individual, modo não motorizado, modo motorizado, modo principal, viagem, viagem a pé*:

- (i) É considerada *família*: uma pessoa que more só, ou um conjunto de pessoas ligadas por laços de parentesco ou de dependência econômica que morem no mesmo domicílio; ou, ainda, conjunto de, no máximo, cinco pessoas que mesmo não tendo laço de parentesco morem num mesmo domicílio. O(a) empregado(a) doméstico(a) que more com algum outro parente na casa do patrão será considerada como outra família, mas caso o(a) empregado(a) more sozinho(a) na residência onde trabalha, será considerado(a) como parte da família do empregador.
- (ii) Compõem o *modo coletivo* o metrô, o trem, o ônibus, o microônibus, o transporte fretado, o transporte escolar, a lotação, a van, o trólebus.
- (iii) Compõem o *modo individual* o automóvel, o táxi, a motocicleta e a bicicleta.
- (iv) São considerados *modos não motorizados* os modos a pé e bicicleta.
- (v) São considerados *modos motorizados* os demais modos exceto a pé e bicicleta.
- (vi) *Modo principal* é o modo de maior hierarquia dentre os modos utilizados numa mesma viagem. Conforme estabelecido pelo Metrô-SP, a hierarquia desses modos é a seguinte, nesta ordem, do que predomina sobre qual: metrô, trem, ônibus, transporte fretado, transporte escolar, lotação, táxi, dirigindo automóvel, passageiro de automóvel, motocicleta, bicicleta, outros e a pé.
- (vii) *Respondente qualificado* é a pessoa com 10 anos ou mais, residente no domicílio sorteado e capaz de responder às perguntas feitas pelo pesquisador. Uma pessoa responsável pode fornecer informações referentes às pessoas menores de 10 anos; ou pessoas que não fossem capazes de responder ao questionário.
- (viii) *Viagem* é uma atividade secundária e refere-se ao deslocamento de uma pessoa, por motivo específico, entre dois pontos determinados (origem e destino), utilizando, para isso, um ou mais modos de transporte. Sendo nominado como origem o local onde a pessoa entrevistada se encontrava quando iniciou o seu deslocamento, e como destino o local para onde a pessoa entrevistada se dirigiu (destino final).
- (ix) *Viagem a pé* é aquela realizada integralmente a pé, da origem ao destino. Além disso, só será contabilizada como viagem a pé se a distância percorrida é superior a 500 metros (ou cinco quadras) ou se o motivo da viagem (na origem ou no destino) é trabalho ou

escola, independente da distância percorrida.

- (x) *Zona* de pesquisa é a unidade territorial de levantamento da origem e do destino das viagens

4.5 Limitações

Como este estudo baseia-se em dados secundários é preciso estar ciente das limitações que conceitos e metodologia de pesquisa adotados podem trazer. O conceito de família é bastante centrado na unidade do domicílio, o que pode desconsiderar laços afetivos e redes de solidariedade que as famílias ensejam, mesmo estando em domicílios separados. Por exemplo, uma criança pequena cujos pais precisam trabalhar, pode significar que vá haver viagens motivo escola, com um dos pais, mais provavelmente a mulher, servindo passageiro. Entretanto, a depender da oferta de serviços do local de residência, pode ser que não haja vaga em creche disponível. Pode ser ainda que a família não disponha de condições financeiras para pagar uma escola particular para essa criança. Um arranjo muitas vezes adotado é deixar a criança com avós ou tios que morem próximos. Isso representa impacto no padrão de mobilidade e também uma “economia” que o arranjo familiar proporciona. Esses arranjos e nuances pouco serão percebidos a partir destas bases de dados, pela forma com que foram construídas.

Outra limitação que merece atenção, é a hierarquia estabelecida entre os modos. Muito embora haja a descrição dos modos utilizados (até três em 1977 e 1987 e até quatro em 1997 e 2007), a duração da viagem disponível no banco de dados é a duração total, geralmente atribuída ao modo principal. Contudo, as viagens por modos não motorizados são as menos “fortes” na hierarquia de modos, sendo consideradas praticamente se forem exclusivas. Isso dificulta e às vezes impossibilita analisar devidamente os modos não motorizados dentro das cadeias de viagens. Ademais, existe uma subrepresentatividade das viagens a pé devido ao conceito adotado. E espera-se que estas viagens sejam importantes na descrição diferencial dos padrões de deslocamento de acordo com os gêneros. A mulher é responsável pela maior parte das tarefas ligadas à administração doméstica (ROOT; SCHINTLER, 1999; VANCE; IOVANNA, 2007), o que inclui compras rápidas e próximas à residência ou levar filhos(as) à escola (FOX, 1983; FAGNANI, 1983; JOHNSTON-ANUMONWO, 1992; MCGUCKIN; ZMUD; NAKAMOTO, 2005; SCHWANEN; DIJST; DIELEMAN, 2002; LEE; MCDONALD, 2003; CRANE, 2007), muitas vezes, a pé (VASCONCELLOS, 2001).

5 Tratamento dos Dados

5.1 Preparação da base de dados

Afim de tornar possível comparar dados das diversas Pesquisas OD objetos deste trabalho (1977, 1987, 1997 e 2007) foi necessário conhecer o banco de dados de cada edição, e para que seja possível analisar a evolução de padrões de mobilidade e de comportamento, é preciso que os dados dos diversos bancos sejam comparáveis. Para isso, foi desenvolvido o desenho de um Banco de Dados Unificado (BDU), com função de integrar e compatibilizar as informações julgadas relevantes até este momento, e também aquelas que acredita-se possam vir a ser úteis em etapas posteriores do trabalho. Os *layouts* dos bancos de dados originais podem ser observados no Anexo C, e o do banco integrador pode ser visto no Quadro 3 a seguir.

Inicialmente, nos bancos de dados originais, foram identificadas as variáveis correspondentes entre os diversos anos e o BDU. Para cada variável e para cada ano foram feitas transformações para que fosse possível gerar um único banco de dados final, possível de ser empilhado - essas transformações estão descritas no Capítulo 5. Devido ao tamanho dos bancos de dados, a preparação dos dados não foi feita em planilhas eletrônicas convencionais, mas por meio de códigos em linguagem *python*. A estrutura dos códigos apresentados no Anexo D é dividida em 5 blocos: (i) *Set up* inicial com chamadas de bibliotecas e configurações; (ii) Definição de *loggers*; (iii) Definição de funções gerais; (iv) Definição da função principal; e (v) Chamada para execução da função principal.

- (i) *Set up*: foram utilizadas as bibliotecas *math*, para as funções matemáticas, e *pandas*, as análises de dados.
- (ii) Definição de *loggers*¹: foram estabelecidos dois *loggers*, o *log_output* que salva o conteúdo num arquivo de saída (de texto, com extensão .log) e o *log_tela*, que além de salvar o conteúdo no arquivo de saída também mostra esse conteúdo na tela.
- (iii) Definição de funções gerais: são definidas 5 funções gerais assessórias que serão utilizadas pelas funções gerais “passo”. Ou seja, as 66 funções gerais “passo” são que, de fato, fazem o trabalho de transformação variável a variável. As assessórias servem para que estas possam ler os arquivos .csv e realizar testes de consistência, entre outras tarefas.

¹ Logger é uma rotina utilizada para acompanhar o processamento dos dados.

São funções gerais assessórias:

- consulta_refext: traz valor de referência externa (em arquivo csv) baseado em valor de referência do arquivo de origem;
- verifica_dummy: verifica se uma variável, do tipo *dummy*, contém algum valor diferente de 0 ou de 1;
- verifica_range: verifica se uma variável, do tipo número inteiro, contém algum valor fora de um intervalo especificado.
- corrige_renda: corrige a renda indicada aplicando um deflator que é passado como parâmetro; e
- coord: preenche as colunas de coordenadas “CO_DOM_X”, “CO_DOM_Y”, “CO_ESC_X”, “CO_ESC_Y”, “CO_TRAB1_X”, “CO_TRAB1_Y”, “CO_TRAB2_X”; “CO_TRAB2_Y”, “CO_ORIG_X”, “CO_ORIG_Y”, “CO_DEST_X” e “CO_DEST_Y”, segundo consulta ao arquivo externo com correspondência entre subzonas e suas coordenadas x e y.

São funções gerais “passo”:

- passo_ano: preenche a coluna “ANO” segundo as categorias do Quadro 3;
- passo_dia_sem: preenche a coluna “DIA_SEM” segundo as categorias do Quadro 3;
- passo_ucod: preenche a coluna “UCOD” segundo consulta ao arquivo externo com correspondência entre zonas e ucods;
- passo_zona_dom, passo_subzona_dom e passo_mun_dom: checa se existe algum erro no intervalo pertinente, respectivamente, às zonas, subzonas e municípios do domicílio de cada ano conforme Quadro 3;
- passo_f_dom: checa se existe algum erro (número diferente de 0 ou 1) na coluna “F_DOM”;
- passo_tipo_dom: checa se existe algum erro no intervalo pertinente aos tipos de domicílio conforme Quadro 3;
- passo_f_fam: checa se existe algum erro (número diferente de 0 ou 1) na coluna “F_FAM”;
- passo_cond_mora: faz transformações nas categorias da variável “COND_MORA” e checa se existe algum erro no intervalo pertinente à condição de moradia conforme Quadro 3;
- passo_ren_fam: corrige a renda familiar segundo o parâmetro deflator passado pela função principal e armazena os valores na coluna “REN_FAM”;
- passo_cd_renfam: faz transformações nas categorias da variável “CD_RENFAM” e checa se existe algum erro no intervalo pertinente à renda individual conforme Quadro 3;
- passo_f_pess: checa se existe algum erro (número diferente de 0 ou 1) em “F_PESS”;
- passo_sit_fam: faz transformações nas categorias da variável “SIT_FAM” e checa se existe algum erro no intervalo pertinente à situação familiar conforme Quadro 3;

- passo_sexo: faz transformações nas categorias da variável “SEXO” de forma que seja uma *dummy* que indica se pessoa é mulher ou não e checa se existe algum erro (número diferente de 0 ou 1);
- passo_grau_instr: faz transformações nas categorias da variável “GRAU_INSTR” e checa se existe algum erro no intervalo pertinente ao grau de instrução conforme Quadro 3;
- passo_estuda: atribui *dummy* que indica se pessoa estuda ou não, ou seja, se zona da escola for zero (0) então, pessoa não estuda (0), caso contrário, a pessoa estuda (1); além de checar se existe algum erro (número diferente de 0 ou 1);
- passo_ocup: faz transformações nas categorias da variável “OCUP” e checa se existe algum erro no intervalo pertinente à condição de ocupação da pessoa conforme Quadro 3;
- passo_setor_ativ: faz transformações nas categorias da variável “SETOR_ATIV” e checa se existe algum erro no intervalo pertinente à condição de ocupação da pessoa conforme Quadro 3;
- passo_ren_ind: corrige a renda familiar segundo o parâmetro deflator passado pela função principal e armazena os valores na coluna “REN_IND”;
- passo_cd_renid: faz transformações nas categorias da variável “CD_RENIND” de forma que seja uma *dummy* que indica se pessoa tem renda ou não e checa se existe algum erro (número diferente de 0 ou 1);
- passo_f_viag: checa se existe algum erro (número diferente de 0 ou 1) na coluna “F_VIAG”;
- passo_fe_viag: não se mexe na coluna “FE_VIAG”;
- passo_zona_esc, passo_subzona_esc e passo_mun_esc: checa se existe algum erro no intervalo pertinente, respectivamente, às zonas, subzonas e municípios da escola de cada ano conforme Quadro 3;
- passo_zona_trab1, passo_subzona_trab1 e passo_mun_trab1: checa se existe algum erro no intervalo pertinente, respectivamente, às zonas, subzonas e municípios do trabalho 1 de cada ano conforme Quadro 3;
- passo_zona_trab2, passo_subzona_trab2 e passo_mun_trab2: checa se existe algum erro no intervalo pertinente, respectivamente, às zonas, subzonas e municípios do trabalho 2 de cada ano conforme Quadro 3;
- passo_zona_orig, passo_subzona_orig e passo_mun_orig: checa se existe algum erro no intervalo pertinente, respectivamente, às zonas, subzonas e municípios da origem da viagem de cada ano conforme Quadro 3;
- passo_zona_dest, passo_subzona_dest e passo_mun_dest: checa se existe algum erro no intervalo pertinente, respectivamente, às zonas, subzonas e municípios do destino da viagem de cada ano conforme Quadro 3;
- passo_serv_pas_orig e passo_serv_pas_dest: atribui *dummy* que indica se pessoa serve passageiro ou não transformando as categorias das variáveis “SERV_PAS_ORIG”

e “SERV_PAS_DEST”, respectivamente, e checa se existe algum erro no intervalo pertinente ao ato de servir passageiro conforme Quadro 3;

- passo_motivo_orig e passo_motivo_dest: faz transformações nas categorias das variáveis “MOTIVO_ORIG” e “MOTIVO_DEST”, respectivamente, e checa se existe algum erro no intervalo pertinente ao motivo conforme Quadro 3;

- passo_modo1, passo_modo2, passo_modo3 e passo_modo4: faz transformações nas categorias das variáveis “MODO1”, “MODO2”, “MODO3” e “MODO4”, respectivamente, e checa se existe algum erro no intervalo pertinente aos modos de transporte usados segundo o Quadro 3;

- passo_modo_prin: faz transformações nas categorias da variável “MODO_PRIN” e checa se existe algum erro no intervalo pertinente ao principal modo de transporte usado segundo o Quadro 3;

- passo_tipo_est_auto: faz transformações nas categorias da variável “TIPO_EST_AUTO” e checa se existe algum erro no intervalo pertinente ao principal modo de transporte usado segundo o Quadro 3;

- passo_valor_est_auto: corrige o valor do estacionamento segundo o parâmetro deflator passado pela função principal e armazena os valores na coluna “VALOR_EST_AUTO”, caso essa variável não existe, não se mexe na coluna e os valores serão tomados como NA;

- passo_no_dom: gera o número do domicílio sendo que para cada “ZONA_DOM” o “NO_DOM” será atualizado sempre que “F_DOM” for igual a 1; caso contrário, se “F_DOM” for igual a zero, então “NO_DOM” será igual ao “NO_DOM” da linha anterior;

- passo_no_fam: gera o número da família sendo que para cada “ID_DOM” o “NO_FAM” será incrementado sempre que “F_FAM” for igual a 1; caso contrário, se “F_FAM” for igual a 0, então o “NO_FAM” será igual ao “NO_FAM” da linha anterior;

- passo_no_pess: gera o número da pessoa sendo que para cada “ID_FAM” o “NO_PESS” será atualizado sempre que “F_PESS” for igual a 1; caso contrário, se “F_PESS” for igual a zero, então “NO_PESS” será igual ao “NO_PESS” da linha anterior;

- passo_no_viag: gera o número da viagem sendo que para cada “ID_PESS” o “NO_VIAG” começa com 1 e será incrementado sempre que “F_VIAG” for igual a 1; caso contrário, se “F_VIAG” for igual a 0, então o “NO_VIAG” será igual ao “NO_VIAG” da linha anterior. Para garantir a representatividade do caso em que as pessoas responderam o questionário e não realizaram viagem, quando o “FE_VIAG” for não nulo e “ZONA_ORIG” e “ZONA_DEST” forem ambas nulas simultaneamente o “NO_VIAG” será igual a 0;

- passo_id_dom: gera o “ID_DOM” que é composto por 8 dígitos, em que o primeiro é o ano conforme Quadro 3, do segundo ao quarto dígitos tem-se a zona, e do quinto ao oitavo dígitos tem-se o número do domicílio;

- passo_id_fam: gera o “ID_FAM” que é composto por 10 dígitos, em que os 8 primeiros

correspondem ao “ID_DOM” e os dois últimos ao “NO_FAM”;

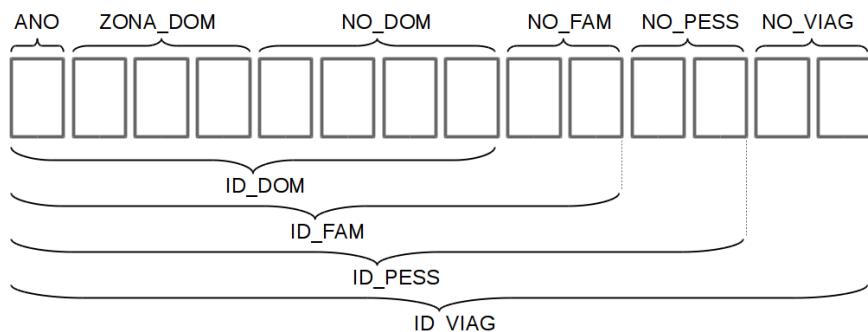
- passo_id_pess: gera o “ID_PESS” que é composto por 12 dígitos, em que os 10 primeiros correspondem ao “ID_FAM” e os dois últimos ao “NO_PESS”;

- passo_id_viag: gera o “ID_VIAG” que é composto por 14 dígitos, em que os 12 primeiros correspondem ao “ID_PESS” e os dois últimos ao “NO_VIAG”;

- passo_tot_viag: calcula e confere o campo “TOT_VIAG” baseado no máximo valor de “NO_VIAG” para cada pessoa;

- passo_cd_entre: substitui-se os valores da coluna “CD_ENTRE” segundo os valores de “TOT_VIAG”, considerando que todas entrevistas são consideradas “completas”, segundo informação do Metrô.

Figura 6 – Esquema indicativo da formação dos IDs



(iv) Definição da função principal: principia lendo o banco de dados original (arquivo tipo OD__.csv) e também outros arquivos auxiliares de dados estruturados, como o ucod__.csv, o setor_ativ__.csv e o coord_subzonas__.csv – sendo um arquivo desses para cada OD analisada. Logo, temos 4 arquivos .csv principais de entrada e 12 auxiliares. O arquivo ucod__.csv contém a relação de cada zona e sua respectiva UCOD, o arquivo setor_ativ__.csv contém a relação de código do setor de atividade do banco original e o respectivo código no banco unificado. O arquivo coord_subzonas__.csv contém as coordenadas dos centroides das subzonas extraídos a partir dos arquivos de mapas e que não constavam originalmente do banco de dados. No caso de 1977, não se conseguiu o arquivo do MapInfo com a granularidade das subzonas, portanto, foram utilizadas as coordenadas dos centroides das zonas. Na sequência, As colunas que não existiam originalmente no banco de dados são geradas e reordenadas. Existe uma variável de controle (“passo”) que a cada passo executado incrementa 1. São chamadas as funções gerais. Vale notar que ficaram para o final aquelas relacionadas à geração dos IDs, variáveis fundamentais para indexação correta do BDU. Assim, primeiro deve-se executar a função NO_DOM (número do domicílio) e depois a ID_DOM (identificador de domicílio). Analogamente isso ocorre com NO_FAM e ID_FAM (sobre as famílias), NO_PESS e ID_PESS (sobre as pessoas) e NO_VIAG e ID_VIAG (sobre as viagens). Não havia

expectativa inicial de que seria necessário gerar essa indexação, porém, a falta de um manual que explicasse a composição dos IDs presentes nos bancos de dados originais inviabilizou sua utilização. A vantagem de gerar nova indexação (ver Figura 6) é que se tem maior controle do seu significado, possibilitando ainda incorporar informações como o ANO, informação não considerada nas bases originais. Por fim, são chamadas as funções gerais referentes à quantidade e à distância da viagem (TOT_VIAG e DIST_VIAG).

(v) Chamada da função principal: é a parte mais enxuta em termos de código, pois trata apenas da chamada da função principal (*main*), que por sua vez chama as funções gerais.

Foi feito um **teste de consistência** que consistiu em identificar, para um mesmo indivíduo, que realizou viagem, se a zona de destino da viagem i é igual à zona de origem da viagem $i+1$. Inicialmente, foram encontradas 548 observações em 1977² que não passaram nesse crivo. Foi feita análise mais minuciosa desses considerando as variáveis de idade, situação familiar, renda individual, horário de saída e horário de chegada. Observou-se que eram casos de linhas i invertidas com as subsequentes, portanto, foram feitas as inversões de linhas e corrigidos 546 registros. Os 2 registros remanescentes foram corrigidos manualmente. Todos os registros de 1987 passaram no teste de consistência. Em 1997, 7 registros não passaram no teste de consistência e, após análise mais cuidadosa, parecia tratar-se de linhas duplicadas, logo, esses 7 registros foram excluídos. Situação semelhante ocorreu com 1 registro de 2007, que também foi excluído.

O Quadro 3 apresenta as variáveis do BDU, suas descrições e também tipificação, onde “Q” indica tratar-se de variável qualitativa, “M”, de variável métrica (quantitativa), “D”, de variável *dummy*, e “ID” variável de indexação (natureza texto).

² É preciso considerar que a base de dados 1977 é bastante antiga, rodada originalmente em computadores de grande porte numa época em que não havia computadores pessoais, e que passou por muitas migrações e manipulações até hoje.

Quadro 3 – *Layout do Banco de Dados Unificado*

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
Q	ANO	Ano de referência da Pesquisa OD	01	1 - OD-1977 2 - OD-1987 3 - OD-1997 4 - OD-2007
D	CD_ENTRE	Código de entrevista	01	0 - Completa sem viagem ou incompleta 1 - Completa com viagem
Q	DIA_SEM	Dia da Semana	01	2 - Segunda-Feira 3 - Terça-Feira 4 - Quarta-Feira 5 - Quinta-Feira 6 - Sexta-Feira
Q	UCOD_DOM	Unidade de Correspondência de Pesquisas OD para domicílio	02	1 a 67
Q	ZONA_DOM	Zona do domicílio da OD original	03	1 a 243 em 1977 1 a 254 em 1987 1 a 389 em 1997 1 a 460 em 2007
Q	SUBZONA_DOM	Subzona do domicílio da OD original	03	1 a 633 em 1977 1 a 9 em 1987 1 a 9 em 1997 não consta em 2007

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
Q	MUN_DOM	Município do domicílio	02	1 a 27 em 1977 1 a 38 em 1987 1 a 39 em 1997 1 a 39 em 2007
M	CO_DOM_X	Coordenada X do domicílio	12	12 dígitos, 2 casas decimais
M	CO_DOM_Y	Coordenada Y do domicílio	12	12 dígitos, 2 casas decimais
ID	ID_DOM	Identifica o domicílio	08	Composição de ID com ano, zona e domicílio
D	F_DOM	Identifica o primeiro registro do domicílio	01	0 - Demais registros 1 - Primeiro registro
M	FE_DOM	Fator de expansão do domicílio	10	10 dígitos, 2 casas decimais
M	NO_DOM	Número do domicílio	04	04 dígitos, número inteiro
D	TIPO_DOM	Tipo do domicílio	01	0 - coletivo 1 - particular
M	TOT_FAM	Total de famílias no domicílio	02	02 dígitos, número inteiro
ID	ID_FAM	Identifica a família	10	Composição de ID com ano, zona, domicílio e família
D	F_FAM	Identifica primeiro registro da família	01	0 - Demais registros 1 - Primeiro registro
M	FE_FAM	Fator de expansão da família	10	10 dígitos, 2 casas decimais
M	NO_FAM	Número da família	02	02 dígitos, número inteiro
Q	COND_MORA	Condição de moradia	01	1 - alugada 2 - própria 3 - outros
M	QT_AUTO	Quantidade de automóveis	01	01 dígito, número inteiro
M	QT_BICI	Quantidade de bicicletas	01	01 dígito, número inteiro
M	QT_MOTO	Quantidade de motocicletas	01	01 dígito, número inteiro
Q	CD_RENFAM	Código de renda familiar	01	0 - Renda declarada como zero 1 - Renda declarada maior do que zero 2 - Renda atribuída
M	REN_FAM	Renda familiar	08	08 dígitos, 2 casas decimais (R\$/mês, ref. out/2007)

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
ID	ID_PESS	Identifica a pessoa	12	Composição de ID com ano, zona, domicílio, família e pessoa
D	F_PESS	Identifica o primeiro registro da pessoa	01	0 - Demais registros 1 - Primeiro registro
M	FE_PESS	Fator de expansão da pessoa	10	10 dígitos, 2 casas decimais
M	NO_PESS	Número da pessoa	2	02 dígitos, número inteiro
Q	SIT_FAM	Situação familiar	01	1 - Pessoa responsável 2 - Cônjuge/Companheiro(a) 3 - Filho(a)/Enteado(a) 4 - Outro parente / agregado 5 - Empregado residente 6 - Outros (visitante não residente / parente do empregado)
M	IDADE	Idade	02	02 dígitos, número inteiro
D	SEXO	Sexo	01	0 - Masculino 1 - Feminino
D	ESTUDA	A pessoa estuda atualmente?	01	0 - Não 1 - Sim
Q	GRAU_INSTR	Grau de instrução da pessoa	01	1 - Não alfabetizado / Fundamental incompleto 2 - Fundamental completo / Médio incompleto 3 - Médio completo / Superior incompleto 4 - Superior completo
Q	OCUP	Condição de ocupação da pessoa	01	1 - Tem trabalho 2 - Em licença 3 - Aposentado(a) / Pensionista 4 - Desempregado(a) 5 - Sem ocupação 6 - Dono(a) de casa 7 - Estudante

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
Q	SETOR_ATIV	Setor de atividade (do 1º trabalho)	01	1 - Agrícola 2 - Construção Civil 3 - Indústria 4 - Comércio 5 - Administração Pública 6 - Serviços de Transporte 7 - Outros Serviços 8 - Outros 9 - Não se aplica
D	CD_RENIND	Condição de Renda Individual	01	0 - Não tem renda 1 - Tem renda
M	REN_IND	Valor da Renda Individual	08	08 dígitos, 2 casas decimais (R\$/mês, ref. out/2007)
Q	UCOD_ESC	Unidade de Correspondência de Pesquisas OD para escola	02	1 a 67
Q	ZONA_ESC	Zona da escola da OD original	03	1 a 243 em 1977 1 a 254 em 1987 1 a 389 em 1997 1 a 460 em 2007
Q	SUBZONA_ESC	Subzona da escola da OD original	03	1 a 633 em 1977 1 a 9 em 1987 1 a 9 em 1997 não consta em 2007
Q	MUN_ESC	Município da escola	02	1 a 27 em 1977 1 a 39 em 1987 1 a 39 em 1997 1 a 39 em 2007
M	CO_ESC_X	Coordenada X da escola	12	12 dígitos, 2 casas decimais
M	CO_ESC_Y	Coordenada Y da escola	12	12 dígitos, 2 casas decimais
Q	UCOD_TRAB1	Unidade de Correspondência de Pesquisas OD para trabalho 1	02	1 a 67

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtd. Díg.	Códigos, Categorias e Faixas de Valores Válidas
Q	ZONA_TRAB1	Zona do trabalho 1 da OD original	03	1 a 243 em 1977 1 a 254 em 1987 1 a 389 em 1997 1 a 460 em 2007
Q	SUBZONA_TRAB1	Subzona do trabalho 1 da OD original	03	1 a 633 em 1977 1 a 9 em 1987 1 a 9 em 1997 não consta em 2007
Q	MUN_TRAB1	Município do trabalho 1	02	1 a 27 em 1977 1 a 39 em 1987 1 a 39 em 1997 1 a 39 em 2007
M	CO_TRAB1_X	Coordenada X do trabalho 1	12	12 dígitos, 2 casas decimais
M	CO_TRAB1_Y	Coordenada Y do trabalho 1	12	12 dígitos, 2 casas decimais
Q	UCOD_TRAB2	Unidade de Correspondência de Pesquisas OD para trabalho 2	02	1 a 67
Q	ZONA_TRAB2	Zona do trabalho 2 da OD original	03	1 a 243 em 1977 1 a 254 em 1987 1 a 389 em 1997 1 a 460 em 2007
Q	SUBZONA_TRAB2	Subzona do trabalho 2 da OD original	03	1 a 633 em 1977 1 a 9 em 1987 1 a 9 em 1997 não consta em 2007
Q	MUN_TRAB2	Município do trabalho 2	02	1 a 27 em 1977 1 a 39 em 1987 1 a 39 em 1997 1 a 39 em 2007
M	CO_TRAB2_X	Coordenada X do trabalho 2	12	12 dígitos, 2 casas decimais
M	CO_TRAB2_Y	Coordenada Y do trabalho 2	12	12 dígitos, 2 casas decimais

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
ID	ID_VIAG	Identifica a viagem	14	Composição de ID com ano, zona, domicílio, família, pessoa e viagem
D	F_VIAG	Identificador de viagem	01	0 - Demais registros 1 - Primeiro registro
M	FE_VIAG	Fator de expansão da viagem	10	10 dígitos, 2 casas decimais
M	NO_VIAG	Número da viagem	02	02 dígitos, número inteiro
M	TOT_VIAG	Total de viagens da pessoa	02	02 dígitos, número inteiro
Q	UCOD_ORIG	Unidade de Correspondência de Pesquisas OD para origem	02	1 a 67
Q	ZONA_ORIG	Zona de origem da viagem da OD original	03	1 a 243 em 1977 1 a 254 em 1987 1 a 389 em 1997 1 a 460 em 2007
Q	SUBZONA_ORIG	Subzona de origem da viagem da OD original	03	1 a 633 em 1977 1 a 9 em 1987 1 a 9 em 1997 não consta em 2007
Q	MUN_ORIG	Município de origem da viagem	02	1 a 27 em 1977 1 a 39 em 1987 1 a 39 em 1997 1 a 39 em 2007
M	CO_ORIG_X	Coordenada X da origem	12	12 dígitos, 2 casas decimais
M	CO_ORIG_Y	Coordenada Y da origem	12	12 dígitos, 2 casas decimais
Q	UCOD_DEST	Unidade de Correspondência de Pesquisas OD para destino	02	1 a 67
Q	ZONA_DEST	Zona de destino da viagem da OD original	03	1 a 243 em 1977 1 a 254 em 1987 1 a 389 em 1997 1 a 460 em 2007
Q	SUBZONA_DEST	Subzona de destino da viagem da OD original	03	1 a 633 em 1977 1 a 9 em 1987 1 a 9 em 1997 não consta em 2007

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
Q	MUN_DEST	Município de destino da viagem	02	1 a 27 em 1977 1 a 39 em 1987 1 a 39 em 1997 1 a 39 em 2007
M	CO_DEST_X	Coordenada X do destino	12	12 dígitos, 2 casas decimais
M	CO_DEST_Y	Coordenada Y do destino	12	12 dígitos, 2 casas decimais
M	DIST_VIAG	Distância da viagem (m)	08	08 dígitos, 2 casas decimais
D	SERV_PAS_ORIG	Serve passageiro na origem?	01	0 - Não 1 - Sim
D	SERV_PAS_DEST	Serve passageiro no destino?	01	0 - Não 1 - Sim
Q	MOTIVO_ORIG	Motivo na origem	01	1 - Trabalho (indústria) 2 - Trabalho (comércio) 3 - Trabalho (serviços) 4 - Educação 5 - Compras 6 - Saúde 7 - Lazer 8 - Residência 9 - Outros
Q	MOTIVO_DEST	Motivo no destino	02	idem Motivo na origem
Q	MODO1	Modo 1	2	1 - Ônibus de linha 2 - Ônibus escolar / empresa 3 - Dirigindo automóvel 4 - Passageiro de automóvel 5 - Táxi 6 - Lotação / van 7 - Metrô 8 - Trem 9 - Motocicleta 10 - Bicicleta 11 - A pé 12 - Outros

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

Tipo	Variável	Descrição	Qtde. Díg.	Códigos, Categorias e Faixas de Valores Válidas
Q	MODO2	Modo 2	02	idem Modo 1
Q	MODO3	Modo 3	02	idem Modo 1
Q	MODO4	Modo 4	02	idem Modo 1
Q	MODO_PRIN	Modo Principal	02	idem Modo 1
Q	TIPO_VIAG	Tipo de viagem	01	1 - Coletivo 2 - Individual 3 - A pé
M	H_SAIDA	Hora de saída	02	Hora de saída
M	MIN_SAIDA	Minuto de saída	02	Minutos de saída
M	ANDA_ORIG	Tempo andando na origem	02	Tempo andando na origem (minutos)
M	H_CHEG	Hora de chegada	02	Hora de chegada
M	MIN_CHEG	Minuto de chegada	02	Minutos de chegada
M	ANDA_DEST	Tempo andando no destino	02	Tempo andando no destino (minutos)
M	DURACAO	Duração da viagem	02	Duração da viagem (minutos)
Q	TIPO_EST_AUTO	Tipo de estacionamento	01	0 - não estacionou 1 - estacionou em local privado (particular avulso ou mensal, próprio ou patrocinado) 2 - estacionou em local público (na rua)
M	VALOR_EST	Valor do estacionamento	02	03 dígitos, 2 casas decimais (R\$/mês, ref. out/2007)

Fonte: Elaboração própria a partir das OD-1977, OD-1987, OD-1997 e OD-2007

5.2 Critérios de Validação

Algumas variáveis (colunas) inicialmente não identificadas como de interesse para este trabalho foram descartadas. Dado o interesse em investigar o comportamento de homens e mulheres nos deslocamentos diários, o critério utilizado para a exclusão de observações (linhas) foi a variável “SEXO” ser igual a 0 na base de dados. Assim, após a preparação da base e da execução de testes de consistência, foram excluídas 1561 linhas - todas pertencentes a 1977 e cujo fator de expansão de viagens também era igual a 0. O pequeno impacto na quantidade de domicílios, pessoas e viagens (registros) está expresso

na Tabela 4.

Tabela 4 – Quantidade de viagens e de domicílios, por ano, antes e depois da preparação do BDU

Ano	Número de Domicílios		Número de Pessoas		Número de Viagens	
	Antes	Depois	Antes	Depois	Antes	Depois
1977	26.132	24.613	108.069	108.027	230.606	229.045
1987	26.070	26.070	110.813	110.813	223.926	223.926
1997	23.841	23.841	98.780	98.780	199.647	199.647
2007	29.957	29.957	91.405	91.405	196.698	196.698
Total	106.000	104.481	409.067	409.025	850.877	849.316

Fonte: Compilação a partir de ([METRÔ-SP, 1977](#); [METRÔ-SP, 1987](#); [METRÔ-SP, 1997](#); [METRÔ-SP, 2007](#))

Os critérios para a adoção de “NA” foram os seguintes:

- (i) Quando em um ano específico não havia informação levantada para uma determinada variável.
- (ii) Quando, mesmo essa variável tendo sido levantada, não houve preenchimento ou resposta.
- (iii) Quando uma determinada informação não existe pelo fato de que a viagem não foi feita.

5.3 Formulação de Novas Variáveis

Foram criadas variáveis de interesse, relativas a viagens, pessoas ou famílias, para gerar informações necessárias de forma a subsidiar as análises subsequentes. Além dessas, também foram criadas 4 *dummies* de ano (ANO_77, ANO_87, ANO_97 e ANO_97) para que fosse possível, nas regressões, tentar captar o efeito do tempo.

5.3.1 Variáveis relativas às viagens

- **Faixa horária e *dummies* de período:** A variável FAIXA_HORARIA foi criada para classificar as viagens em faixas horárias, considerando o horário de término da viagem obtido a partir da variáveis H_CHEG (hora de chegada) e MIN_CHEG (minuto de chegada). Isso porque o horário de chegada corresponde ao horário de início da atividade que motivou a viagem, e são as atividades que, de fato, interessam e motivam o comportamento humano. Foram adotadas sete categorias para a divisão dos períodos do dia, derivados do estudo de [Vespucci \(2003\)](#). Para cada faixa horária, foi gerada uma *dummy* de período correspondente. Tal divisão, também adotada

por Germani (2005), foi a que melhor refletiu as concentrações de atividades e é a seguinte:

- 0 -> não fez viagem
- 1 -> madrugada (entre 0h01 e 5h00), originou a *dummy* VIAG_PER_MADRUG
- 2 -> começo da manhã (entre 5h01 e 9h00), originou a *dummy* VIAG_PER_COM_MAN
- 3 -> manhã (entre 9h01 e 12h), originou a *dummy* VIAG_PER_MANHA
- 4 -> meio-dia tarde (entre 12h01 e 14h), originou a *dummy* VIAG_PER_MEIODIA
- 5 -> tarde (entre 14h01 e 17h), originou a *dummy* VIAG_PER_TARDE
- 6 -> começo da noite (17h01 e 22h), originou a *dummy* VIAG_PER_COM NOI
- 7 -> noite (22h01 e 0h), originou a *dummy* VIAG_PER_NOITE

- **Contagem da utilização dos modos:** Como cada viagem pode comportar a utilização de até 4 modos³ é feita a contagem de quantas vezes um determinado modo foi utilizado em uma viagem. Quem não fez viagem terá toda as variáveis listadas a seguir setadas com 0. A consolidação da quantidade de modos diferentes utilizados na viagem é sumarizado na variável VIAG_NO_MODOS.

VIAG_MODO_ONIBUS -> viagens feitas por ônibus de linha (municipal ou intermunicipal), ônibus escolar, ônibus de empresa, lotação e van

VIAG_MODO_DIRIG -> viagens feitas por automóvel com a pessoa dirigindo

VIAG_MODO_PASS -> viagens feitas por automóvel com a pessoa como passageiro, incluindo as viagens de táxi

VIAG_MODO TREM -> viagens feitas por metrô ou trem

VIAG_MODO_MOTO -> viagens feitas por motocicleta

VIAG_MODO_BICI -> viagens feitas por bicicleta

VIAG_MODO_APE -> viagens feitas a pé

VIAG_MODO_OUTROS -> viagens não contempladas pelas categorias anteriores

VIAG_NO_MODOS -> quantos modos diferentes fora utilizados, por viagem, de acordo com estas categorias

- **Dummies dos motivos de viagem:** Cada viagem só tem um motivo na origem e um motivo no destino. Foram considerados os motivos do destino (que relacionam-se com a atividade fim daquele deslocamento) e criados seis marcadores, conforme pode ser observado a seguir. Vespucci⁴ (2003) e Germani⁵ (2005) agregam as viagens de manutenção de compras e saúde com as de lazer e outros, além de desagregar as

³ As pesquisas OD-77 e OD-87 previam campo para até 3 modos, já as pesquisas OD-97 e OD-07 previam a utilização de até 4 modos por viagem.

⁴ Fez estudos sobre a sequência diária de atividades e cadeias de viagens a partir das pesquisas OD-87 e OD-97.

⁵ Fez estudos sobre comportamento de demanda usando método de alinhamento de sequências multidimensionais a partir da pesquisa OD-97.

viagens de motivo residência em temporária e final. [Dalmaso⁶](#) (2009) verificou que os resultados eram muito parecidos categorizando o motivo residência temporária (com realização de outras viagens posteriormente) e final (sem realização de outras viagens posteriormente). Além disso, este autor constatou que havia grandes diferenças nos resultados mantendo o motivo manutenção único ou desmembrando-o em compras/saúde e lazer/outros. Por fim, ele considerou o motivo “servir passageiro”, que significa que a pessoa ao fazer uma viagem para acompanhar alguém (à escola, ao médico, etc.) não faz essa viagem por um motivo seu, mas acompanhando o motivo da outra pessoa, a quem “serve”. Optou-se por seguir essa divisão de categorias de motivos de [Dalmaso](#) porque a revisão de literatura indica que as mulheres são as principais cuidadoras do núcleo familiar, assim, espera-se que apresentem maior frequência no motivo “servir passageiro”.

VIAG_MOTIVO_SERV_PAS -> marcador da viagem motivo servir passageiro
 VIAG_MOTIVO_TRAB -> marcador da viagem motivo trabalho
 VIAG_MOTIVO_EDUC -> marcador da viagem motivo educação
 VIAG_MOTIVO_RES -> marcador da viagem motivo residência
 VIAG_MOTIVO_MANUT_COMPRA -> marcador da viagem motivo manutenção da casa (compras) ou da família (saúde)
 VIAG_MOTIVO_LAZER_OUTROS -> marcador da viagem motivo destino lazer ou outros

- **Dummies de viagens inter-zonas:** Foi criado um marcador que indicasse quando uma viagem teve sua zona de destino distinta da sua zona de origem. Não se busca mensurar distância ou tempo, mas a eventual superação de barreiras urbanas. Uma viagem intrazonal pode ser mais longa (em tempo e distância) que uma viagem interzonal, e mesmo assim ser preferível. Em São Paulo, por exemplo, a distância de uma margem a outra da Marginal Pinheiros, ou mesmo o tempo levado apenas para atravessar um de suas pontes, não representam valores muito grandes. Porém, é comum que os municípios optem por fazer suas atividades (e deslocamentos) apenas de um lado do rio, pois este constitui-se uma barreira urbana, um ponto de impedância no sistema de deslocamentos. O desenho do zoneamento das pesquisas OD não foi formulado com a intenção de evidenciar as barreiras urbanas, mas mesmo assim, as reflete.

5.3.2 Variáveis relativas às pessoas

- **Faixa etária e dummies de faixa etária:** A partir da variável IDADE, foi criada a variável FAIXA_ETARIA, seguindo os critérios adotados pelo IBGE, resultando

⁶ Fez estudos sobre a identificação e caracterização de grupos de indivíduos segundo padrões de sequências de atividades multidimensionais a partir da pesquisa OD-97.

em 21 faixas etárias, conforme pode ser observado a seguir. Para cada faixa etária, foi gerada uma *dummy* correspondente.

0 -> 0 a 4 anos, originou a *dummy* FX_ET_0
 1 -> 5 a 9 anos, originou a *dummy* FX_ET_1
 2 -> 10 a 14 anos, originou a *dummy* FX_ET_2
 3 -> 15 a 19 anos, originou a *dummy* FX_ET_3
 4 -> 20 a 24 anos, originou a *dummy* FX_ET_4
 5 -> 25 a 29 anos, originou a *dummy* FX_ET_5
 6 -> 30 a 34 anos, originou a *dummy* FX_ET_6
 7 -> 35 a 39 anos, originou a *dummy* FX_ET_7
 8 -> 40 a 44 anos, originou a *dummy* FX_ET_8
 9 -> 45 a 49 anos, originou a *dummy* FX_ET_9
 10 -> 50 a 54 anos, originou a *dummy* FX_ET_10
 11 -> 55 a 59 anos, originou a *dummy* FX_ET_11
 12 -> 60 a 64 anos, originou a *dummy* FX_ET_12
 13 -> 65 a 69 anos, originou a *dummy* FX_ET_13
 14 -> 70 a 74 anos, originou a *dummy* FX_ET_14
 15 -> 75 a 79 anos, originou a *dummy* FX_ET_15
 16 -> 80 a 84 anos, originou a *dummy* FX_ET_16
 17 -> 85 a 89 anos, originou a *dummy* FX_ET_17
 18 -> 90 a 94 anos, originou a *dummy* FX_ET_18
 19 -> 95 a 99 anos, originou a *dummy* FX_ET_19
 20 -> mais de 100 anos, originou a *dummy* FX_ET_20

- **Faixa de renda individual:** Os valores de renda individual foram levados para out/2007 e classificados em sete categorias explicitadas a seguir, conforme a quantidade de salários mínimos (SM)⁷.

0 -> sem renda
 até R\$380 (exclusive) -> até 1 SM
 de R\$380 (inclusive) até R\$760 (exclusive) -> de 1 a 2 SM
 de R\$760 (inclusive) até R\$1140 (exclusive) -> de 2 a 3 SM
 de R\$1140 (inclusive) até R\$1900 (exclusive) -> de 3 a 5 SM
 de R\$1900 (inclusive) até R\$3800 (exclusive) -> de 5 a 10 SM
 de R\$3800 (inclusive) até R\$5700 (exclusive) -> de 10 a 15 SM
 mais de R\$5700 (inclusive) -> mais de 15 SM

- **Contagem da utilização dos modos:** Baseadas nas variáveis de contagem de

⁷ O valor de um salário mínimo era R\$380,00 em 2007 segundo a Lei do salário mínimo vigente e disponível em <https://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2007/Lei/L11498.htm>. Acesso em 30 de setembro de 2015.

modo das viagens, foram feitos agrupamentos por pessoas, e somadas a quantidade de vezes que um determinado modo é utilizado para cada pessoa - ver Equação (5.1). A variável PESS_NO_MODOS sintetiza a quantidade de modos diferentes que a pessoa utilizou ao longo do dia.

$$PESS_MODO_k = \sum_{i=1}^n VIAG_MODO_k \quad (5.1)$$

- **Contagem dos motivos de viagem:** Baseadas nas *dummies* dos motivos das viagens, foram feitos agrupamentos por pessoas, e somadas a quantidade de vezes que um determinado motivo foi declarado para cada pessoa - ver Equação (5.2). A variável PESS_NO_MOTIVOS sintetiza a quantidade de motivos diferentes que levou a pessoa a se deslocar ao longo do dia.

$$PESS_MOTIVO_k = \sum_{i=1}^n VIAG_MOTIVO_k \quad (5.2)$$

- **Distância total:** A informação das distâncias percorridas por viagem já está armazenada na variável DIST_VIAG do BDU. A variável PESS_DIST_TOTAL summariza essa informação para a pessoa, conforme Equação (5.3).

$$PESS_DIST_TOT = \sum_{i=1}^n DIST_VIAG \quad (5.3)$$

- **Distância média:** A distância média percorrida pela pessoa é a distância total da pessoa dividida pelo seu número total de viagens - ver Equação (5.4).

$$PESS_DIST_MED = \frac{\sum_{i=1}^n DIST_VIAG}{TOT_VIAG} \quad (5.4)$$

- **Duração total:** A informação das distâncias percorridas por viagem já está armazenada na variável DURACAO do BDU. A variável PESS_DURACAO_TOTAL summariza essa informação para a pessoa, conforme Equação (5.5).

$$PESS_DURACAO_TOT = \sum_{i=1}^n DURACAO \quad (5.5)$$

- **Duração média:** A duração média percorrida pela pessoa é a duração total da pessoa dividida pelo seu número total de viagens - ver Equação (5.6)

$$PESS_DURACAO_MED = \frac{\sum_{i=1}^n DURACAO}{TOT_VIAG} \quad (5.6)$$

5.3.3 Variáveis relativas às famílias

- **Tamanho da família:** O tamanho da família (variável TOT_PESS) é determinado pelo máximo número da pessoa (NO_PESS) de uma determinada família.
- **Faixa de renda familiar:** A partir da variável renda familiar, em valores de outubro de 2007, foi criada a variável FAIXA_REN_FAM, que contém a faixa de renda familiar segundo o critério de classificação do IBGE para classes econômicas ⁸ (A, B, C, D e E) – ver categorias abaixo. Portanto, as faixas de renda familiar foram construídas a partir da quantidade de salários mínimos, sendo o valor de um salário mínimo R\$380,00 em 2007 ⁹.

0 -> sem renda
 até R\$760 (exclusive) -> Classe E
 de R\$760 (inclusive) até R\$1520 (exclusive) -> Classe D
 de R\$1520 (inclusive) até R\$3800 (exclusive) -> Classe C
 de R\$3800 (inclusive) até R\$7600 (exclusive) -> Classe B
 mais de R\$7600 (inclusive) -> Classe A

- **Distância total:** A informação das distâncias percorridas por viagem já está armazenada na variável DIST_VIAG do BDU. A variável FAM_DIST_TOTAL sumariza essa informação para a família, conforme Equação (5.7).

$$FAM_DIST_TOT = \sum_{i=1}^n DIST_VIAG \quad (5.7)$$

⁸ Definição de Classe Econômica – Fonte: <http://www.sae.gov.br/wp-content/uploads/ebook_ClasseMedia1.pdf> Acesso em 30 de setembro de 2015.

⁹ Lei do salário mínimo de 2007: <https://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2007/Lei/L11498.htm> Acesso em 30 de setembro de 2015.

- **Distância média:** A distância média percorrida da família é a soma das distâncias das pessoas dividida pelo seu total de viagens da família - ver Equação (5.8).

$$FAM_DIST_MED = \frac{\sum_{i=1}^n DIST_VIAG}{FAM_VIAG_TOT} \quad (5.8)$$

- **Duração total:** A duração total das viagens da família são armazenadas na variável FAM_DURACAO_TOTAL, que summariza essa informação para a família, conforme Equação (5.9).

$$FAM_DURACAO_TOT = \sum_{i=1}^n DURACAO \quad (5.9)$$

- **Duração média:** A duração média das viagens da família são armazenadas na variável FAM_DURACAO_MED, que toma a duração total da família dividida pelo seu número total de viagens da família - ver Equação (5.10)

$$FAM_DURACAO_MED = \frac{\sum_{i=1}^n DURACAO}{FAM_VIAG_TOT} \quad (5.10)$$

- **Presença de automóveis na família:** Foi gerada uma *dummy* (PRESENCA_AUTO) para indicar a presença de automóvel na família. Segundo Strambi e Van de Bilt (2001 apud PEIXOTO, 2002) a motorização é um fator de grande influência sobre o padrão de mobilidade. Peixoto (2002), ao estudar a evolução temporal da mobilidade na Região Metropolitana de Porto Alegre, entre 1986 e 1997, detectou ser a posse de veículos uma característica importante para segmentação de grupos de comportamento semelhantes. Geralmente, a utilização do automóvel é decidida dentro de uma determinada dinâmica familiar. No contexto familiar surgem questões como: em não existindo um automóvel para cada pessoa acima de 18 anos, quem tem a prioridade no uso? Quem trabalha mais longe de casa? Quem tem uma rotina mais complexa? Quem leva os filhos na escola? Quem é autônomo?
- **Presença de crianças na família:** Como um indicativo do estágio no ciclo de vida familiar (ORTÚZAR; WILLUMSEN, 1994), foram geradas *dummies* para indicar, na família, a presença de crianças/adolescentes de quatro faixas etárias, a saber: PRESENCA_FILH_ate4 -> indica se a família conta com crianças até 4 anos

PRESENCA_FILH_5a9 -> indica se a família conta com crianças entre 5 e 9 anos

PRESENCA_FILH_10a14 -> indica se a família conta com crianças entre 10 e 14 anos

PRESENCA_FILH_15a19 -> indica se a família conta com adolescentes entre 15 e 19 anos

- **Presença de idosos na família:** A presença de idoso na família, também é um indicativo de estágio no ciclo de vida familiar (ORTÚZAR; WILLUMSEN, 1994), tanto mais expressivo quanto mais a população envelhece. Oliveira (2014) ao estudar a correlação e efeitos das composições familiares na mobilidade do idoso, considerando aspectos econômicos e sociais, constatou que o tamanho da família afeta negativamente a mobilidade dos idosos e que quanto maior o número de idosos na mesma família, maior sua mobilidade. Vasconcellos (2001) aponta que aspectos econômicos e sociais influenciam os padrões de mobilidade das pessoas e destaca, dentre os sociais, a família como um dos fatores importantes na tomada de decisões em relação aos deslocamentos e arranjos das atividades diárias. Como o próprio conceito de quem é o idoso vem mudando com o tempo, e este trabalho observa quatro *cross sections* e três décadas, decidiu-se por destacar a presença de idosos em duas faixas etárias:

PRESENCA_IDOSO_60_70 -> indica se a família conta com pessoa idosa entre 60 e 70 anos

PRESENCA_IDOSO_70 -> indica se a família conta com pessoa idosa com mais de 70 anos

6 Análises Realizadas

6.1 Estatísticas Descritivas

A seguir serão apresentadas algumas estatísticas descritivas das variáveis, bem como outras observações julgadas pertinentes.

A variável **ANO**, de natureza qualitativa, não possui “NA” e conta com quatro valores únicos, cujas frequências absolutas são apresentadas na Tabela 5. Esta variável não era original dos bancos de dados e foi inserida para poder gerar *dummies* de marcação da *cross-section* observada para análises de caráter longitudinal.

Tabela 5 – Estatísticas da variável “ANO”

Categoria	Frequência Absoluta
1 - 1977	229.046
2 - 1987	223.926
3 - 1997	199.640
4 - 2007	196.698

A variável **CD_ENTRE**, *dummy*, não possui “NA” e conta com dois valores únicos (0 e 1) em que o 1 indica se houve viagem (83%) e 0, se não houve viagem (17%). Sua frequência absoluta, segundo o ano e sexo é apresentada na Tabela 6.

Tabela 6 – Estatísticas da variável “CD_ENTRE”

ANO	1977	1987	1997	2007	Total
CD_ENTRE=0	41.514	40.808	36.106	27.033	145.461
CD_ENTRE=1	187.532	183.118	163.534	169.665	704.849

A variável **DIA_SEM**, de natureza qualitativa, conta com cinco valores únicos, cujas frequências absolutas são apresentadas na Tabela 7. Ela possui 296.893 registros “NA”, sendo 229.046 de 1977, ano em que esta variável não foi levantada. Os demais *missing values* foram adotados para o caso em que os valores eram originalmente iguais a 0, dividindo-se entre os anos de 1987 e 2007 e indicando observações relativas a não-viagens. Nas transformações efetuadas na preparação da base de dados, em 1987, foram feitas duas correções porque pressupôs-se erro de preenchimento: (i) 187 casos com registro 1 (valor que não consta nos layouts oficiais) que foram transformados para 2 (segunda-feira); e (ii) 170

casos de registro 7 (valor que não consta nos layouts oficiais) que foram transformados para 6 (sexta-feira). Vale observar que a sexta-feira (DIA_SEM=6) é superrepresentado devido à metodologia de coleta utilizada: na pesquisa, domiciliar, pregunta-se ao(à) respondente sobre as viagens que fizera no dia anterior e sábado é o dia em que mais se encontram as pessoas em casa, assim, há mais respostas sobre a sexta-feira do que sobre os demais dias da semana.

Tabela 7 – Estatísticas da variável “DIA_SEM”

ANO	1977	1987	1997	2007	Total
DIA_SEM=2	0	27.771	32.956	25.129	85.856
DIA_SEM=3	0	34.089	36.817	23.764	94.670
DIA_SEM=4	0	35.275	37.695	27.242	100.212
DIA_SEM=5	0	33.003	32.924	28.010	99.937
DIA_SEM=6	0	52.980	59.248	65.514	177.742

As variáveis **UCOD** (domicílio, escola, trabalhos, origem e destino), qualitativas, estabelecem correspondência com as zonas conforme Anexo A.

As **ZONAS** e **SUBZONAS** (domicílio, escola, trabalhos, origem e destino), qualitativas, de cada ano podem ser observadas no Anexo B. Não foi disponibilizado o mapa com as subzonas de 1977 e, em 2007, não existem subzonas.

As **coordenadas X e Y** (domicílio, escola, trabalhos, origem e destino), variáveis métricas, foram extraídas diretamente dos mapas pelo uso do software QGIS¹. Em 1977, elas foram determinadas a partir dos centroides das zonas. Em 1987 e 1997, elas foram determinadas a partir dos centroides das subzonas. Já em 2007, o levantamento contou com a tecnologia GPS² e o banco de dados já continha as coordenadas.

As variáveis **F_DOM**, **F_FAM** e **F_PESS** são *dummies* que identificam o primeiro registro do domicílio, família e pessoa, respectivamente. Não contam com “NA” e têm suas frequências absolutas apresentadas pelas Tabelas 8, 9 e 10. Vale observar que os valores correspondentes a F_DOM=1, F_FAM=1 e F_PESS=1 significam exatamente os tais de domicílios, famílias e pessoas entrevistados em cada um dos anos.

As variáveis **FE_DOM**, **FE_FAM** e **FE_PESS**, variáveis métricas, são fatores de expansão (pesos) de domicílio, família e pessoa, respectivamente, já fornecidos pelas bases de dados e que não sofreram transformação alguma.

¹ “O QGIS é um Sistema de Informação Geográfica (SIG) de Código Aberto licenciado segundo a Licença Pública Geral GNU.” Fonte: <http://www.qgis.org/pt_BR/site/about/index.html> Acesso em 26 de dezembro de 2015

² GPS significa *Global Positioning System* e é um sistema de posicionamento que permite localização em qualquer ponto da Terra desde que o receptor móvel esteja no campo de visão de quatro satélites GPS. Até meados dos anos 2000, a utilização do GPS para uso civil ainda tinha pouca precisão.

Tabela 8 – Estatísticas da variável “F_DOM”

ANO	1977	1987	1997	2007	Total
F_DOM=0	204.433	197.856	175.799	166.741	744.829
F_DOM=1	24.613	26.070	23.841	29.957	104.481

Tabela 9 – Estatísticas da variável “F_FAM”

ANO	1977	1987	1997	2007	Total
F_FAM=0	202.889	195.709	172.795	165.843	737.236
F_FAM=1	26.157	28.217	26.845	30.855	112.074

Tabela 10 – Estatísticas da variável “F_PESS”

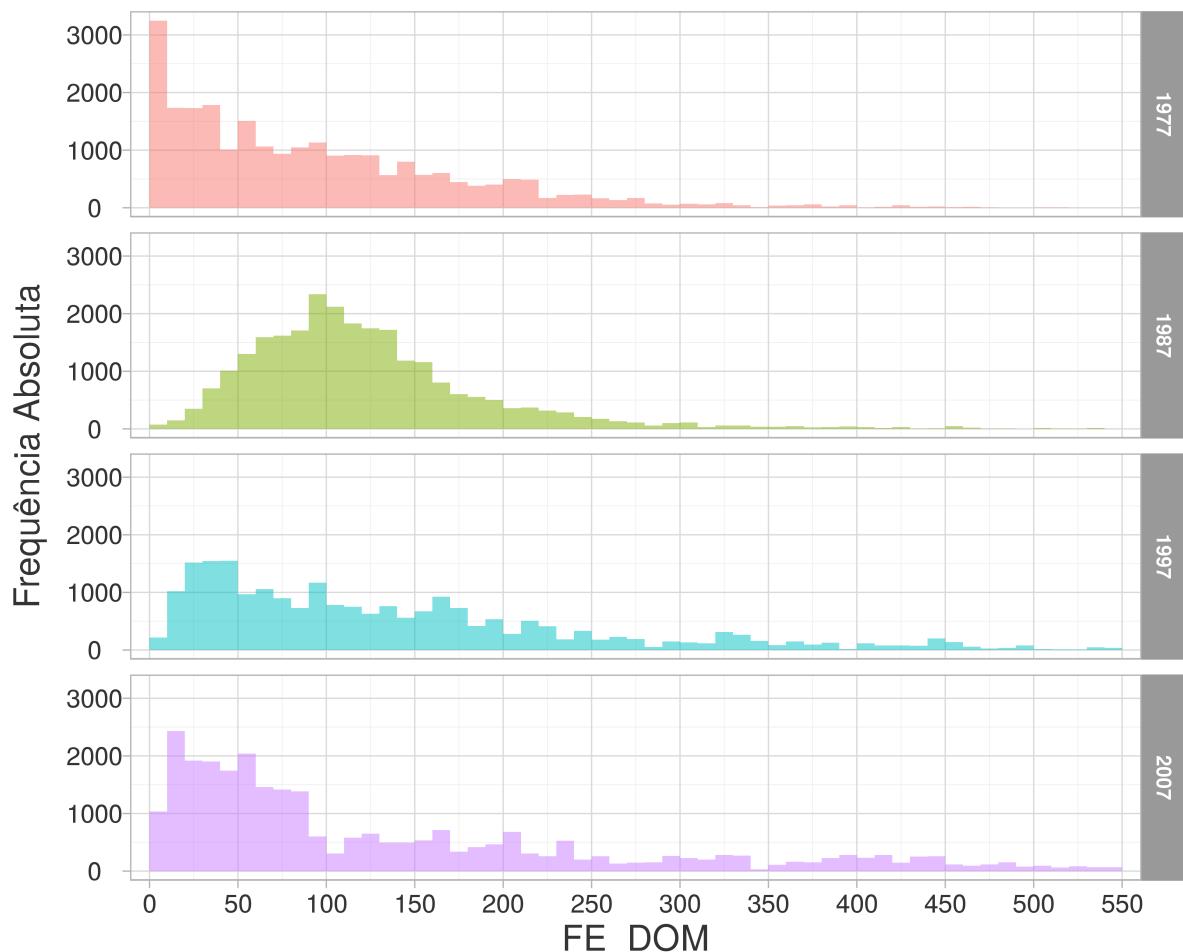
F_PESS=0		ANO				Total
SEXO	1977	1987	1997	2007		
Masculino	72.040	62.264	52.146	51.732	238.182	
Feminino	48.978	50.849	48.714	53.561	202.102	
Total	121.018	113.113	100.860	105.293	440.284	
F_PESS=1		ANO				Total
SEXO	1977	1987	1997	2007		
Masculino	52.162	53.176	47.326	42.289	194.953	
Feminino	55.866	57.637	51.454	49.116	214.073	
Total	108.028	110.813	98.780	91.405	409.026	

Os fatores de expansão da família e da pessoa têm intervalos praticamente iguais aos do domicílio (ver Tabela 11), sendo que as médias, medianas e intervalos inter-quartis se alteram, pouco, dentro do mesmo ano. O Gráfico 3 apresenta as distribuições por ano da variável FE_DOM, cujo máximo valor da abscissa ficou em 550 pois o maior valor do intervalo inter quartil foi 535,5 em 2007. As variáveis FE_FAM e FE_PESS possuem distribuições bastante semelhantes a esta.

Tabela 11 – Estatísticas da variável “FE_DOM”

ANO	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
1977	0,50	25,75	72,09	139,25	1562,75
1987	1,02	78,32	110,49	150,89	2210,00
1997	1.00	51,41	117,14	213,09	2548,63
2007	2,55	40,92	87,42	234,90	2256,41
Geral	0,50	48,67	99,20	174,06	2548,63
ANO	Média	Desvio Padrão	Assimetria	Curtose	Nº de “NA”
1977	94,92	92,42	2,63	18,78	0
1987	127,53	90,95	5,02	54,44	0
1997	178,75	209,77	3,51	20,44	0
2007	183,82	228,96	2,69	10,66	0
Geral	147,67	174,64	3,79	23,63	0

Gráfico 3 – Distribuição da variável “FE_DOM”, por ano



A variável **TIPO_DOM**, em 1977 e 1987, contava somente com as categorias particular e coletivo; já em 1997 e 2007 passou a existir também a categoria favela. Foram adotadas as categorias coletivo e particular, de forma que funcionasse como uma **dummy** indicativa de domicílio particular (1) ou não (0). Quem não respondeu foi tratado como “NA”, encerrando 4 domicílios (23 registros) em 1987, conforme pode ser observado na Tabela 12. Nota-se que o percentual de moradias unifamiliares sempre permaneceu, em média, superior a 90%.

Tabela 12 – Estatísticas da variável “TIPO_DOM”

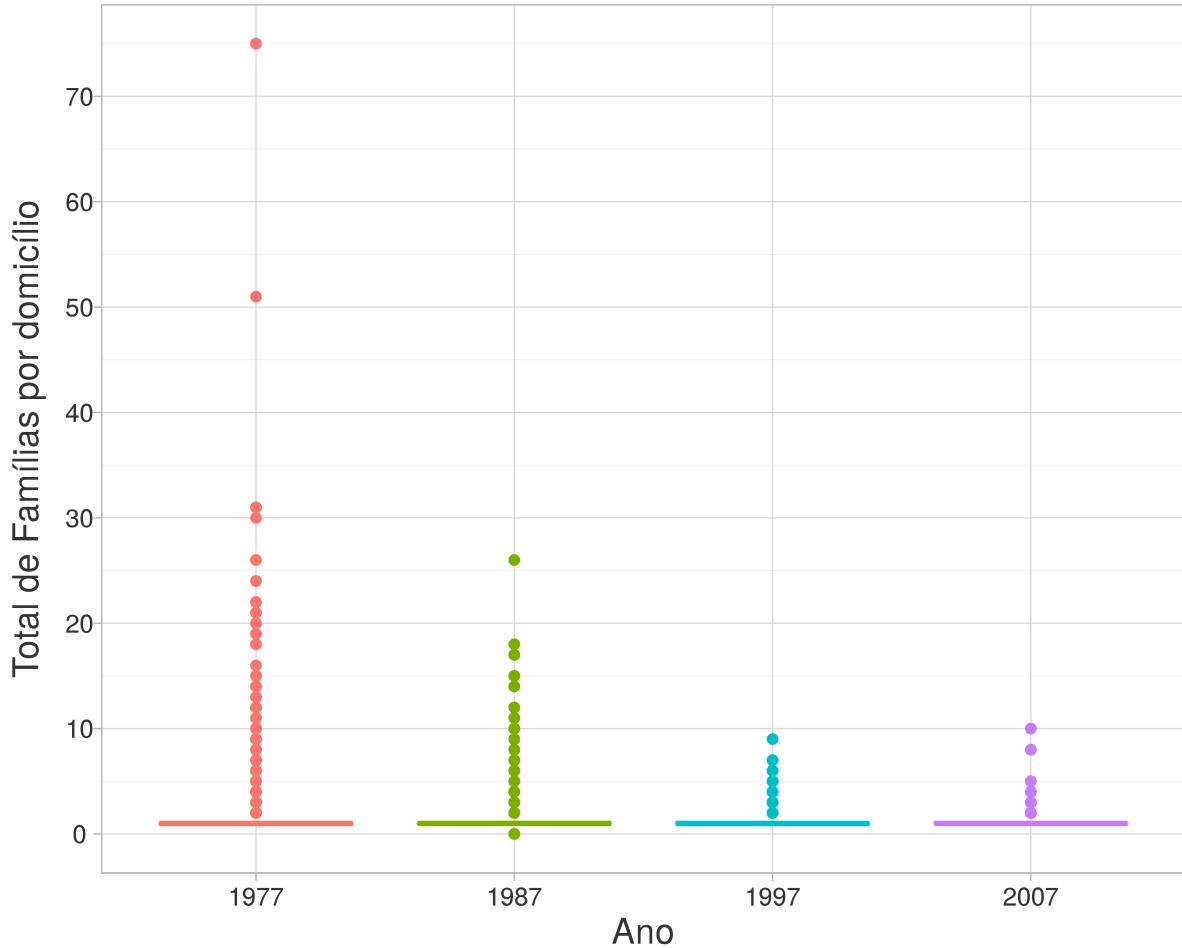
ANO	1977	1987	1997	2007	Total
TIPO_DOM=0	148	207	2.054	1.195	3.604
TIPO_DOM=1	24.465	25.859	21.787	28.762	100.873
TIPO_DOM=“NA”	0	4	0	0	4

A variável **TOT_FAM**, um dado de contagem, indica quantas famílias existem no domicílio e tem suas estatísticas descritivas apresentadas na Tabela 13. O valor médio indica haver pouco mais de uma família por domicílio, sendo a maior média de 1997 - dado coerente com a menor porcentagem de domicílios individuais (91%) indicada pela variável TIPO_DOM. A Figura 7 mostra que a influência dos *outliers* diminui com o tempo, o que também se reflete na queda dos desvios padrão, que indica menor dispersão dos dados.

Tabela 13 – Estatísticas da variável “TOT_FAM”

ANO	Média	Desvio Padrão	Assimetria	Curtose	Máximo
1977	1,08	0,99	33,70	1777,74	75
1987	1,12	0,61	13,72	306,37	26
1997	1,13	0,45	4,85	32,42	9
2007	1,03	0,23	14,00	346,13	10
Geral	1,09	0,62	35,68	2734,92	75

Figura 7 – Box plot da variável “TOT_FAM”, por ano



A variável **COND_MORA**, de natureza qualitativa, conta com três valores únicos, cujas frequências absolutas de registros são apresentadas na Tabela 14. A categoria “alugada” (1) que em 1977 representava 35% da amostra, caiu para 23% em 2007. A posse de residência (categoria “própria” - 2), subiu de 57% para 68% da amostra anual. A categoria “outros” (3), que abrange as categorias originais “cedida”, “outros” e “não se aplica”, oscila numa faixa próxima dos 10%. Quem não respondeu foi tratado como “NA”, encerrando 717 domicílios: 9 em 1977, 264 em 1987, 26 em 1997 e 460 em 2007.

Tabela 14 – Estatísticas da variável “COND_MORA”

ANO	1977	1987	1997	2007	Total
COND_MORA=1	9.174	7.967	5.168	7.268	29.577
COND_MORA=2	14.984	17.429	18.040	21.062	71.515
COND_MORA=3	1.990	2.557	3.611	1.830	2.065
COND_MORA=“NA”	9	264	26	460	759

Não foram mantidas no BDU as variáveis relativas aos bens de consumo, pois a função delas era servir de *input* para determinar a renda atribuída (na ausência de declaração da renda), informação de que já se dispõe no banco de dados. Apenas os bens que são meios de transporte foram mantidos (automóveis, motocicletas e bicicletas) e serão explorados mais adiante.

Os valores monetários de renda familiar, renda individual e valor de estacionamento encontravam-se, em cada banco de dados, em um mês de referência diferente (segundo Quadro 2). Foram estudadas as possibilidades de correção pelos seguintes índices:

- IPC-Brasil: índice de preços ao consumidor, que mede a variação de preços de um conjunto fixo de bens e serviços componentes de despesas habituais de famílias com nível de renda situado entre 1 e 33 salários mínimos mensais. Sua pesquisa de preços se desenvolve diariamente, cobrindo as sete principais capitais do país. Suas variações (DI, M e 10) referem-se ao período da coleta dos preços. A série histórica que se obteve de fontes oficiais (página da FGV e do Banco Central) não disponibilizava valores anteriores a 1989.
- IGP: índice geral de preços, (a partir de 1950) é a média aritmética ponderada de três outros índices de preços (desde : Índice de Preços ao Produtor Amplo (50%), Índice de Preços ao Consumidor (30%) e Índice Nacional de Custo da Construção (10%). O IGP-M/FGV se refere ao período do dia vinte e um do mês anterior ao dia vinte do mês de referência e o IGP-DI/FGV se refere ao período do dia um ao dia trinta do mês em referência. A série histórica que se obteve de fontes oficiais (página da FGV e do Banco Central) não disponibilizava valores anteriores a 1989 para o IGP-M, ao passo que o IGP-DI³ dispunha de valores para correção desde a década de 1940.
- IPCA: índice geral de preços ao consumidor amplo, que mede a variação de preços de um conjunto fixo de bens e serviços componentes de despesas habituais de famílias com nível de renda situado entre 1 e 40 salários mínimos mensais. Sua pesquisa de preços se desenvolve diariamente, cobrindo as dez principais capitais do país e Brasília. A série histórica que se obteve de fontes oficiais (página do IBGE e do Banco Central) divulga valores após dezembro de 1979. Sua variação IPCA-E é ainda mais recente (1991) e tem divulgação trimestral.
- INPC: índice nacional de preços ao consumidor amplo, que mede a variação de preços de um conjunto fixo de bens e serviços componentes de despesas habituais de famílias com nível de renda situado entre 1 e 5 salários mínimos mensais. Sua pesquisa de preços se desenvolve continuamente, cobrindo as dez principais capitais do país e Brasília. A série histórica que se obteve de fontes oficiais (página do IBGE e do Banco Central) divulga valores após março de 1979.

³ Metodologia do IGP-DI disponível em: <<http://portalibre.fgv.br/lumis/portal/file/fileDownload.jsp?fileId=8A7C82C54DB5CA9F014DD9322ADD306E>> Acesso em 30 de agosto de 2015

Assim, pela abrangência histórica foi utilizado o IPG-DI para levar os valores de setembro de 1977 para setembro de 1987. Estes valores, bem como os advindos da OD de 1987, foram levados a outubro de 2007 sofrendo correção do INPC. Este mesmo índice foi utilizado para levar os valores de outubro de 1997 a outubro de 2007. Escolheu-se o INPC, dentro os demais disponíveis, devido à faixa de abrangência da renda familiar utilizada em sua determinação. Não se quis priorizar índices cujas cestas tivessem um perfil de consumo alto, dado que as políticas de transporte público devem priorizar especialmente as menores faixas de renda. A composição final dos deflatores utilizados é apresentada na Tabela 15.

Tabela 15 – Deflatores utilizados para correção dos valores monetários para outubro/2007

ANO	1977	1987	1997	2007
Deflator	0,44234590	0,09664666	1,94136464	1

A renda familiar (variável **REN_FAM**) tem seus maiores de média e mediana em 1977, conforme Tabela 16, talvez por reflexo da época do “milagre econômico brasileiro”, comumente atribuído ao período do final dos anos 1960 até a metade dos anos 1970. Após isso, 1987 apresenta a menor média e também queda da mediana, pode ser devido à recessão que marcou o Brasil nos anos 1980, quando se registraram baixo crescimento do PIB, alto nível de desemprego, perda do poder de compra da população, e índices de inflação extremamente elevados. Em 1997, a média da renda familiar sobe, porém a mediana continua a cair, o que pode significar uma recuperação do aquecimento econômico, porém, sem frear o aprofundamento das desigualdades. Com o valor da média de 2007, parece realmente estar ocorrendo uma recuperação econômica (aumento da renda média familiar) e também uma redução das desigualdades (aumento da mediana da renda média familiar) - ver Gráfico 4.

Para a construção do Gráfico 5, a partir da variável **FAIXA_REN_FAM**, foram retiradas as famílias que declararam renda nula. Posto isso, vê-se que as classes E e D, em 1977, correspondiam a 29% da amostra. Em 1987 esse valor vai a 49% e depois começa a decrescer para 45% em 1997 e 37% em 2007. A classe C (classe média) decresce de 1977 (37%) até 1997 (33%) e só aumenta novamente em 2007 (36%). Para estas classificações, foram utilizados sempre os valores em R\$ de outubro de 2007, bem como a classificação de classes econômicas vigente neste mesmo período.

Tabela 16 – Estatísticas da variável “REN_FAM”

ANO	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
1977	0	1.351,51	2.534,08	4.796,58	42.234,2
1987	0	875,62	1.523,44	2.801,21	61.830,1
1997	0	330,03	1.203,65	2.912,05	172.781,0
2007	0	1.144,82	2.080,00	4.000,00	46.000,0
Geral	0	935,73	1.801,63	3.582,21	172.181,0
ANO	Média	Desvio Padrão	Assimetria	Curtose	Nº de “NA”
1977	4.037,27	4.724,88	3,56	18,74	0
1987	2.393,58	2.826,96	4,06	29,85	0
1997	2.453,89	4.148,06	7,20	149,78	0
2007	3.186,19	3.312,25	2,85	13,73	0
Geral	3.009,86	3.845,58	4,74	63,49	0

Uma das variáveis que são *proxy* da renda é a quantidade de automóveis na família (**QT_AUTO**). As Tabelas 17 e 18 mostram que a frota de automóveis de uso familiar vem crescendo tanto no Brasil como na RMSP, assim como a motorização familiar (número de veículos por família) ao longo do tempo. Isto é, mesmo com o crescimento populacional do período, a aquisição de carros pelas famílias aumenta em taxa ainda maior. A média de automóveis por família aumentou com o passar do tempo, conforme observa-se na coluna *média* da Tabela 19.

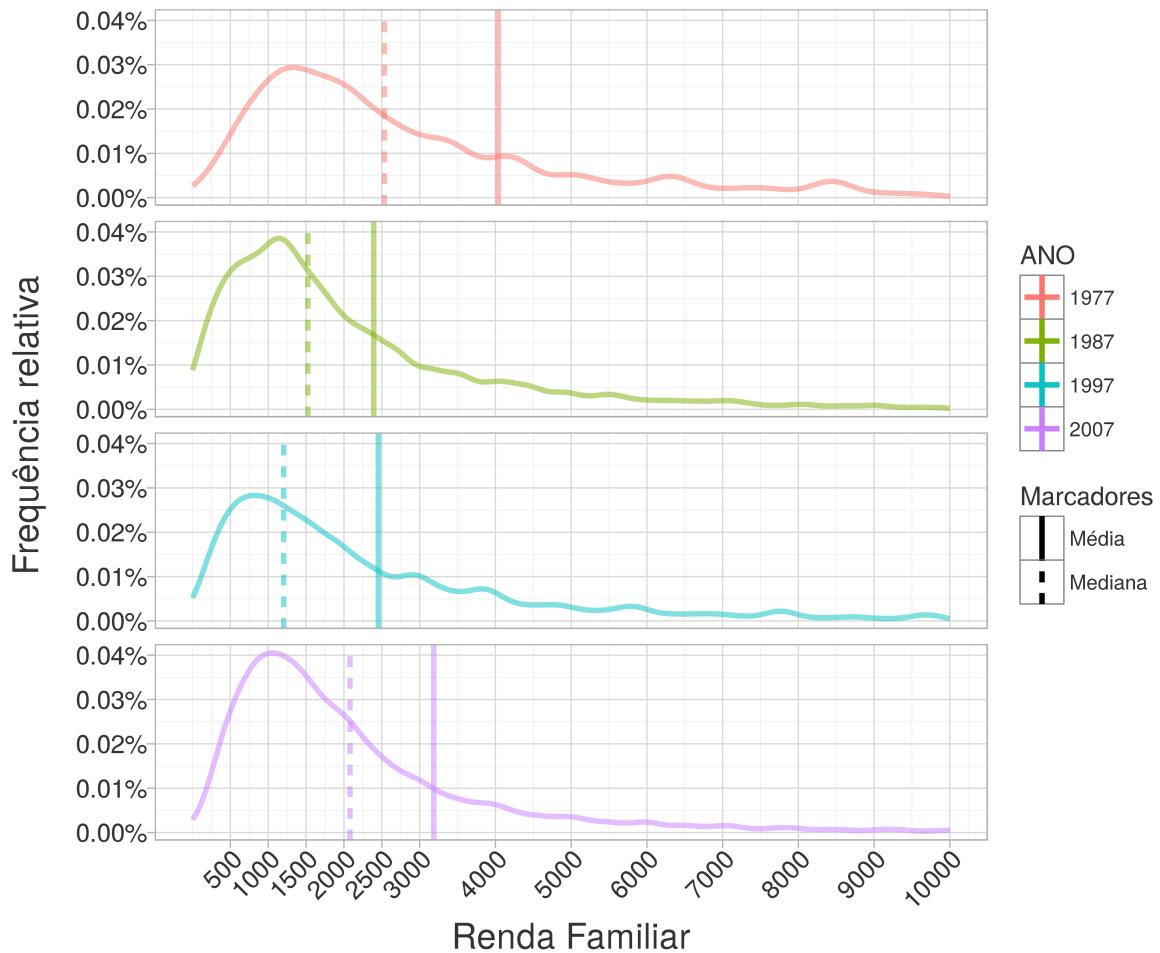
Tabela 17 – Evolução da frota de automóveis na RMSP segundo as Pesquisas OD

ANO	1977	1987	1997	2007
Frota	1.391.832	2.014.474	3.092.238	3.545.263

Nota: Segundo o Departamento Estadual de Trânsito de São Paulo, em 2007 o município de São Paulo contava com cerca de 4,5 milhões de automóveis (Fonte: <<http://www.detran.sp.gov.br/wps/wcm/connect/1eb3cf81-624b-4a71-b626-317730638e74/%28Frota%2B2008%29.pdf?MOD=AJPERES&CACHEID=1eb3cf81-624b-4a71-b626-317730638e74>>). Tal discrepância pode ocorrer porque as Pesquisas OD domiciliares tem seu número de viagens aferido (FE_VIAG deduzido) pelas viagens feitas realizadas nos sistemas de transportes de massa.

Diversos estudos foram realizados para a construção de modelos desagregados que explicassem a posse de autos (RYAN; HAN, 1999; DARGAY; VYTHOULKAS, 1999; DARGAY, 2001; CHU, 2002; KARLAFTIS; GOLIAS, 2002; PFEIFFER; STRAMBI, 2005). É comum a utilização de variáveis como renda, número de trabalhadores, tamanho da família, número de estudantes, presença ou não de crianças na família, sexo e idade

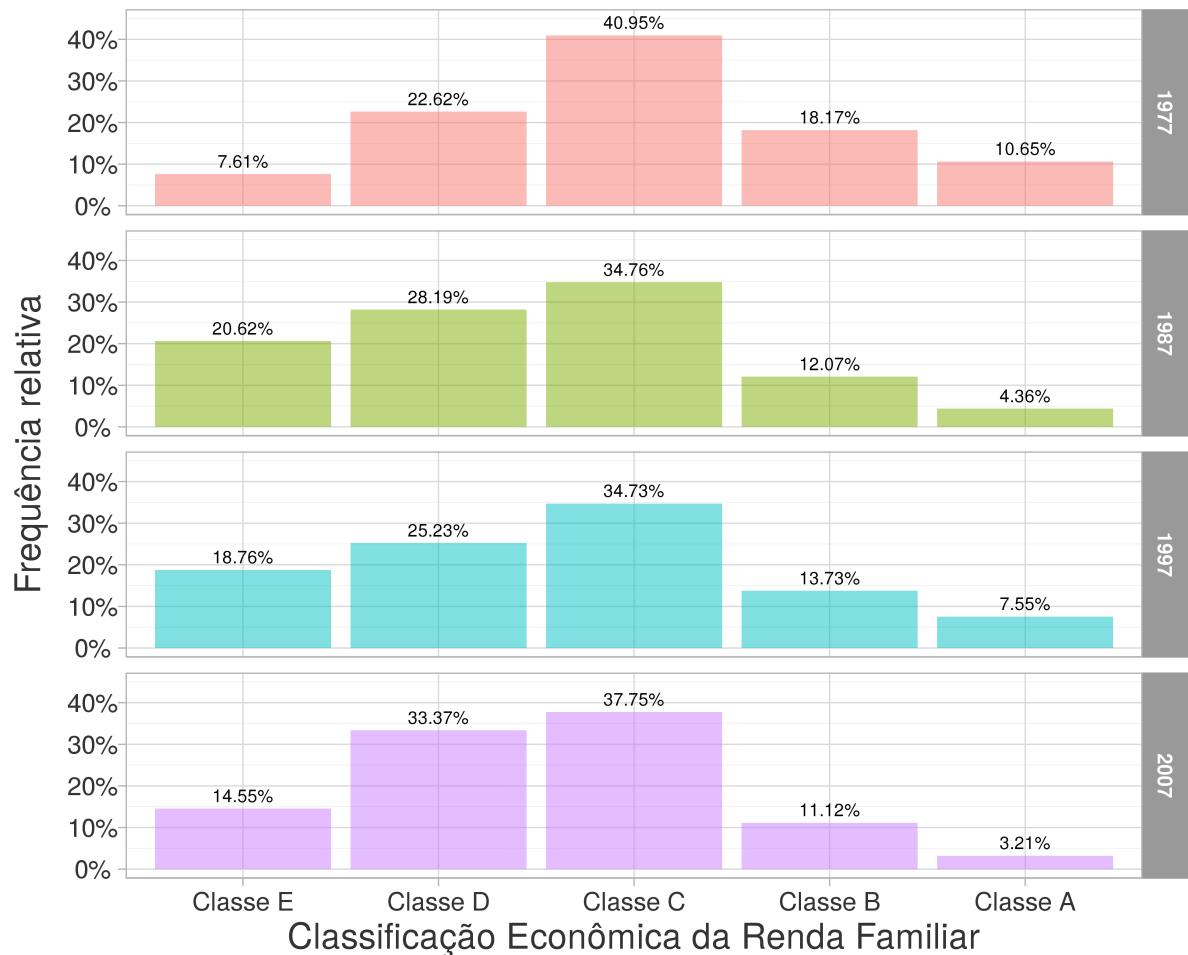
Gráfico 4 – Distribuição da variável “REN_FAM”, por ano



da pessoa responsável pela família. Embora não seja o foco deste estudo modelar a posse de autos pelas famílias, compreender o que influencia a motorização é importante porque a motorização influencia diretamente a mobilidade dos indivíduos. Pfeiffer e Strambi (2005), que analisam a evolução da motorização da RMSP entre 1987 e 1997, concluem que, para explicar a posse de autos “*as variáveis de estrutura familiar utilizadas na análise e modelagem da posse de autos perderam importância ao longo do tempo, assim como a própria renda familiar*”. Eles sugerem que isso pode ter de dado pela maior facilidade de financiamentos para aquisição de automóveis por uma família, ou ainda pela evolução das condições do transporte público, assim como da organização espacial da região metropolitana.

Segundo a Tabela 20, percebe-se que a proporção de famílias sem automóvel particular oscilou entre os anos, mantendo ainda assim um tendência de queda entre 1977 e 2007. Oscilação entre os anos também ocorreu na proporção das famílias com dois ou mais automóveis, com tendência de crescimento entre 1977 e 2007. O comportamento mais consistente com os cenários macro econômicos brasileiros foi o das famílias com

Gráfico 5 – Distribuição da variável “FAIXA_REN_FAM”, por ano



um automóvel: tiveram leve queda na posse de um automóvel em 1987 (década de crise econômica), aumento de pouco mais de 10% em 1997 (após a estabilização da moeda em 1994), e aumento tímido (0,3%) em 2007.

As *dummies* relativas à presença de automóveis na famílias foram divididas em: (i) presença de 1 automóvel e (ii) presença de 2 ou mais automóveis. Segmentando as famílias segundo o sexo da pessoa responsável e analisando as frequências relativas da posse de auto através dessas *dummies*, obtém-se o Gráfico 6. Nele, observa-se que as variações das taxas de motorização das famílias chefiadas por homens ou por mulheres são semelhantes, porém, as famílias da amostra com 2 ou mais automóveis, cujos responsáveis são homens, tiveram uma leve queda de 2 pontos percentuais. Ademais, as famílias chefiadas por mulheres apresentam taxas de motorização ligeiramente mais altas (2%) do que aquelas chefiadas por homens.

A quantidade de motocicletas (**QT_MOTO**) e bicicletas (**QT_BICI**) foram levantadas apenas na Pesquisa OD de 2007, e tem suas estatísticas descritivas são apresentadas na Tabela 21. Percebe-se que, embora um meio de transporte individual motorizado

Tabela 18 – Venda interna de veículos no Brasil entre 1960 e 2009

Ano	Autos	Total	Fator de crescimento (total)
1960	40.980	131.499	1
1970	308.024	416.704	3,2
1980	739.028	980.261	7,5
1990	532.906	712.741	5,4
2000	1.176.774	1.489.481	11,3
2009	2.474.764	3.141.240	23,9

Fonte: Adaptado de ([VASCONCELLOS, 2012](#), p.29)

Tabela 19 – Estatísticas da variável “QT_AUTO”

ANO	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
1977	0	0	0	1	7
1987	0	0	0	1	9
1997	0	0	1	1	9
2007	0	0	1	1	8
Geral	0	0	0	1	9

ANO	Média	Desvio Padrão	Assimetria	Curtose	Nº de “NA”
1977	0,65	0,81	1,36	2,28	0
1987	0,57	0,77	1,59	4,09	0
1997	0,72	0,89	1,57	3,78	0
2007	0,78	0,86	1,19	1,99	0
Geral	0,68	0,84	1,43	3,03	0

mais barato, a incidência da posse de motocicletas é uma fenômeno mais raro, provavelmente devido ao risco associado a esse meio de transporte ⁴.

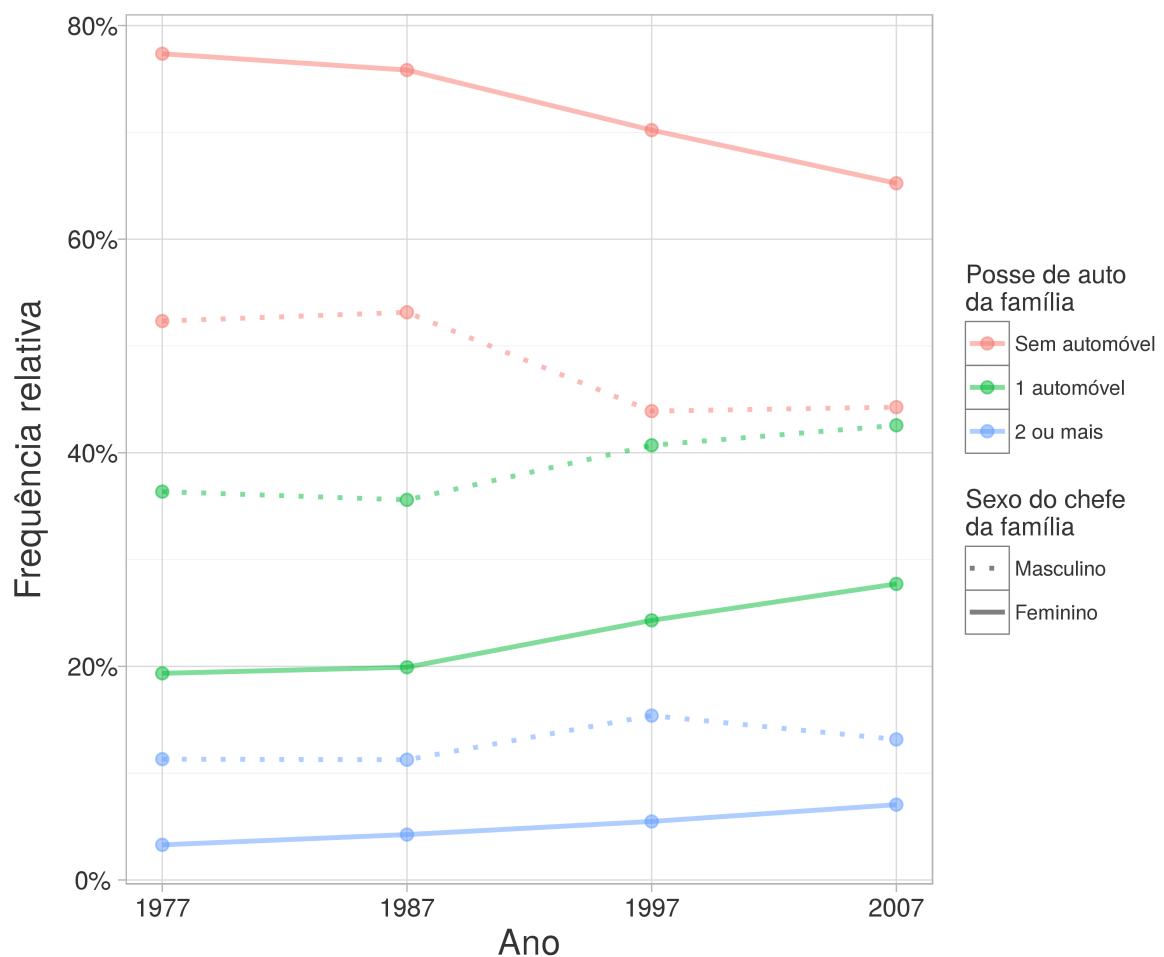
Essa falta de segurança na condução do meio de transporte também ocorre com a bicicleta, porém, além de ser ainda mais barata que a motocicleta, ela não é motorizada e, por isso, comumente trafega nos passeios públicos e calçadas, onde o(a) condutor(a) sofre menor risco de colisão com meios motorizados de transporte (motocicletas, carros, ônibus e caminhões). Com a recente política municipal de incentivo à bicicleta como meio de transporte, por meio de investimentos em infraestrutura cicloviária, é possível que

⁴ Segundo dados da CET o índice de acidentes com motociclistas...

Tabela 20 – Proporção das famílias segundo posse de automóveis

ANO	1977	1987	1997	2007
% de famílias sem auto	55,7	56,9	49,4	51,3
% de famílias com 1 auto	34,1	33,0	37,3	37,6
% de famílias com 2 ou mais autos	10,2	10,1	13,3	11,1

Gráfico 6 – Proporção de famílias com pessoa responsável do sexo feminino e do sexo masculino, segundo posse de automóveis, por ano



na Pesquisa OD de 2017 haja evoluções destes dados que mereçam investigação. Vale lembrar que existe também a questão do status associado muito mais fortemente à posse do automóvel do que da motocicleta; e no caso das bicicletas, seu uso é associado à ideia de pobreza e falta de recursos suficientes para comprar um carro.

A variável **TOT_PESS**, um dado de contagem, indica quantas pessoas existem na família e tem suas estatísticas descritivas apresentadas na Tabela 22. Não haviam *missing values* e o valor médio indica que o tamanho da família diminuiu, passando de

Tabela 21 – Estatísticas das variáveis “QT_MOTO” e “QT_BICI”

	2007	Mínimo	1º Quartil	Mediana	3º Quartil	Máximo
motocicleta	0	0	0	0	9	
bicicleta	0	0	0	1	9	
	ANO	Média	Desvio Padrão	Assimetria	Curtose	Nº de “NA”
motocicleta	0,07	0,29	5,86	74,31	0	
bicicleta	0,46	0,8	2,27	7,5	0	

pouco mais de 4 pessoas em 1977 para pouco menos de 3 pessoas por família em 2007. Em comportamento análogo a TOT_FAM, a Figura 8 indica que a influência dos *outliers* de TOT_PESS diminui com o tempo, o que também se reflete na queda dos desvios padrão, que indica menor dispersão dos dados.

Tabela 22 – Estatísticas da variável “TOT_PESS”

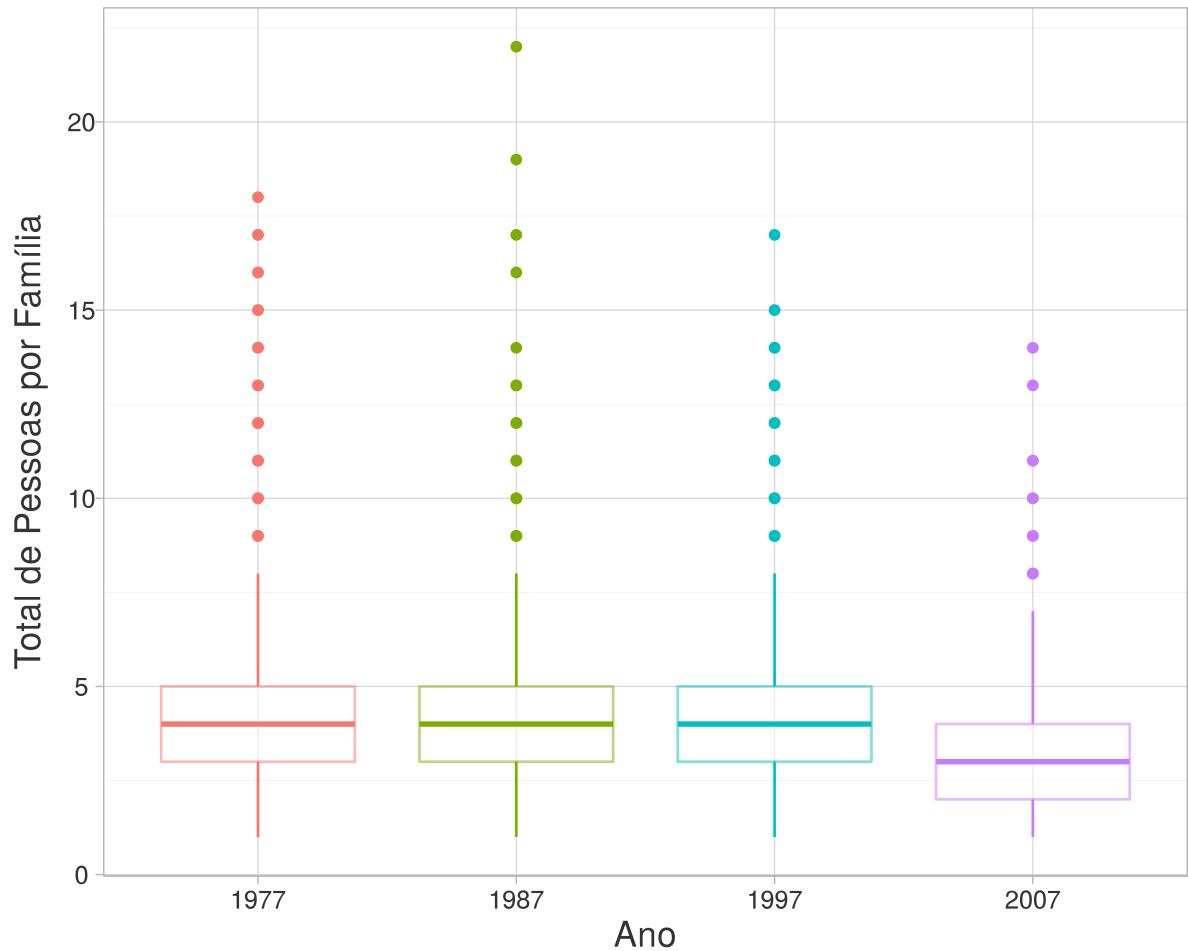
	ANO	Média	Desvio Padrão	Assimetria	Curtose	Máximo
	1977	4,13	2,09	0,99	1,78	18
	1987	3,93	1,80	0,92	2,03	22
	1997	3,68	1,73	0,82	1,68	17
	2007	2,96	1,46	0,78	1,00	14
	Geral	3,65	1,83	1,00	2,10	22

Percebe-se pelo Gráfico 7 a tendência de diminuição geral nas porcentagens de famílias com filhos pequenos (até 9 anos) a partir de 1987, efeito que será percebido na faixa etária seguinte (entre 10 e 19 anos) em 1997. Essa diminuição da presença de dependentes jovens (crianças/adolescentes) mantém-se em 2007. Nos mesmos períodos de análise, ocorre o envelhecimento da população, efeito capturado no gráfico pela presença de idosos (acima de 60 ou de 70 anos) com taxas positivas de crescimento.

Como um dos fatores do indivíduo que influencia seu padrão de deslocamentos está o estágio no “ciclo de vida” (Van de Bilt, 1997). Isto é, as atividades desenvolvidas por uma pessoa depende em que fase da vida ela, e também os demais membros da família, se encontram. A variável **IDADE** é uma das variáveis que definem o ciclo de vida das pessoas. É possível perceber no Gráfico 8 que houve uma transição da pirâmide etária da RMSP, indicando envelhecimento da população tanto masculina como feminina.

Embora não exista relação identitária entre sexo e gênero, conforme já exposto na revisão de literatura, a variável **SEXO** é uma componente importante para compreender a categoria de análise gênero. Percebe-se na Tabela 23 que em todos os anos a proporção

Figura 8 – Box plot da variável “TOT_PESS”, por ano



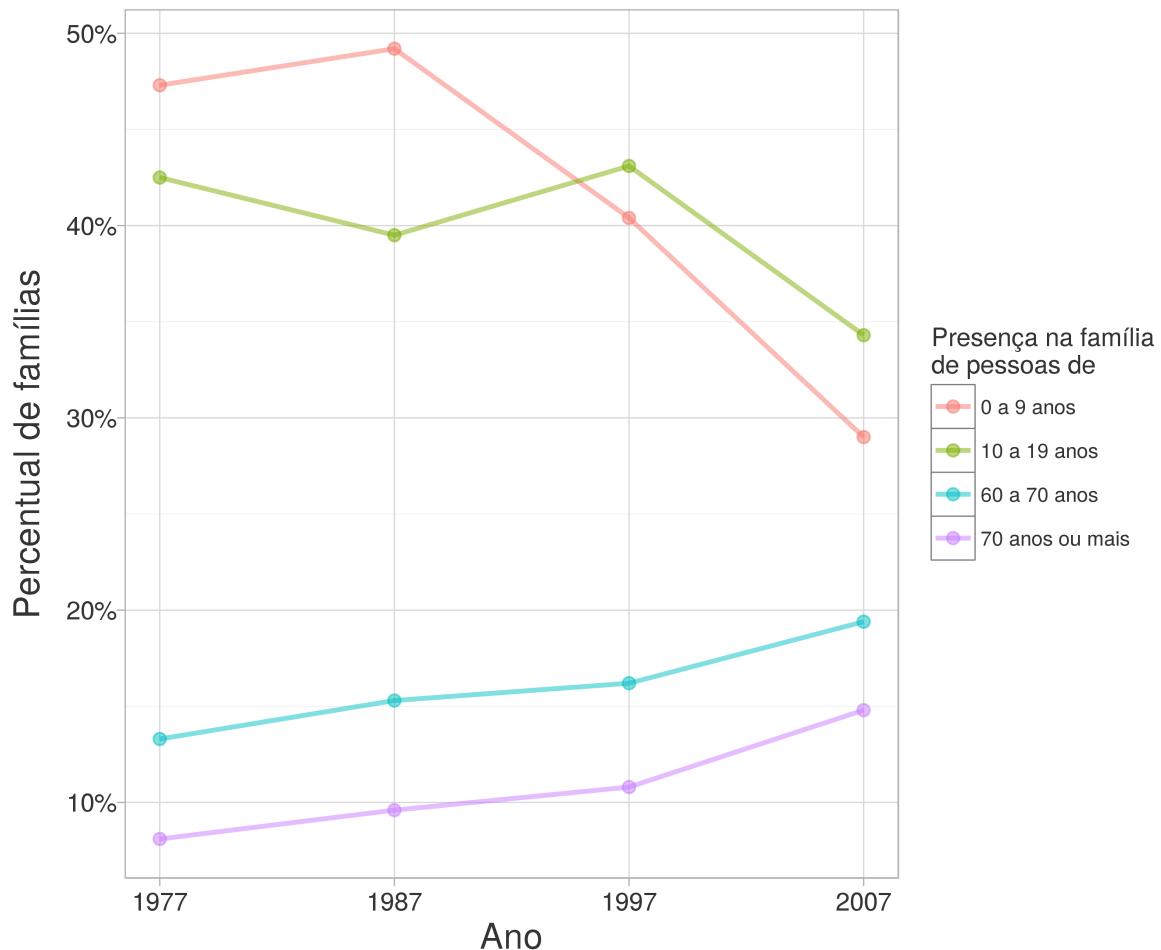
de mulheres entrevistadas é superior à de homens, provavelmente devido ao fato de a pesquisa ser domiciliar e ser mais provável encontrar mulheres do que homens em casa.

Tabela 23 – Frequências absoluta e relativa da variável SEXO, por ano

ANO	1977	1987	1997	2007
quantidade de pessoas do sexo masculino na amostra	52.162	53.176	47.326	42.289
% de pessoas do sexo masculino	48,3	48,0	47,9	46,3
quantidade de pessoas do sexo feminino na amostra	55.866	57.637	51.454	49.116
% de pessoas do sexo feminino	51,7	52,0	52,1	53,7

Observando a variável situação familiar (**SIT_FAM**), nota-se que para as mulheres houve uma mudança ao longo dessas três décadas - ver Gráfico 9. Em 1977, era mais frequente elas ocuparem a posição de filhas (41,8%), em seguida de cônjuges (36,3%). A posição de “pessoa responsável” pela família é a quarta categoria mais frequente

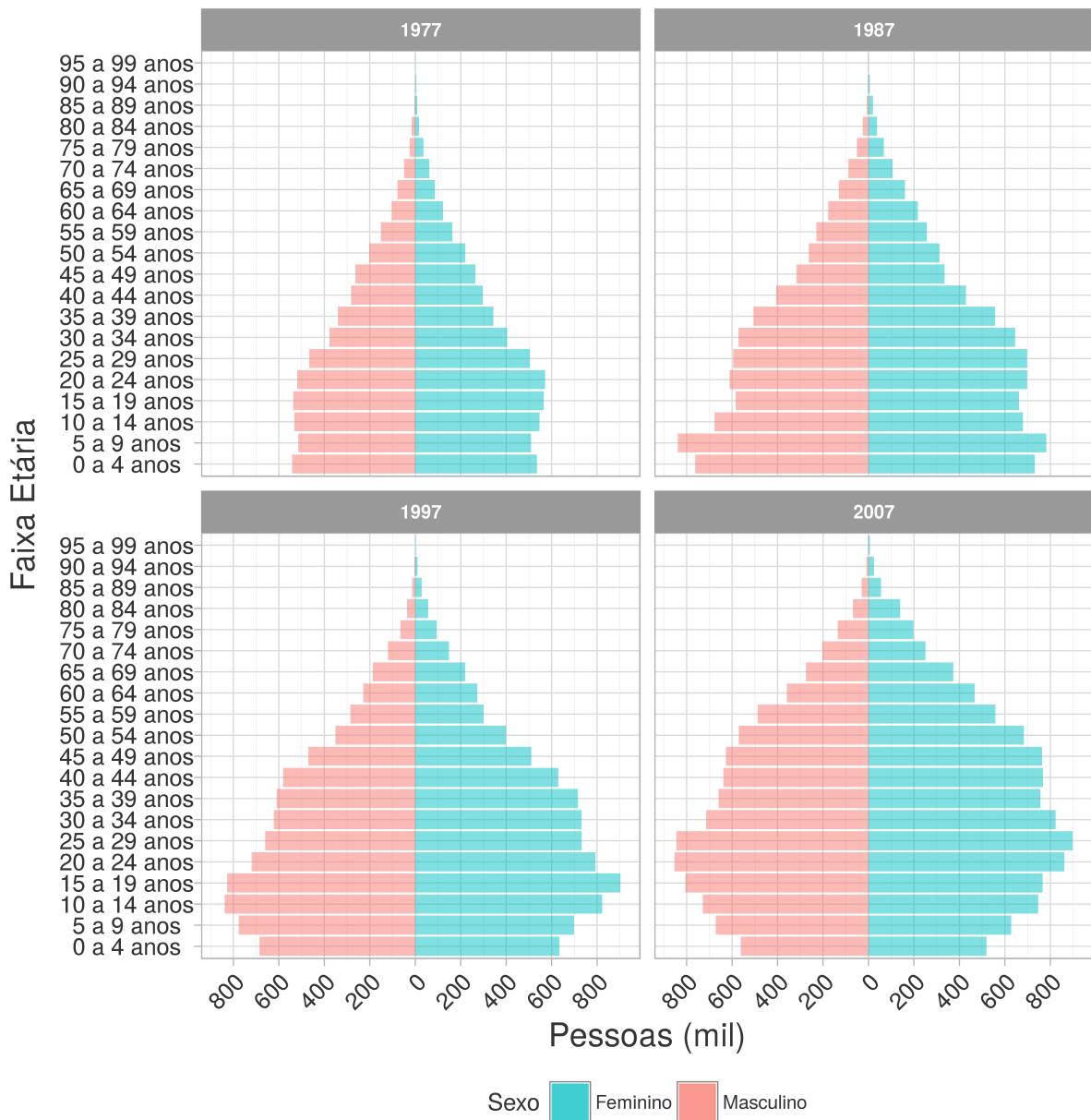
Gráfico 7 – Proporção de famílias com presença de dependentes, por ano



(6,3%), de seis. Tal distribuição permanece semelhante em 1987. Em 1997, no entanto, a posição de “pessoa responsável” pela família (11,1%) já quase se equipara à posição de “outro parente/agregado(a)” (11,3%). Em 2007, se aproxima de um quarto proporção das mulheres entrevistadas que são responsáveis por suas famílias (22,6%), representando aumento demais de 3,5 vezes em relação aos percentuais de 1977. O percentual de mulheres cônjuges/companheiras pouco se altera ao longo do tempo, permanecendo na faixa dos 35%. Há diminuição da posição de empregado(a) doméstico(a) para as mulheres (da ordem da metade). Existe, também, queda da frequência daqueles que se declaram na posição de filho(a) ou enteado(a) tanto para homens como para mulheres - em ordem de grandeza próxima: cerca de 10% para mulheres e 9% para homens. Isso pode ser reflexo da diminuição das taxas de fecundidade⁵ da população (ver Tabela 24). Entre os homens percebe-se que houve crescimento entre aqueles com posição de “pessoas responsável” de

⁵ Por “taxa de fecundidade total” entende-se o número médio de filhos que teria uma mulher de uma coorte hipotética (15 e 49 anos de idade) ao final de seu período reprodutivo. Fonte: IBGE. Disponível em: <<http://www.ibge.gov.br/home/estatistica/populacao/condicaodevida/indicadoresminimos/conceitos.shtml#tf>>

Gráfico 8 – Distribuição da variável “IDADE” de respondentes, por ano e por sexo



cerca de 3,5%, e também dos que declaram-se cônjuge/companheiro (cerca de 20 vezes) - esta última constatação é coerente com o fato de mais mulheres serem a principal fonte de renda doméstica, ou seja, a “pessoa responsável” da família.

Tem-se como uma das hipótese deste trabalho que houve evolução dos padrões de mobilidade por gênero. Articular as variáveis sexo e situação familiar é uma melhor estratégia para tentar utilizar o gênero como categoria de análise. Assim, a transformação dos papéis sociais desempenhados por homens e mulheres dentro do núcleo familiar ao longo das últimas décadas pode ter alterado de maneira significativa os respectivos padrões de mobilidade.

Tabela 24 – Evolução das taxas de fecundidade no Brasil, de 1970 a 2010

Ano	1970	1980	1991	2000	2010
Taxa de fecundidade (Brasil)	5,8	4,4	2,7	2,4	1,9
Taxa de fecundidade (Sudeste)	4,6	3,2	2,4	2,1	1,7
Taxa de fecundidade (São Paulo)	3,94	3,24	2,28	2,05	1,67

Fonte: Compilação a partir de dados dos censos do IBGE disponíveis em <<http://seculoxx.ibge.gov.br/populacionais-sociais-politicas-e-culturais/busca-por-palavra-chave/populacao/810-fecundidade>> Acesso em 17 de novembro de 2014

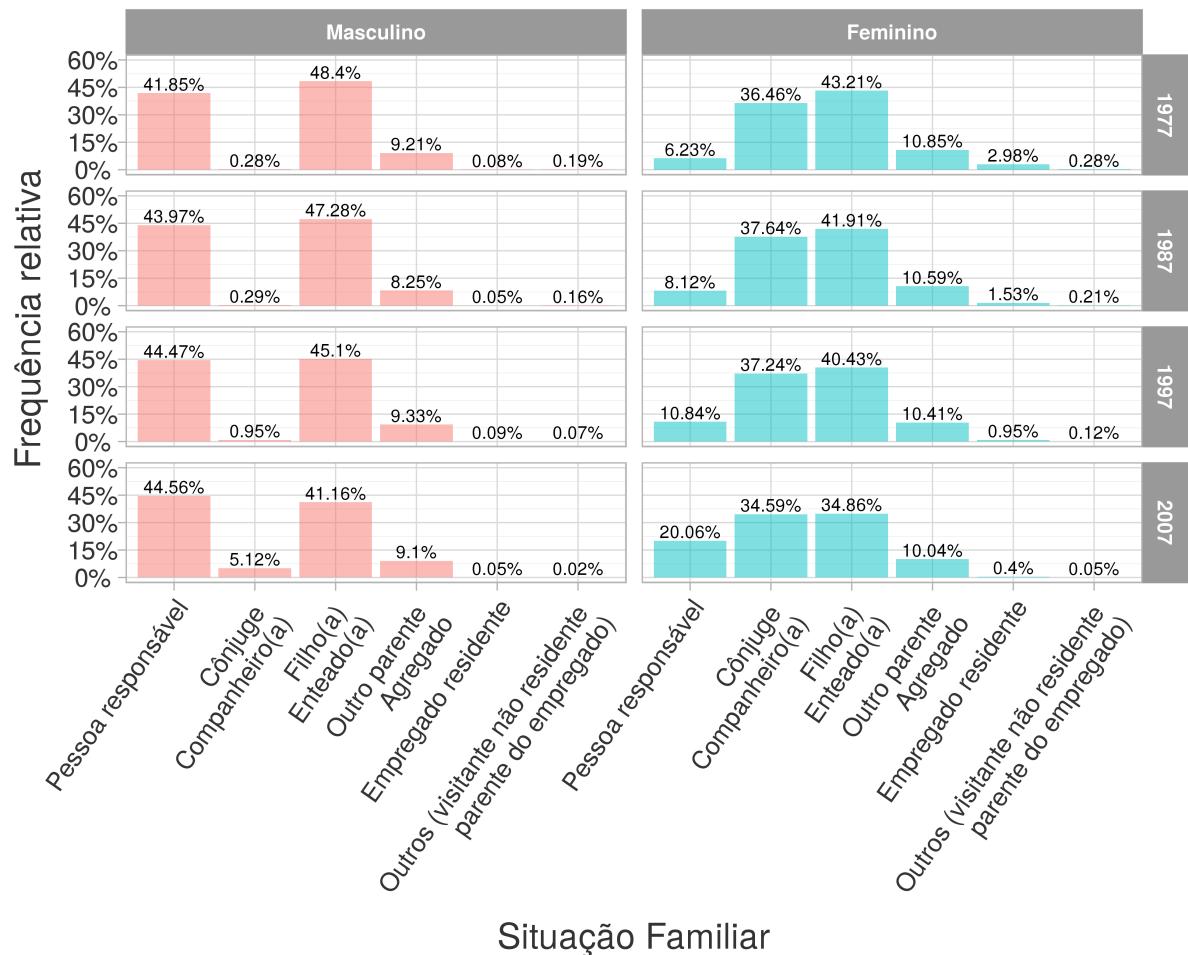
Nota: Ao analisar as taxas de fecundidades para as Grandes Regiões, nota-se que o Sudeste tem os menores percentuais de mulheres que tiveram filhos em todos os subgrupos etários.

Van de Bilt (1997), a partir da análise de trabalhos de diversos autores, para compreender melhor o comportamento das pessoas em relação à participação em atividades e a consequente geração de viagens, elenca como conceito importante, além do estágio no ciclo de vida familiar e dos papéis sociais, o estilo de vida. Duas das variáveis que auxiliam a caracterizar o estilo de vida da pessoa respondente é saber se ela estuda e se ela trabalha.

A variável **TRABALHA** foi construída a partir da variável OCUP da seguinte maneira: se a categoria da ocupação fosse 1, correspondente a “tem trabalho”, a variável recebia valor 1, caso contrário, recebia valor 0. A variável **ESTUDA** conta com as categorias sim (1) e não (0). Em 1987, 1997 e 2007 esta variável existe, mas somente com a categoria “não” em comum. Assim, as demais alternativas de resposta tornaram-se simplesmente “sim”, independente das subdivisões que apresentassem. Para 1977, ano em que essa variável não existe, ela foi preenchida segundo o seguinte critério: a pessoa foi considerada estudante se o campo “Zona da escola” dela não fosse vazio ou igual a zero. Preferiu-se não utilizar a categoria “estudante” da variável ocupação para não perder a informação de quem estuda e trabalha, pois neste caso, estudante não seria a ocupação principal da pessoa e sim “tem trabalho”.

As Tabelas 26 e 25 indicam as frequências das *dummies* TRABALHA e ESTUDA, por ano e por sexo, em relação à população (aplicados os fatores de expansão). Percebe-se que o percentual de trabalhadores(as) aumenta na população devido à maior participação feminina no mercado de trabalho, dado que os percentuais dos trabalhadores permanecem no mesmo patamar (24% da população). Segmentando por sexo, vê-se que houve um aumento de quase 20% na participação feminina no mercado de trabalho e pouco mais de 2% de incremento na masculina. Analisando as proporções de mulheres e homens estudantes, nota-se que houve pequeno crescimento entre 1977 e 1997 (3,2% feminino e 2,5% masculino) seguido de leve queda (1,5% pra ambos sexos). A expectativa era de ter havido um grande crescimento percentual do número de estudantes, especialmente

Gráfico 9 – Distribuição da variável “SIT_FAM”, por ano e por sexo



mulheres, para o período observado. O crescimento tímido pode dever-se ao fato de que, embora a população da RMSP tenha elevado seus níveis de escolarização, o crescimento (desacelerando) e o envelhecimento (em ascensão) da população implicam haver mais gente fora da faixa etária escolar do que dentro dela. As quedas no percentual de estudantes pode estar relacionada tanto ao envelhecimento da população quanto à saturação do sistema de ensino, cuja obrigatoriedade de oferta pública limita-se à Educação Básica (9 anos de Ensino Fundamental e 3 anos de Ensino Médio).

Tabela 25 – Frequências da variável TRABALHA, por ano e sexo

ANO	1977	1987	1997	2007
% de trabalhadoras relativo ao total da população	9,78	12,18	16,27	19,88
% de trabalhadores relativo ao total da população	24,42	24,73	24,61	24,82
% de trabalhadoras relativo ao total de mulheres	19,08	23,44	31,38	37,76
% de trabalhadores relativo ao total de homens	50,12	51,50	51,13	52,42
% de trabalhadores(as) relativo ao total da população	34,20	36,91	40,89	44,70

Tabela 26 – Frequências da variável ESTUDA, por ano e sexo

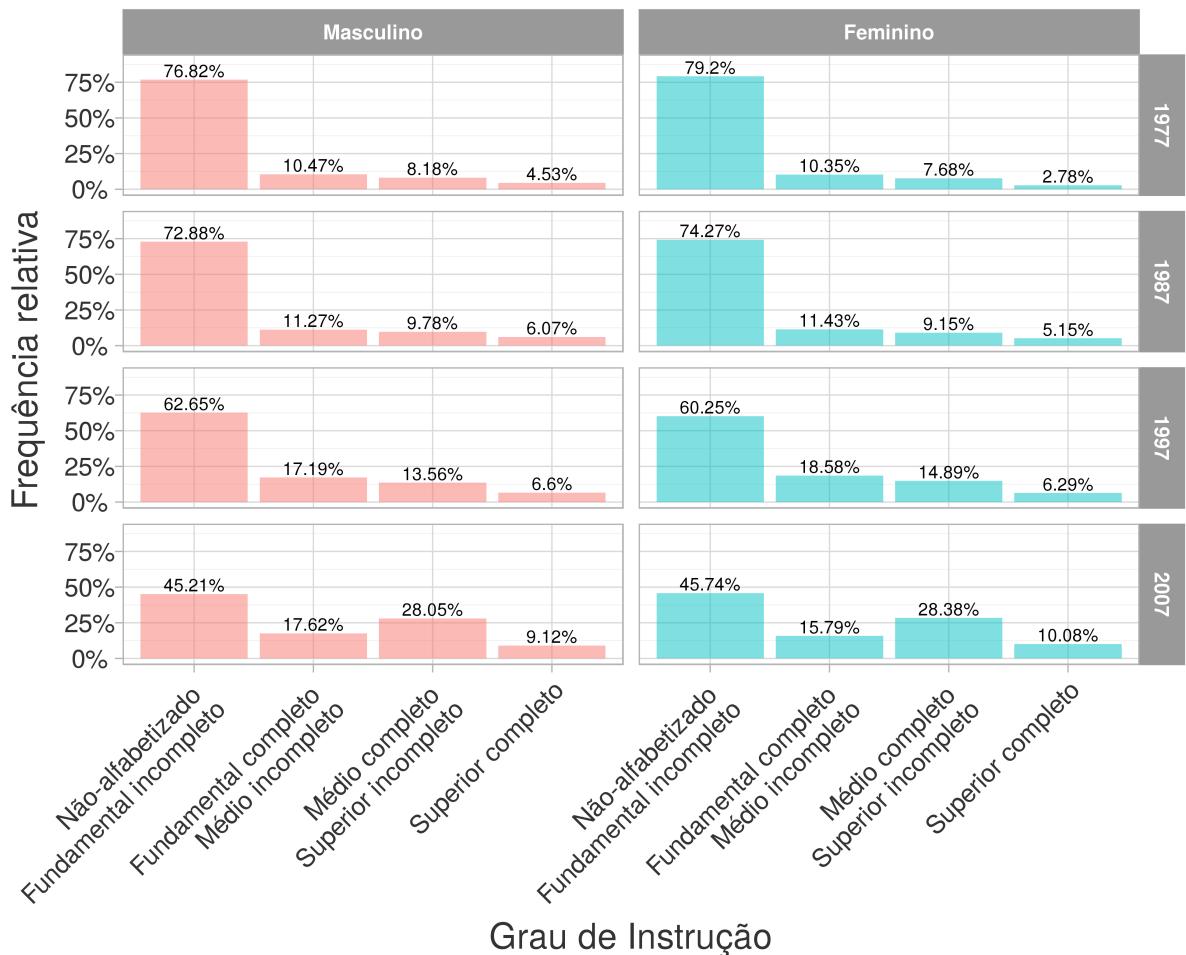
ANO	1977	1987	1997	2007
% de estudantes mulheres relativo ao total da população	11,94	12,88	15,17	13,75
% de estudantes homens relativo ao total da população	12,55	12,92	14,99	13,42
% de estudantes mulheres relativo ao total de mulheres	23,30	24,79	29,25	26,12
% de estudantes homens relativo ao total de homens	25,74	26,90	31,15	28,35
% de estudantes relativo ao total da população	24,49	25,80	30,17	27,18

No Gráfico 10 nota-se que em 1977 tanto homens como mulheres dispunham de pouco tempo de escolaridade - cerca de três quartos da população era analfabeto ou possuía no máximo o fundamental incompleto. Nessa época, nos três níveis de instrução superiores a esse os homens tinham índices maiores que as mulheres. O grau de instrução (**GRAU_INSTR**) da população vai aumentando e em 1987, o grau de escolarização feminino é levemente superior ao masculino nas categorias “fundamental completo / médio incompleto” e “médio completo / superior incompleto”. Na categoria “superior completo” o grau de instrução masculino é um pouco superior, situação que se inverte em 2007. Neste último ano de análise, as mulheres apresentam maiores percentuais nos dois níveis de maior grau de instrução e os homens, nos dois níveis de menor grau de instrução.

A elevação do grau de instrução entre 1977 e 2007 influencia não apenas as viagens motivo trabalho (por eventual aumento da empregabilidade) mas também as viagens motivo escola, realizadas por um contingente de pessoas cada vez maior, mais diverso e contendo mais faixas etárias. A maior participação feminina no mercado de trabalho além de impactar as rendas (individual e familiar) deve influenciar bastante as viagens motivo trabalho.

A renda individual (variável **REN_IND**) tem seu comportamento de médias e medianas análogo ao da renda familiar. Explora-se aqui, então, as rendas individuais de

Gráfico 10 – Distribuição da variável “GRAU_INSTR”, por ano e por sexo



quem tem 10 anos ou mais⁶, segmentando por sexo e e por duas situações familiares: pessoa responsável e cônjuge/companheiro(a). Em todos anos, a renda individual média masculina é superior à feminina para a mesma situação familiar, conforme Tabela 27. Em 1977, quando pessoa responsável pela família, o homem ganhava 2,5 vezes mais que a mulher em mesma condição. Essa marca vem caindo (cada vez mais devagar) até que, em 2007, eles ganham 1,5 vezes mais do que elas. Para a situação de cônjuges, a relação entre as rendas médias masculina e feminina giram em torno de dois, oscilando para 1,8 em 1987 e 2,4 em 1997. Ou seja, homens cônjuges em geral ganham cerca do dobro das mulheres cônjuges e essa situação pouco se alterou com o passar das décadas.

Pelo Gráfico 11 percebe-se que houve um aumento de quem ganhava até 1 salário mínimo entre 1977 e 1987. De 1987 para 2007 a proporção de pessoas que têm rendimentos nessa faixa salarial vem decrescendo. De 1977 a 1997 a faixa de rendimento de 1 a 2

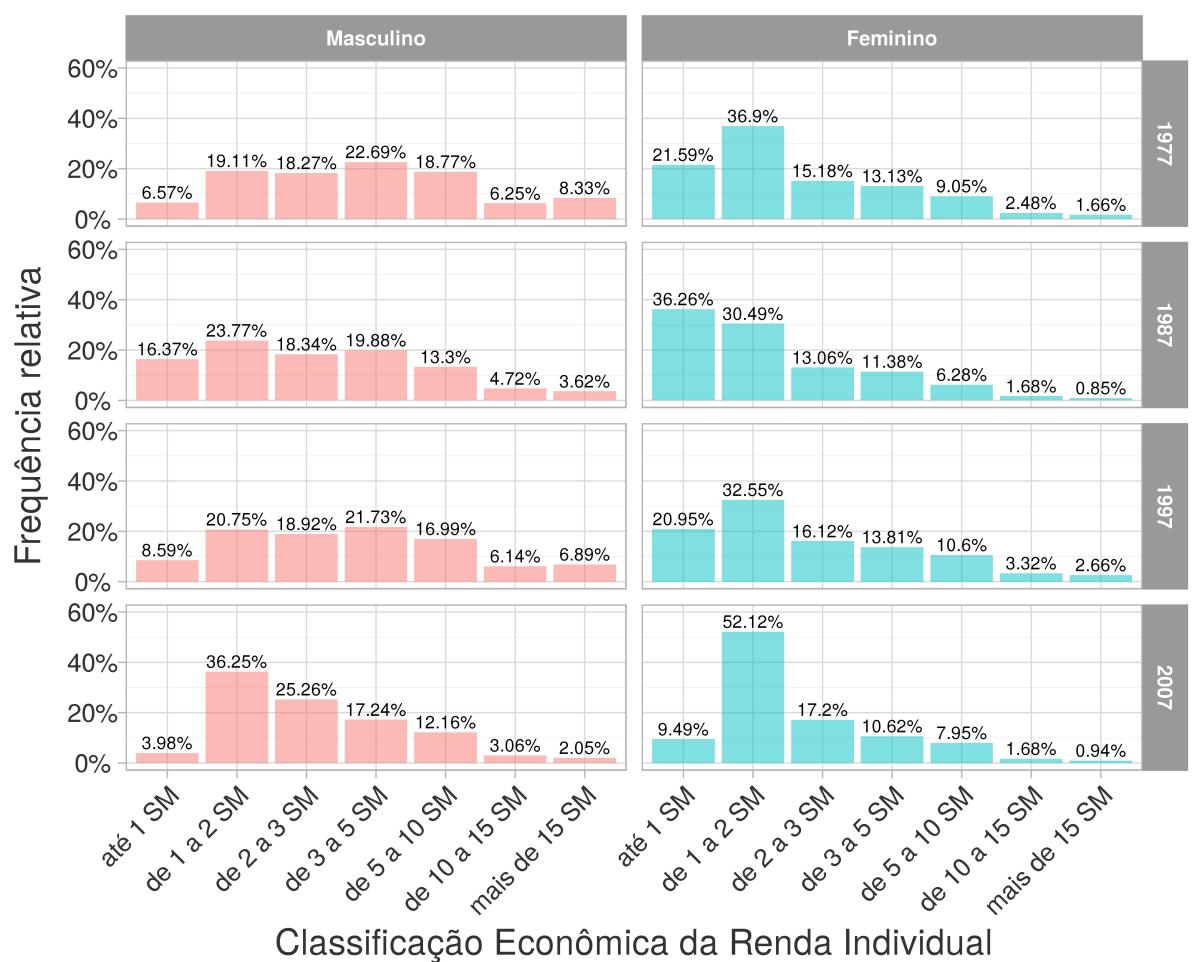
⁶ Foi adotada essa idade como limite de corte porque o IBGE produz estatísticas de “valor do rendimento nominal médio mensal e mediano mensal para pessoas de 10 anos ou mais de idade, total e com rendimento”. Fonte: <<http://www.sidra.ibge.gov.br/bda/tabela/listabl.asp?c=1381&z=cd&o=7>> Acesso em 15 de janeiro de 2016

Tabela 27 – Estatísticas da variável “REN_IND”

Homem		pessoa responsável			cônjuge/companheiro		
ANO	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana	
1977	2.762,85	3.910,71	1.478,21	703,24	1.461,49	0,00	
1987	1.610,76	2.146,33	985,80	522,67	953,85	99,64	
1997	1.769,16	3.272,09	873,61	1.143,30	2.115,36	485,34	
2007	1.265,37	2.330,39	600,00	1.147,71	2.396,79	300,00	
Total	1.869,37	3.059,03	970,68	1.098,29	2.280,25	291,73	
Mulher		pessoa responsável			cônjuge/companheira		
ANO	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana	
1977	1.098,95	2.108,50	506,81	306,95	1.097,29	0,00	
1987	807,79	1.398,68	397,99	284,87	872,94	0,00	
1997	1.034,83	1.923,79	465,93	478,78	1.332,07	0,00	
2007	866,43	1.661,52	380,00	498,39	1.204,76	0,00	
Total	926,53	1.752,55	388,27	383,26	1.131,58	0,00	

salários mínimos ficou próxima dos 25% e, em 2007, cresceu aproximadamente 10%. A faixa de rendimento de 2 a 3 salários mínimos subiu cerca de 3% entre 1977 e 2007, o mesmo que a faixa de rendimentos de 3 a 5 salários mínimos diminuiu no período.

Gráfico 11 – Distribuição da variável “FAIXA_REN_IND”, por ano



6.2 Análises Preliminares

O grupo de análises que se segue busca compreender como é o comportamento das pessoas e das famílias em termo de viagens realizadas, em cada ano e diferencialmente entre os anos, olhando para tanto variáveis como:

- número de viagens;
- modos de viagem;
- motivos de viagens;
- duração das viagens;
- distâncias das viagens.

A variável **TOT_VIAG** representa o número total de viagens realizadas pela pessoa. Não existem *missing values* neste campo, os valores mínimos para todos anos e ambos sexo são 0, bem como também são 0 os valores do primeiro quartil (25%). Os valores das demais estatísticas (considerando os fatores de expansão para a população) estão apresentados na Tabela 28. Conforme já era de se esperar, para quem faz viagem no dia da pesquisa (número de viagem é não nulo) existe a predominância do valor 2, ou seja, são pessoas que saem de suas residências com um propósito único (trabalhar, estudar, fazer compras) e depois retornam à residência após a atividade. Percebe-se que, independente do sexo, o número médio de viagens por pessoa em relação a 1977 caiu um pouco em 1987 e 1997 (de 1,67 para 1,64) e subiu novamente em 2007 (para 1,70). Os desvios padrão caíram ao longo do tempo, indicando tendência de menor dispersão dos dados. Os valores de assimetria são positivos, indicando maior concentração à esquerda e cauda longa à direita da distribuição. Os valores de curtose evidenciam não tratar-se de distribuição normal.

Analizando esses dados segmentados por sexo, vê-se que as medianas são iguais. O número médio e máximo de viagens para mulheres é sempre inferior ao dos homens, para o mesmo ano. Os valores de assimetria para o sexo feminino e o masculino são positivos e convergem para o valor geral com o passar das décadas. A diferença entre o número médio de viagens de mulheres e homens vem diminuindo.

Ao considerar apenas quem faz viagens, os valores mínimos para todos anos e ambos sexo passam para 1, bem como também são 1 os valores do primeiro quartil (25%). Os valores das demais estatísticas (considerando os fatores de expansão para a população) estão apresentados na Tabela 29. Para todos anos e para ambos sexos, as médias passaram para valores superiores a 2 e há uma tendência de diminuição do número de viagens por pessoa com o passar do tempo. Os desvios padrão caíram ao longo do tempo para as mulheres e para o conjunto de homens e mulheres, indicando menor dispersão dos dados. Os valores de assimetria aumentaram e continuaram positivos, indicando maior concentração à esquerda e cauda longa à direita da distribuição. Os valores de curtose

Tabela 28 – Estatísticas da variável “TOT_VIAG”, por ano e por sexo

Geral						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	1,67	1,75	2,00	24	1,48	4,71
1987	1,64	1,61	2,00	24	1,37	4,80
1997	1,64	1,60	2,00	21	1,36	4,38
2007	1,70	1,48	2,00	18	1,09	3,53
Sexo Feminino						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	1,35	1,58	2,00	16	1,35	3,16
1987	1,42	1,58	2,00	19	1,40	4,14
1997	1,52	1,62	2,00	18	1,48	5,00
2007	1,61	1,51	2,00	17	1,10	2,88
Sexo Masculino						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	2,01	1,85	2,00	24	1,52	5,31
1987	1,88	1,61	2,00	24	1,41	5,83
1997	1,78	1,58	2,00	21	1,26	3,87
2007	1,81	1,44	2,00	18	1,12	4,48

também aumentaram e a distribuição continua não sendo aderente à normalidade. Na segmentação por sexo, as medianas e a quantidade máxima de viagens permanecem iguais. Entretanto, o número médio de viagens para mulheres era superior ao dos homens em 1977 e 1987. Já em 1997 e em 2007, a média delas passa a ser inferior à deles.

Foram feitos testes t para avaliar se as médias de mulheres e de homens eram estatisticamente diferentes, em cada ano, tanto considerando quem não fez viagem ($TOT_VIAG=0$), com desconsiderando esse caso. Os p-valores resultantes foram todos inferiores a 0,05, logo, rejeitou-se a hipótese nula de que as médias seriam iguais (nível de confiança de 95%). Foram feitos testes t para avaliar se as médias entre os anos eram diferentes, para o grupo de mulheres, , tanto considerando quem não fez viagem ($TOT_VIAG=0$), com desconsiderando esse caso. Também aqui os p-valores resultantes foram todos inferiores a 0,05, logo, rejeitou-se a hipótese nula de que as médias seriam iguais (nível de confiança de 95%). Portanto, ao considerar o efeito de quem não sai de casa, o número médio de viagens das mulheres sempre é menor que o dos homens, para um dado ano, e realmente houve aumento no número médio de viagens das mulheres, da ordem de 0,1 viagem/ano. Ao desconsiderar o efeito de quem não sai de casa, o número médio de viagens das mulheres

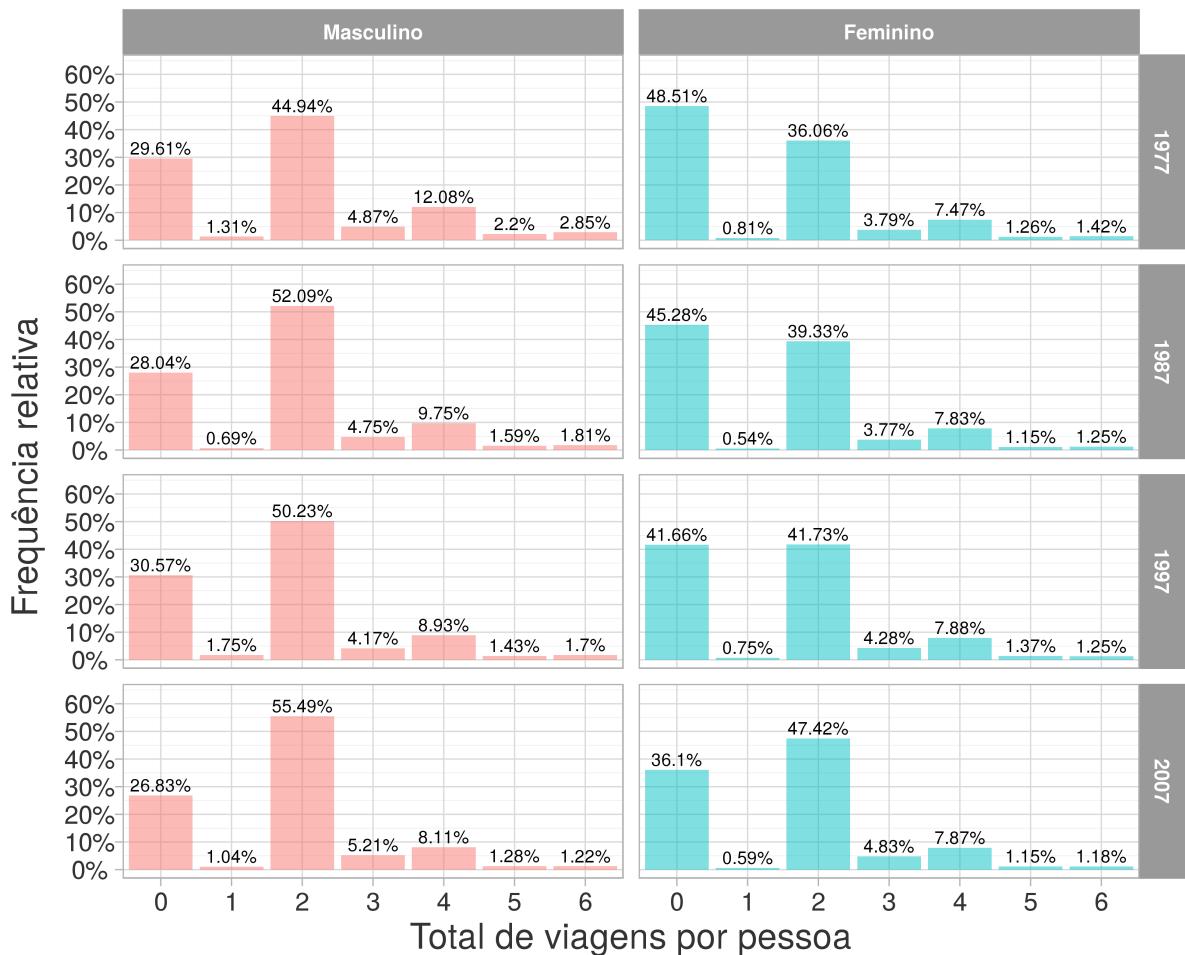
Tabela 29 – Estatísticas da variável “TOT_VIAG”, por ano e por sexo, considerando apenas quem fez viagem

Geral						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	2,75	1,44	2,00	24	2,70	11,75
1987	2,61	1,27	2,00	24	3,01	14,66
1997	2,58	1,27	2,00	21	2,89	12,59
2007	2,49	1,11	2,00	18	2,94	12,77
Sexo Feminino						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	2,85	1,57	2,00	16	2,65	11,19
1987	2,61	1,29	2,00	19	3,08	15,47
1997	2,56	1,26	2,00	18	2,73	10,55
2007	2,47	1,09	2,00	17	3,10	15,08
Sexo Masculino						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	2,62	1,23	2,00	24	2,55	9,84
1987	2,60	1,24	2,00	24	2,90	13,40
1997	2,59	1,29	2,00	21	3,06	14,56
2007	2,52	1,13	2,00	18	2,79	10,71

tornou-se menor que o dos homens em 1997 e o número médio de viagens de homens e mulheres cai com o tempo.

O Gráfico 12 apresenta a distribuição de viagens (considerando os fatores de expansão para a população) até o limite de 6 viagens, corte feito apenas para melhorar a visualização do gráfico, já que a cauda é bastante longa. Deste gráfico, vale destacar a relação entre as viagens nulas (quem não sai de casa) e as viagens de ida e volta (valores iguais a 2). Para homens, o número de viagens nulo é menos frequente que o número de viagens de valor 2 para todos anos de análise. Já para as mulheres, em 1977 as viagens nulas eram a maioria, indicando certa fixitude delas na residência. Essa porcentagem vai diminuindo e a porcentagem no número de viagens igual a 2 vai crescendo, ficam próximas em 1997 e, em 2007, inverte-se a situação observada em 1977. Provavelmente devido à maior participação no mercado de trabalho, as mulheres ganharam mobilidade, restringindo-se menos ao espaço doméstico.

Gráfico 12 – Distribuição da variável “TOT_VIAG” por ano e por sexo



Ao focar no agregado da família, os valores das principais estatísticas (considerando os fatores de expansão para a população) da variável **FAM_VIAG_TOT** estão apresentados na Tabela 30. O número médio de viagens da família com ao longo do tempo, comportamento consistente tanto com a queda do número médio de viagens por pessoa (Tabelas 28 e 29) quanto com a diminuição do tamanho da família (Tabela 22). Aqui também os desvios padrão caem ao longo do tempo (tendência de menor dispersão de dados) e os valores de assimetria são positivos (maior concentração à esquerda e cauda longa à direita da distribuição). Os valores de curtose indicam que a distribuição não é normal. Ao retirar aquelas famílias com total de viagens nulo (ninguém fez viagem), não se observam diferenças nas tendências das estatísticas, à exceção da mediana que em 1997 era 7 em 1997 e passa para 6 ao inserir famílias cujo total de viagens é zero.

Tabela 30 – Estatísticas da variável “FAM_VIAG_TOT”, por ano

Considerando famílias em que há pela menos viagem						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	8,97	6,00	8,00	59	1,36	2,90
1987	8,04	5,14	7,00	50	1,39	3,43
1997	7,68	4,80	7,00	45	1,29	2,85
2007	7,11	4,31	6,00	68	1,52	6,84
Considerando inclusive família em que ninguém fez viagem						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	8,58	6,14	8,00	59	1,27	2,62
1987	7,71	5,28	7,00	50	1,27	3,05
1997	7,22	5,00	6,00	45	1,14	2,39
2007	6,62	4,53	6,00	68	1,28	5,38

Nos campos MODO1, MODO2, MODO3 e MODO4 a categoria “ônibus de linha” inclui as categorias originais “ônibus trólebus”, “trólebus”, “ônibus diesel”, “ônibus”, “ônibus município de São Paulo”, “ônibus outros municípios” e “ônibus metropolitano”. A categoria “ônibus escolar/empresa” inclui também as categorias originais “ônibus fretado”, “escolar”, “transporte escolar”. A categoria “lotação/van” inclui as categorias originais “lotação/perua”, “microônibus/van município de São Paulo”, “microônibus/van outros municípios” e “microônibus/van metropolitano”. Vale destacar que para os anos de 1977 e 1987 foram levantados no máximo três modos, e para os anos 1997 e 2007, no máximo quatro modos para cada viagem.

Na Tabela 31 foram agrupados em “alta capacidade” os modos metroferroviários (metrô e trem), em “ônibus” todos os tipos de ônibus (de linha, escolar, de empresas, lotações e vans), em “passageiro de automóvel” os passageiros de automóvel particular e também de táxis, em “Outros” as viagens realizadas por motocicletas e bicicletas pois estas foram diagnosticadas apenas para 2007. Nesta tabela estão apresentadas as frequências relativas destes agrupamentos para o **MODO1** (primeiro modo utilizado na viagem) e também para o **MODO2** (segundo modo utilizado), buscando avaliar a divisão modal por ano, para quem realizou viagem. Assim, para o primeiro modo o total soma 100% em todos anos, mas o total por ano do segundo modo em diante não necessariamente, porque nem todas viagens utilizaram mais de um modo.

Em relação à utilização do automóvel, sua proporção sobe cerca de 5% de 1977 até 1997 e cai pouco mais de 1% em 2007, talvez por conta dos congestionamentos cada vez mais frequentes e da evolução do sistema de transporte coletivo público da RMSP. Quando

Tabela 31 – Frequência relativa das variáveis “MODO1” e “MODO2”, por ano

MODO 1		Alta		Dirigindo	Passageiro			
ANO	Capacidade	Ônibus	Automóvel	de Automóvel	A pé	Outros	Total	
1977	3,0%	41,7%	15,4%	11,3%	28,0%	0,8%	100%	
1987	4,7%	30,5%	17,2%	9,7%	36,2%	1,7%	100%	
1997	6,4%	28,9%	20,5%	10,8%	34,4%	1,3%	100%	
2007	7,0%	31,8%	19,2%	8,6%	33,1%	2,9%	100%	
MODO 2		Alta		Dirigindo	Passageiro			
ANO	Capacidade	Ônibus	Automóvel	de Automóvel	A pé	Outros	Total	
1977	2,0%	7,0%	0,1%	0,2%	-	0,0%	9,2%	
1987	3,6%	6,5%	0,1%	0,1%	-	0,0%	10,3%	
1997	3,5%	5,9%	0,0%	0,1%	-	0,0%	9,5%	
2007	4,2%	8,2%	0,1%	0,1%	-	0,0%	12,5%	

utilizado, o carro é quase sempre o único modo da viagem. Os ônibus têm uma queda de ~ 13% pontos percentuais entre 1977 e 1997, com recuperação de ~ 3% em 2007. Quem deixou de utilizar o ônibus nas primeiras 3 décadas passou a utilizar a caminhada como método de deslocamento (~ 6%), ou o carro (~ 5%) o ainda o transporte coletivo de alta capacidade (~ 3%). Os modos metrô e trem vêm recebendo um incremento de viagens década a década, saindo de 3% em 1977 para 7% em 2007. A contribuição baixa deste modo, apesar de sua alta capacidade, deve-se provavelmente à cobertura insuficiente de pouca capilaridade no tecido urbano. Vale ressaltar que “alta capacidade” é o único modo que tem sua utilização ainda muito presente como segundo modo (relação MODO1/MODO2 ~ 1,5) - as viagens de ônibus são reduzidas pelo menos da ordem de um quarto, as de carro e outros caem a quase 0% no segundo modo. Isto significa que as viagens com metrô e trem são mais frequentemente precedidas de viagens com outro modo (principalmente ônibus), funcionando como tronco numa lógica de sistema tronco-alimentador. As viagens a pé aumentaram percentualmente entre 1977 e 1997 e caíram em 2007, talvez pelas distâncias necessárias de viagem dado o crescimento da área metropolitana de São Paulo. Conforme o conceito de viagem a pé anteriormente exposto, não haverá modo a pé nos modos 2, 3 ou 4. As tabelas de contingência dos modos 3 e 4 não serão apresentadas por serem pouco significativos no contexto geral: o modo 3 varia de 0,9% (em 1977) a 2,8% (2007) do total de viagens, sendo predominante o modo ônibus; o modo 4, só existente em 1997 e 2007, corresponde ao total de 0,4% das viagens realizadas.

Os Gráficos 13 e 14 apresentam a segmentação por sexo dos modos 1 e 2 de viagem, respectivamente. No primeiro modo de viagem, as viagens a pé são o modo mais frequente para as mulheres em 1987, 1997 e 2007, somente em 1977 o ônibus era mais frequentemente

utilizado por elas. Para os homens o modo mais frequente em 1977 e 2007 é o ônibus, e em 1987 e 1997, a pé. O modo outros é o menos frequente em 1977, 1987 e 1997 para ambos sexos; em 2007, eles superam a alta capacidade, provavelmente porque a expressividade do transporte por motocicleta e bicicleta passou a ser mais expressiva e foi incluída na categoria outros. Comparando ambos sexos por categoria do primeiro modo:

- (i) As mulheres sempre fizeram mais viagens a pé que os homens: a diferença de 6,7 pontos percentuais de 1977 aumenta para 11,3 em 1987, cai para 7,9 em 1997 e sobe novamente em 2007 para 8,8%.
- (ii) A utilização feminina do ônibus é quase sempre superior à masculina, exceto em 1987, ano em que as proporções praticamente se igualam e desde quando a diferença vem aumentando com o tempo.
- (iii) É na utilização do automóvel em que residem as diferenças mais gritantes entre os gêneros: homens predominantemente motoristas e mulheres, passageiras. O destaque aqui reside no fato que entre 1997 e 2007, dentro do grupo de mulheres, elas passaram a dirigir mais do que ser passageiras de automóvel.
- (iv) O transporte de alta capacidade gira em torno de 2,5% a 5,5% do *share* modal, com o homem tendo uma utilização um pouco mais frequente dentro do seu grupo do que as mulheres, mais devido ao trem (cujos percentuais dos homens são sempre superiores aos das mulheres) do que ao Metrô (onde os percentuais das mulheres supera o dos homens a partir de 1997).

Analizando agora as categorias do segundo modo, por sexo:

- (i) As duas categorias que apresentam relevância como segundo modo de transporte equivalem aos modos coletivos ônibus e alta capacidade (metrô e trem), os demais modos ou não foram considerados (por exemplo, a pé) ou não são muito significativos (automóvel e outros).
- (ii) A utilização feminina do ônibus, como segundo modo da viagem, é inferior à masculina entre 1977 e 1997, ao contrário do que ocorria com o primeiro modo. Em 2007 a situação se altera, quando pessoas do sexo feminino passam a utilizar mais frequentemente o ônibus.
- (iii) A frequência do uso de alta capacidade é mais expressiva no conjunto do modo 2, embora ainda seja menor que a do ônibus em todos os anos e para ambos sexos.
- (iv) Para as mulheres, de 1977 para 1987, parece ter havido uma troca modal no segundo modo: houve queda de $\sim 1\%$ no uso do ônibus e aumento também de $\sim 1\%$ no uso de alta capacidade. Foi nesse período em que houve a primeira expansão da rede metroferroviária para a zona leste (trecho Sé-Penha).

Gráfico 13 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o primeiro modo da viagem, por ano

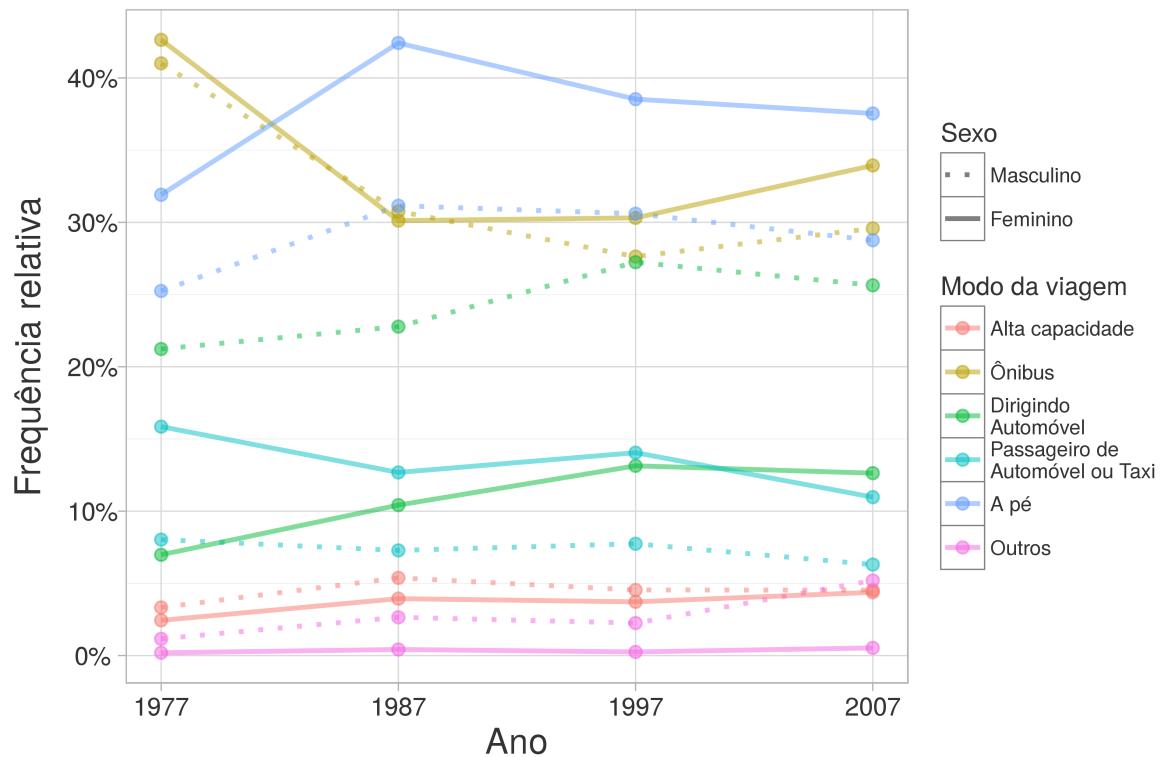
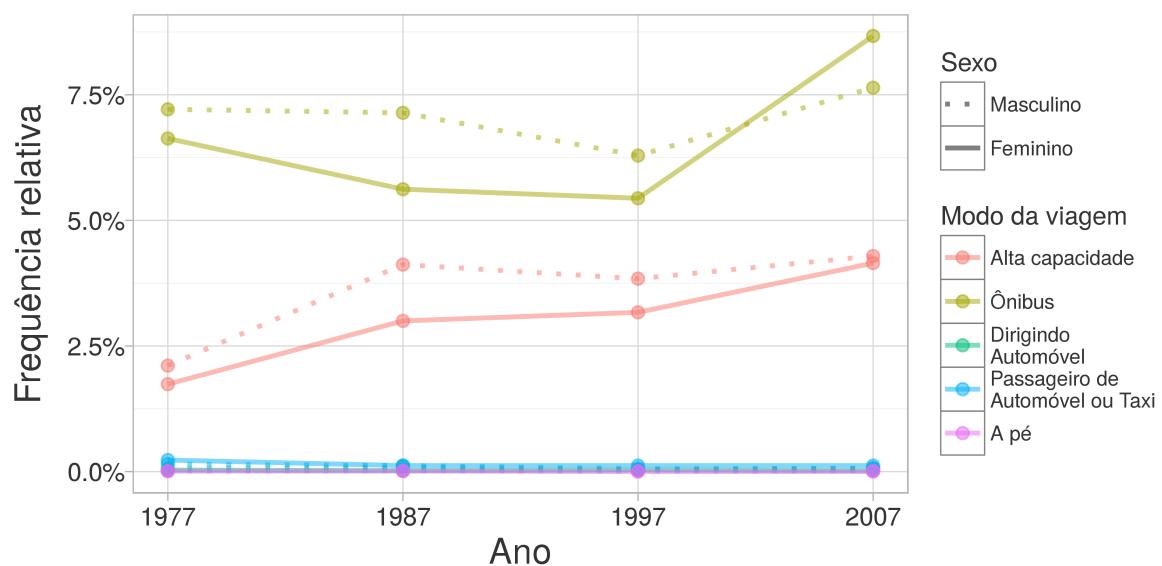


Gráfico 14 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o segundo modo da viagem, por ano



É possível também analisar os modos agregando os modos em “coletivo”, “individual” e “a pé”, o que já fora feito na variável **TIPO_VIAG**, cujas frequências relativas são apresentadas na Tabela 32. O transporte individual cresceu de 1977 a 1997 e recuou um pouco em 2007. O transporte coletivo decresceu entre 1977 e 1987, mas a taxa bem maior que o crescimento do individual, o que significa que essas viagens deixaram de ser feitas de transporte coletivo para, principalmente, serem feitas a pé ou, com menor frequência, de carro. Entre 1987 a 1997 tanto o transporte coletivo como o modo a pé sofrem ligeiras quedas (em torno de 2 pontos percentuais), período em que o transporte individual aumenta sua taxa de crescimento. Neste ano o cenário da divisão modal fica quase equitativamente dividido com cerca de um terço para cada uma das três categorias. Em 2007 a forma de deslocamento a pé sofre ligeira queda ($\sim 1\%$), o transporte individual também cai ($\sim 2,5\%$) e essas viagens passam a ser feitas pelo transporte coletivo que assume proporção um pouco superior à que tinha em 1987.

Tabela 32 – Frequência relativa da variável “**TIPO_VIAG**”, por ano

ANO	Coletivo	Individual	A pé
1977	45,0%	27,0%	28,0%
1987	35,6%	28,2%	36,2%
1997	33,3%	32,3%	34,4%
2007	36,5%	29,5%	33,1%

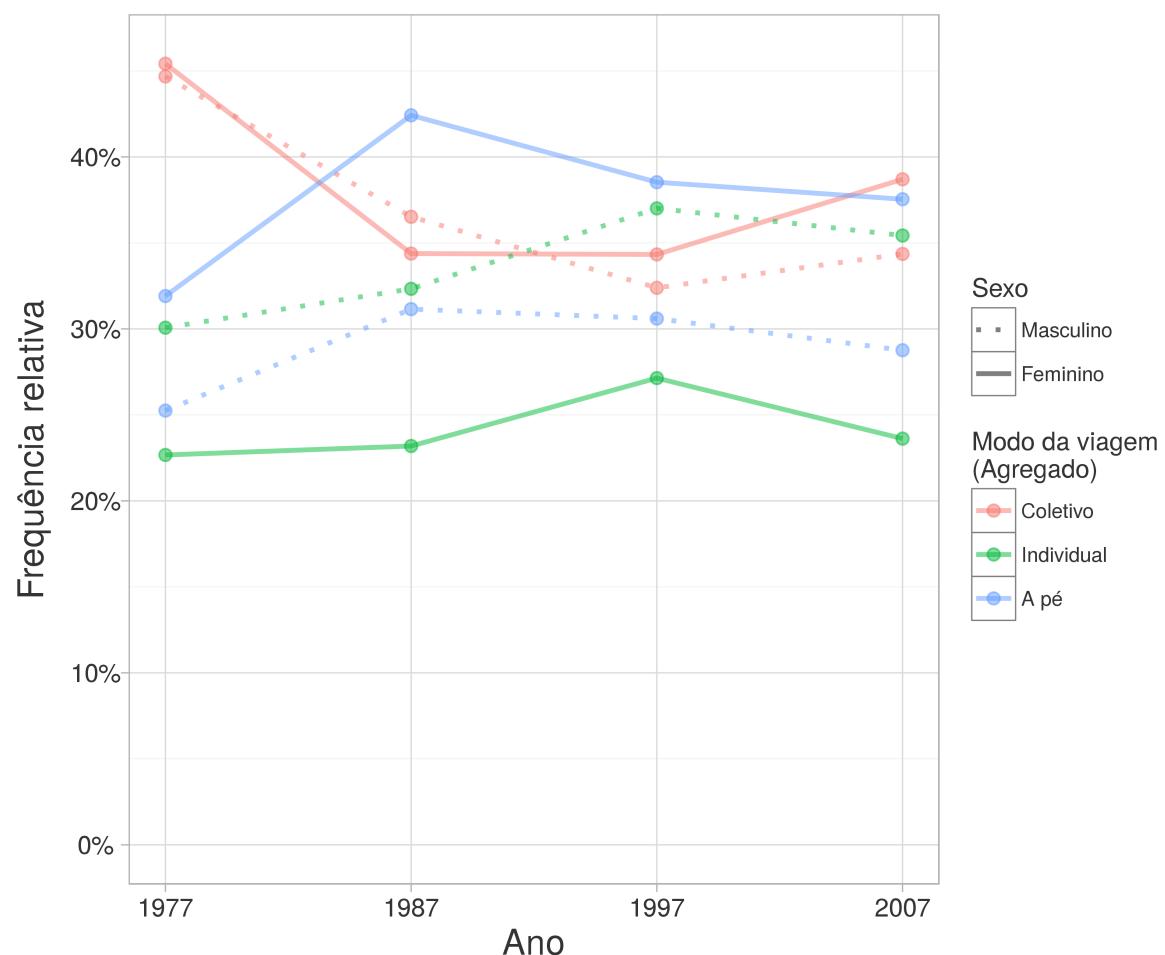
O Gráfico 15 apresentam a segmentação por sexo do modo (agregado) de viagem. Comparando ambos sexos por categoria do primeiro modo:

- (i) Em 1977, 45,4% das mulheres usavam o transporte coletivo, 31,9% deslocavam-se a pé e 22,7% usavam transporte individual. Em 1987, para elas, o transporte individual permanece no mesmo patamar ($\sim 23,2\%$) e ocorre uma migração do coletivo para o a pé com 34,4% e 42,4%, respectivamente.
- (ii) Em 1977, 44,7% dos homens usavam o transporte coletivo, 30,1% o individual e 25,3% deslocavam-se a pé. Em 1987, para eles, o transporte individual cresceu ($\sim 32,3\%$) e as viagens a pé também ($\sim 31,2\%$) indicando uma migração do transporte coletivo (36,5%) para estes modos.
- (iii) Entre 1987 e 1997, a proporção de mulheres a usar o transporte coletivo permanece inalterada, mas ocorre uma migração do modo a pé para o transporte individual.
- (iv) Entre 1987 e 1997, a proporção de homens a usar o transporte individual continua aumentando (para 37%), superando a participação do coletivo (32,4%), enquanto as viagens a pé permanecem no mesmo patamar.
- (v) Entre 1997 e 2007, a proporção do uso feminino do transporte coletivo cresce (para

38,7%) indicando migração para este modo das viagens advindas, especialmente, do transporte individual (que cai para 23,6%) e, em menor medida, do modo a pé (com 37,5%).

(vi) Entre 1997 e 2007, a proporção do uso masculino do transporte coletivo também cresce (para 34,4%) indicando migração para este modo das viagens advindas tanto do transporte individual (que cai para 35,4%) como do modo a pé (com 28,8%).

Gráfico 15 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o modo da viagem (agregado), por ano



Para as variáveis de motivo (**MOTIVO_ORIG** e **MOTIVO_DEST**) foi criada a categoria “servir passageiro”. Para tanto, olhava-se a variável de cada OD “servir passageiro na origem”; caso fosse afirmativo (1), a categoria adotada é “servir passageiro”, porque o que motiva esse deslocamento é o motivo de outrem, não o da pessoa respondente. Caso contrário, adota-se o motivo de origem indicado originalmente na base de dados. Tal procedimento foi realizado com as bases de 1997 e 2007. A base de 1977 já conta com a categoria “servir passageiro”. A base de 1987 é a única que não possui informações suficientes para identificar esse motivo de viagem.

Será explorada a variável motivo no destino porque essa variável indica a atividade fim que gerou o deslocamento. A Tabela 33 foi feita expandindo as viagens com o FE_VIAG, consequentemente consideradas apenas as viagens realizadas (cujo FE_VIAG não foram iguais a zero). Observa-se que o motivo “residência” corresponde à maior parte das viagens (cerca de 45%) realizadas, se alterando pouco ao longo dos anos - resultado próximo ao esperado (pouco menos de 50%) dado que o comportamento de deslocamentos da demanda tem a residência como base, ou seja, é para onde a maior parte das pessoas retornam após executar alguma outra atividade. O segundo motivo mais frequente é “trabalho” girando em torno dos 23,5% e também oscilando pouco (1%) ao longo dos anos, seguido por “educação”, que cresce de 1977 (13,2%) para 1987 (16,9%) e depois decresce em 1997 (14%) e se mantém em 2007 (14%). Assim, trabalho, educação e residência são os motivos de pouco mais de 80% das viagens em todos os anos. A proporção das viagens motivo “manutenção-compras” cresce de 1977 (3,9%) para 1987 (4,5%) e praticamente retorna ao mesmo patamar em 1997 (3,8%), caindo um pouco em 2007 (3,6%). O percentual de viagens motivo “lazer/outros” vem diminuindo com o tempo, cerca de 2 pontos percentuais por década. O percentual de viagens “servir passageiro”, por sua vez, vem aumentando com o tempo, saindo de 1,0% em 1977 para 7,0% em 2007.

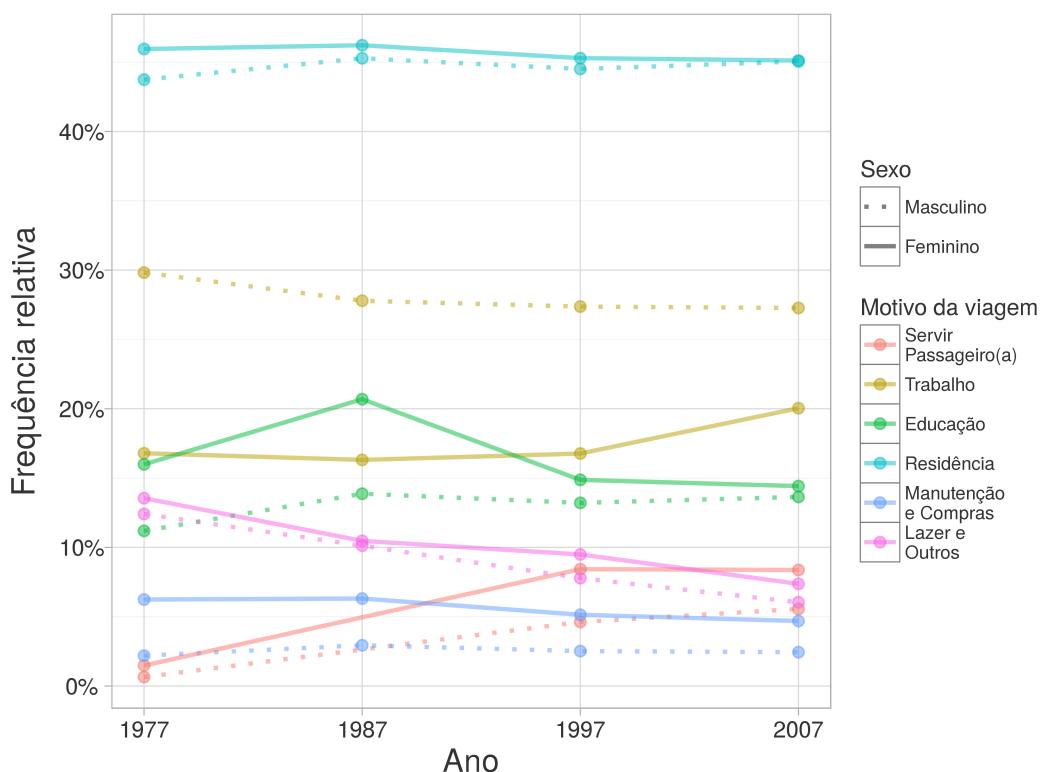
Tabela 33 – Frequência relativa da variável “MOTIVO_DEST”, por ano

ANO	Servir				Manutenção/ compras	Lazer/ Outros
	Passageiro	Trabalho	Educação	Residência		
1977	1,0%	24,4%	13,2%	44,6%	3,9%	12,9%
1987	-	22,6%	16,9%	45,7%	4,5%	10,3%
1997	6,4%	22,3%	14,0%	44,9%	3,8%	8,6%
2007	7,0%	23,7%	14,0%	45,0%	3,6%	6,7%

Ao observar o Gráfico 16, que dentro de cada ano segmenta por sexo os motivos de viagens, percebe-se que as proporções das viagens motivo “trabalho” femininas são sempre inferiores às masculinas e essa diferença vem diminuindo com o tempo por conta da maior participação das mulheres no mercado de trabalho. As proporções das viagens motivo “educação” femininas são sempre superiores às masculinas e essa diferença vem diminuindo e em 2007 essa diferença não chega a 1%. As viagens motivo “lazer / outros” caem para ambos sexos sendo as porcentagens das viagens femininas superiores às masculinas em todos os períodos. As viagens motivo “manutenção / compras” são sempre mais frequentes para mulheres do que para homens e a diferença entre ambos caiu de 4,05 ponto percentual em 1977, quando as mulheres faziam 2,8 vezes mais viagens deste tipo do que os homens, para 2,25 pontos percentuais em 2007, quando as mulheres passaram a fazer quase o dobro (1,9 vezes) deste tipo de viagem que os homens. As viagens motivo “servir passageiro” são menos representativas do total para ambos sexos e sempre mais frequentes para mulheres

do que para homens. Excluindo 1987, cujos dados não estavam disponíveis nesta categoria, a relação entre o percentual feminino e o masculino era de 2,3 em 1977, passou para 1,8 em 1997 e para 1,5 em 2007.

Gráfico 16 – Proporção das viagens do sexo feminino e do sexo masculino, segundo o motivo de destino, por ano



A variável **DURACAO** tem suas principais estatísticas apresentadas na Tabela 34. Não existem *missing values* neste campo e, desconsiderando quem não fez viagem (duração igual a 0 minutos), os valores mínimos para todos anos e ambos sexos são 1 minuto. As medianas da duração, independente do sexo, saem de 30 minutos em 1977 para 20 minutos em 1987 e 1997 e retornam para o valor 30 em 2007. Os valores de assimetria são todos positivos, indicando maior concentração à esquerda e cauda longa à direita da distribuição. Os valores de curtose evidenciam não se tratar de distribuição normal. O tempo médio geral de viagem vai de 1977 para 1987 e daí em diante só aumenta, comportamento semelhante ao segmento feminino. Os tempos médios das viagens dos homens cresce sistematicamente década a década, da ordem de 1 minuto entre 1977, 1987 e 1997. Já 2007 o acréscimo no tempo médio de viagem masculino subiu 4,5 minutos - tal efeito pode ser explicado pela expansão urbana da RMSP. Analisando esses dados segmentados por sexo, vê-se que as medianas das mulheres são sempre 5 minutos a menos que as dos homens. A diferença entre as durações médias das viagens de mulheres e de homens cresce de quase 0,5 minuto em 1977 para pouco mais de 5,5 minutos em 1987 e vem diminuindo desde então. Foram

feitos teste t para avaliar se eram estatisticamente significantes (intervalo de confiança de 95%) as médias entre os sexos, no mesmo ano; e entre os anos, para o mesmo sexo. Todas médias foram estatisticamente diferentes umas das outras.

Tabela 34 – Estatísticas da variável “DURACAO”, por ano

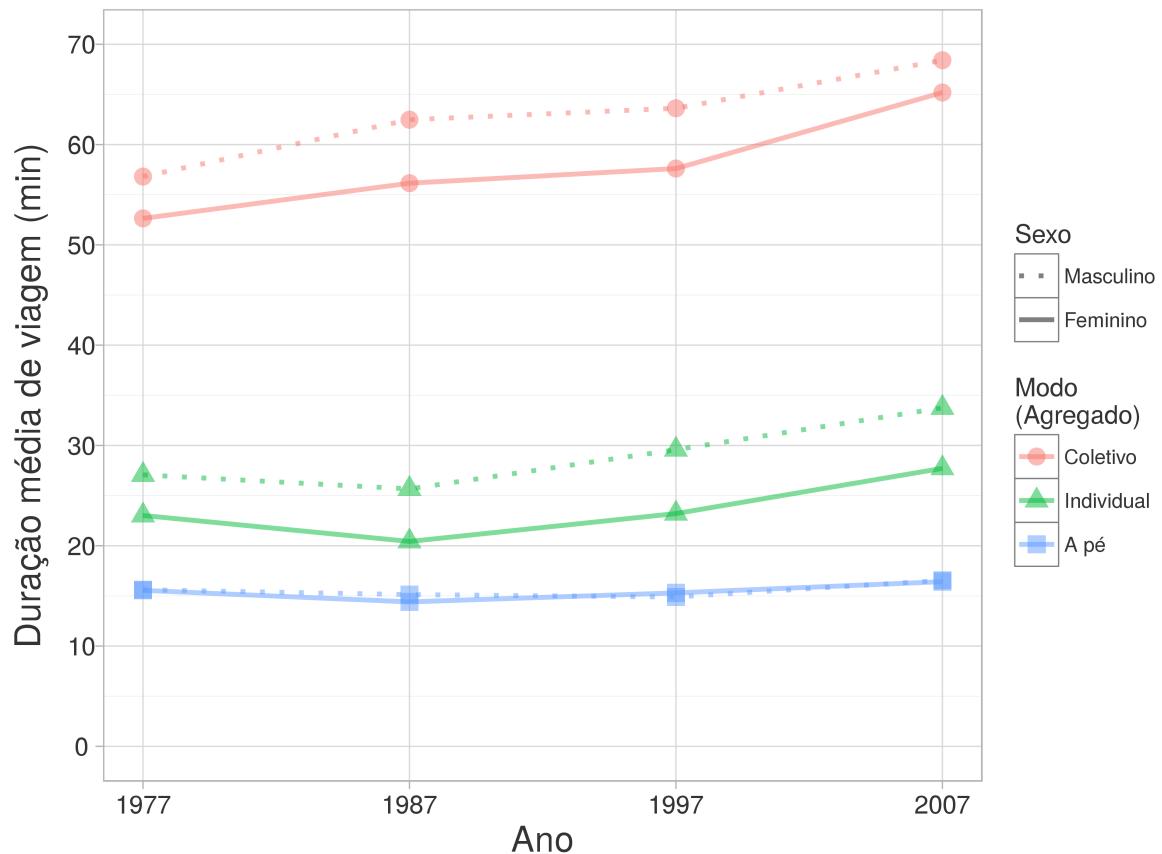
Total						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	36,07	31,99	30	240	1,74	3,78
1987	33,27	31,99	20	360	1,92	4,97
1997	34,13	33,52	20	370	1,94	4,54
2007	39,29	37,22	30	299	1,74	3,37
Sexo feminino						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	34,09	30,54	25	240	1,81	4,12
1987	30,15	29,76	20	360	2,08	5,89
1997	31,97	31,67	20	315	2,02	4,78
2007	37,96	36,72	25	299	1,81	3,68
Sexo Masculino						
ANO	Média	Desvio Padrão	Mediana	Máximo	Assimetria	Curtose
1977	34,47	32,89	30	240	1,69	3,53
1987	35,83	33,50	25	360	1,80	4,36
1997	36,11	35,02	25	370	1,86	4,25
2007	40,61	37,67	30	270	1,67	3,11

Com o intuito de melhor explorar as durações médias das viagens analisando modos e motivos, foram elaborados os Gráficos 17, 18 e 19. O Gráfico 17 apresenta as durações médias do transporte coletivo, individual e a pé, para homens e para mulheres. Verifica-se que:

- (i) As durações médias de homens e mulheres são muito próximas para as viagens a pé, girando em torno de 15 minutos para ambos.
- (ii) A duração média no transporte individual cai um pouco de 1977 para 1987 (de 23,0 para 20,4 min para mulheres e de 27,07 para 25,7 min para homens).
- (iii) A duração média no transporte individual aumenta de 1987 para 2007 (7,3 min para mulheres e 8,1 min para homens).
- (iv) A duração média no transporte coletivo aumenta entre 1977 e 2007 para ambos sexos, sendo a taxa mais acentuada de 1997 para 2007.
- (v) As diferenças nas durações médias nas viagens feitas por transporte coletivo entre

mulheres e homens aumenta de 1977 (4,2 min) para 1987 (6,3 min) e depois decresce nas décadas seguintes (6,0 min em 1997 e 3,2 min em 2007).

Gráfico 17 – Comparação entre as durações médias de viagem por ano e por sexo, segundo os modos (agregados)



O Gráfico 18 apresenta as durações médias dos modos de transporte coletivo (ônibus de linha, ônibus de empresa/escolar, lotação/perua/van/microônibus, Metrô e trem), para homens e para mulheres - foram separados os modos de alta capacidade dos demais para facilitar a compreensão dos gráficos. Verifica-se que:

- (i) A duração média do trem é superior a todos outros modos, inclusive o Metrô, com médias gerais caindo de 83,3 min em 1977 para 80,8 min em 1987 e crescendo para 89,45 min em 1997. Em 2007, esse valor permanece no mesmo patamar de 1997 (90,2 min).
- (ii) Para o trem, a duração média feminina é inferior à masculina em 1977 (diferença de 2,9 min) e 1987 (diferença de 2,1 min), supera a masculina em 1997 (por 1,6 min) e distancia da masculina em 2007 (diferença de 8,0 min).
- (iii) A duração média das viagens de Metrô crescem sistematicamente, pelo menos 6 min por década, de 1977 (53,8 min) a 2007 (74,4 min).
- (iv) Para o Metrô, a duração média feminina é superior à masculina em 1977 (diferença de 1,3 min) e em 2007 (diferença de 1,7 min). A situação é inversa, com durações médias

das viagens masculinas superiores às femininas em 1987 (diferença de 3,8 min) e 1997 (diferença de 5,0 min).

(v) A duração média das viagens dos ônibus de linha crescem sistematicamente, 5 min por década entre 1977 e 1987, 0,9 min entre 1987 e 1997, e 8,8 min entre 1997 e 2007.

(vi) Para ônibus de linha, a duração média feminina é inferior à masculina - a diferença é de 4,0 min em 1977, de 6,9 min em 1987, de 6,3 min em 1997 e de 3,9 min em 2007.

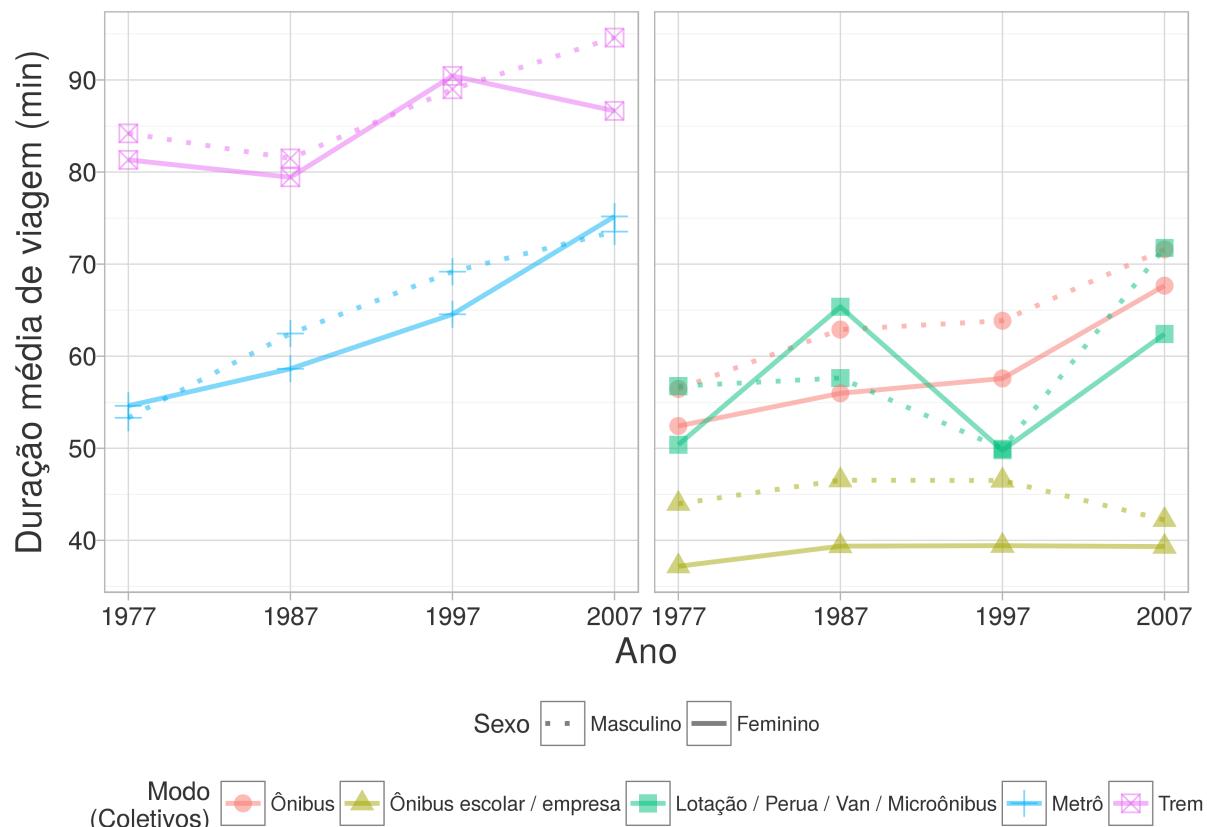
(vii) A duração média das viagens de lotações/vans crescem de 1977 (54,1 min) para 1987 (60,8 min), caem em 1997 (49,9 min) e sobem novamente em 2007 (66,3 min).

(viii) Para viagens de lotações/vans, a duração média feminina era inferior à masculina em 1977 (diferença de 6,4 min) e em 2007 (diferença 9,4 de min). A situação é inversa em 1987, com diferenças de 7,7 min entre os sexos. E, em 1997, as durações médias são estatisticamente iguais.

(ix) A duração média das viagens de ônibus escolar/fretado são as menores entre os transportes coletivos, para ambos sexos variando numa faixa de 40,9 a 44,1 min.

(x) Para viagens de ônibus escolar/fretado, a duração média feminina é sempre inferior à masculina - a diferença é de 6,8 min em 1977, de 7,2 min em 1987, de 7,1 min em 1997 e de 2,9 min em 2007.

Gráfico 18 – Comparação entre as durações médias de viagem por ano e por sexo, segundo os modos coletivos

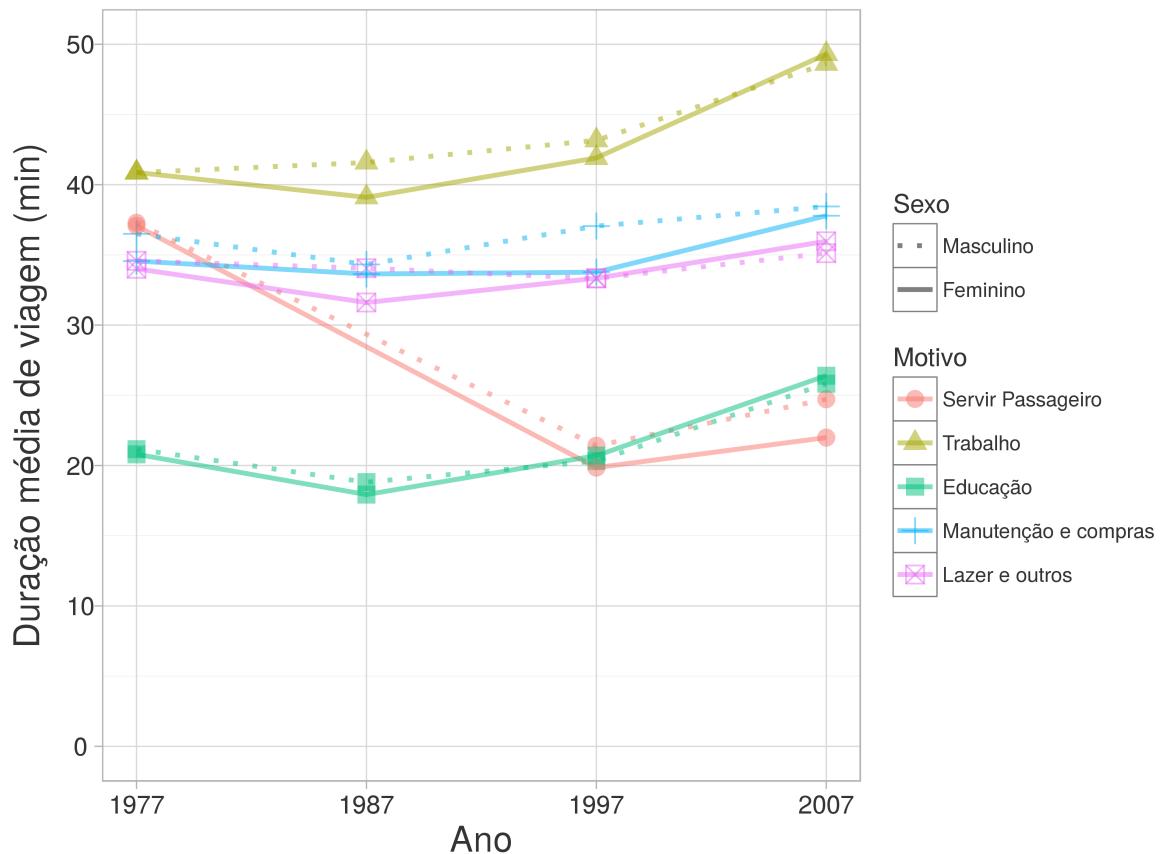


O Gráfico 19 apresenta as durações médias por motivo de viagem (trabalho, educação, servir passageiro, manutenção/compras e lazer/outros), para homens e para mulheres. Verifica-se que:

- (i) A duração média da viagem motivo trabalho (de natureza compulsória) é superior a de todos outros motivos. Ela permanece no mesmo patamar entre 1977 (40,9 min) e 1987 (40,8 min), sobe um pouco em 1997 (42,7 min) e continua a subir em 2007 (48,9 min).
- (ii) Para o motivo trabalho, a duração média feminina é estatisticamente igual à masculina em 1977, inferior, em 1987 (diferença de 2,5 min) e em 1997 (diferenças de 1,3 min) e levemente superior em 2007 (0,7 min).
- (iii) A duração média das viagens motivo educação decresce de 1977 (21,0 min) para 1987 (18,3 min), a partir de quando sobem até 2007 (26,1 min).
- (iv) Para o motivo educação, a duração média feminina é praticamente igual à masculina em 1977 e em 1997 (diferenças de menos de 0,5 min), levemente inferior à masculina em 1987 (diferença de 0,9 min) e superior em 2007 (diferença de 0,6 min).
- (v) A duração média das viagens motivo manutenção/compras decresce de 1977 (35,2 min) para 1987 (33,9 min), a partir de quando sobem até 2007 (38,0 min).
- (vi) Para viagens motivo manutenção/compras, a duração média feminina é inferior à masculina - a diferença é de 1,9 min em 1977, de 0,7 min em 1987, de 3,3 min em 1997 e de 0,7 min em 2007.
- (vii) A duração média das viagens motivo lazer/outros decresce de 1977 (37,5 min) para 1987 (34,7 min), a partir de quando sobem até 2007 (40,7 min).
- (viii) Para viagens motivo lazer/outros, a duração média feminina é inferior à masculina - a diferença é de 3,3 min em 1977, de 5,6 min em 1987, de 4,0 min em 1997 e de 2,7 min em 2007.
- (ix) A duração média das viagens motivo servir passageiro decresce de 1977 (37,2 min) para 1997 (20,43 min), a partir de quando sobem até 2007 (23,1 min). Vale destacar que em 1987 não havia o modo servir passageiro.
- (x) Para viagens motivo servir passageiro, a duração média feminina é estatisticamente igual à masculina em 1977, e inferior em 1997 (diferenças de 1,6 min) e em 2007 (diferenças de 2,7 min).

Em “Estatísticas sob Suspeita”, Carrasco (2012, p.100-101) propõe indicadores com base na experiência das mulheres e, especificamente, no capítulo relativo ao acesso à mobilidade e ao planejamento territorial, recomenda a formulação e consideração dos seguintes indicadores quando da formulação de políticas públicas: motivos dos deslocamentos, meio utilizado nos deslocamentos e distâncias dos deslocamentos, entre outros. Anteriormente foram apresentados os panoramas de motivos e modos, agora, serão exploradas as distâncias de deslocamento.

Gráfico 19 – Comparação entre as durações médias de viagem por ano e por sexo, segundo o motivo da viagem



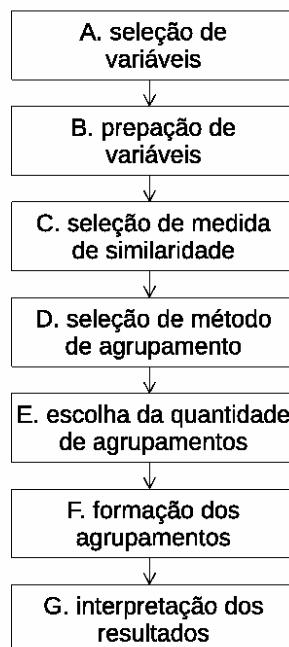
A variável de distância da viagem (**DIST_VIAG**) contém a distância euclidiana entre as coordenadas x e y de origem e as coordenadas x e y de destino, com as limitações que a determinação dessas coordenadas impõem, já que foram calculadas a partir dos centroides das subzonas ou zonas (na ausência das subzonas).

6.3 Análise de conglomerados para identificação de grupos

A análise de conglomerados tem como objetivo segregar elementos (observações) “em grupos homogêneos internamente, heterogêneos entre si e mutuamente exclusivos, a partir de determinados parâmetros conforme uma medida similaridade ou distância” (FÁVERO et al., 2009, p.196). Esta técnica estatística de interdependência foi escolhida pois se deseja agrupar as pessoas, ou mesmo famílias, em grupos homogêneos em função da similaridade do comportamento de viagens.

Foi utilizada a função *hclust* do pacote *stats*⁷ (R Core Team, 2011) para realizar as análises hierárquicas de conglomerados. Devido ao tamanho da base de dados e ao baixo desempenho em termos computacionais do pacote *stats*, preferiu-se a função *hclust* do pacote *fastcluster*⁸. Segundo Müllner (2013), no pior caso, a função do pacote *stats* tem seu tempo de execução proporcional a N^3 , já a mesma função no pacote *fastcluster*, é proporcional a N^2 . As análises de conglomerados não hierárquicas foram executadas pela função *kmeans* do pacote *stats*. Na Figura 9 pode-se observar os passos realizados nesta etapa de análise.

Figura 9 – Etapas para a realização de análise de conglomerados (*clusters*)



⁷ Também existe a função *agnes*, do pacote *cluster*, cuja documentação está disponível em <<https://cran.r-project.org/web/packages/cluster/cluster.pdf>> e que se propõe a resolver o mesmo algoritmo. Porém, o resultado obtido dos três pacotes (*stats*, *fastcluster* e *cluster*) não foram iguais. Murtagh e Legendre (2014) já apontavam divergência entre as funções *hclust* e *agnes* quando aplicadas à mesma matriz de distâncias.

⁸ Detalhes da implementação disponível em Müllner (2011).

Na **etapa A**, foram delimitados dois conjuntos iniciais de variáveis de análise:

- **conjunto I:** De atributos de viagem, relativas à família, a saber: FAM_DIST_TOT, FAM_DIST_MED, FAM_DURACAO_TOT, FAM_DURACAO_MED, FAM_VIAG_TOT;
- **conjunto II:** De atributos de viagem, relativas à pessoa, a saber: PESS_DIST_TOT, PESS_DIST_MED, PESS_DURACAO_TOT, PESS_DURACAO_MED, PESS_MODO_ONIBUS, PESS_MODO_DIRIG, PESS_MODO_PASS, PESS_MODO_TREM, PESS_MODO_MOTO, PESS_MODO_BICI, PESS_MODO_APE, PESS_MODO_OUTROS, PESS_NO_MODOS, PESS_MOTIVO_TRAB, PESS_MOTIVO_EDUC, PESS_MOTIVO_RES, PESS_MOTIVO_SERV_PAS, PESS_MOTIVO_MANUT_COMPRA, PESS_MOTIVO_LAZER_OUTROS, PESS_NO_MOTIVOS, PESS_PER_MADRUG, PESS_PER_COM_MAN, PESS_PER_MANHA, PESS_PER_MEIODIA, PESS_PER_TARDE, PESS_PER_COM NOI, PESS_PER_NOITE, PESS_NO_PERIODOS, TOT_VIAG.

Ocorre em diversos bancos de dados, inclusive neste, de se desejar analisar variáveis cujas unidades e ordens de grandezas não são as mesmas. [Faria \(2009\)](#) indica que há duas razões para padronizar dados: (i) "evitar que as unidades escolhidas para mensurar as características afetem arbitrariamente a similaridade entre indivíduos", e (ii) fazer com "que as características contribuam igualmente na avaliação da similaridade entre indivíduos". Se houver alguma unidade de medição que tenha uma amplitude maior que as demais (como é o caso das distâncias totais da família em relação à distância média da pessoa), ela terá maior peso na análise de *cluster*. Então, para mitigar o efeito dessas diferenças é indicado padronizar os dados antes de submetê-los à análise de conglomerados. Ao fazer isso, o pesquisador assume que a importância da variável decresce conforme a variabilidade aumenta ([EVERITT et al., 2011](#)). Há diversas formas de fazer essa padronização, as mais comuns são *z-scores* e normalização ([FÁVERO et al., 2009](#)). Há também possibilidades como dividir pela mediana dos desvios absolutos ou pelos intervalos de valores da variável ([GNANADESIKAN; KETTENRING; TSAO, 1995 apud EVERITT et al., 2011](#)), porém, a complexidade deste últimos métodos não se justificam frente ao conjunto de dados em discussão. Assim, na **etapa B** do presente trabalho, foi feita padronização das variáveis pelo método *z-scores* conforme Equações (6.1), (6.2) e (6.3).

$$Z(x)_i = \frac{x_i - \bar{x}}{\sigma(x)} \quad (6.1)$$

sendo:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (6.2)$$

$$\sigma(x) = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (6.3)$$

Um dos principais problemas das técnicas de aglomeração não hierárquica (K-means) é definir de início o número de *clusters* desejado (HARTIGAN, 1985; FÁVERO et al., 2009; EVERITT et al., 2011). Existem algumas técnicas para essa determinação:

- (i) A ***upper tail rule*** que considera os valores de fusão como uma série. Calcula-se a média, desvio padrão, estatística t como o desvio normalizado a partir da média. Em seguida, calcula-se o desvio padrão para cada valor de fusão (assumida distribuição normal), seleciona o primeiro “significativo” como sendo aquele cuja estatística t excede o nível de 5% de significância. Assim, a hipótese nula é que o valor fusão do k-ésimo termo advém da distribuição normal dos valores de fusão (MOJENA, 1977).
- (ii) O índice **RMSSTD** (*root mean square standard deviation*), ou raiz quadrada do desvio padrão médio (ver Equação (6.4)), calcula a homogeneidade dos agrupamentos (SHARMA, 1996), de maneira que quanto mais compactos os grupos, menor o valor desta estatística.

$$RMSSTD = \sqrt{\frac{\sum_{j=1}^{nc} \sum_{k=1}^{n_j} (x_k - \bar{x}_j)^2}{\sum_{i=1}^{nc} (n_{ij} - 1)}} \quad (6.4)$$

- (iii) O índice **R^2 ajustado** (ver Equação (6.5)), indica dissimilaridade entre agrupamentos (SHARMA, 1996), assim, quanto mais alto o valor de R^2 ajustado, mais dissimilaridade existe entre os grupos. O pesquisador pode estabelecer um valor desejado para R^2 e, a partir daí, determinar o número de *clusters*.

$$R^2 = \frac{\left[\sum_{i=1}^{nc} \sum_{j=1}^{n_j} (x_i - \bar{x}_j)^2 \right] - \left[\sum_{i=1}^{nc} \sum_{k=1}^{n_{ij}} (x_k - \bar{x}_j)^2 \right]}{\sum_{j=1}^d \sum_{k=1}^{n_j} (x_k - \bar{x}_j)^2} \quad (6.5)$$

- (iv) O ***best cut*** é um método que se baseia em um dendrograma (representação bidimensional em forma de árvore) que deve ser cortado quando as diferenças entre grupos forem visualmente mais significativas. Nesta representação, as linhas são ligadas segundo níveis de similaridade que agragará os indivíduos (EVERITT et al., 2011).

Então, primeiro foi realizada a aglomeração hierárquica para definir as quantidades grupos. No método hierárquico, se há n observações (linhas), parte-se de n grupos, ou seja, existe uma observações por grupo. A partir de medidas de similaridade, num processo iterativo, as observações vão sendo agrupadas até que se chegue a um único grupo no final. Segundo (MAXWELL, 1977), primeiro é feita a conversão da matriz n versus p de dados em uma matriz n versus n de medidas de similaridade ou dissimilaridade, tendo-se

n unidades amostrais e p características. Optou-se por primeiro utilizar o **best cut** com os dendrogramas⁹ e, em seguida, seriam avaliados os índices RMSSTD e R^2 ajustado de 90%.

As medidas de (dis)similaridade podem ser de distância, correlação ou associação, esta última indicada para variáveis qualitativas. Como as variáveis selecionadas na **etapa A** são métricas ou *dummies*, são indicadas medidas de distância ou correlação. Na Tabela 35 é possível observar algumas das principais medidas de dissimilaridade comumente utilizadas. Optou-se, na **etapa C**, pela medida de distância euclidiana, indicada pela literatura (Hair Jr et al., 2005 apud FÁVERO et al., 2009), a ser adotada em conjunto com os métodos de agrupamento Ward (Ward Jr, 1963) e centroide. O arcabouço teórico indica que tanto as distâncias euclidianas quanto as euclidianas quadráticas resultarão nos mesmos *clusters* e, no caso da função *hclust* utilizada, a distância padrão calculada é a euclidiana.

Tabela 35 – Medidas de dissimilaridade utilizadas em análise de *cluster*

Medida	Fórmula
Distância Euclidiana	$d_{ij} = \left[\sum_{k=1}^p w_k^2 (x_{ik} - x_{jk})^2 \right]^{1/2}$
Distância <i>city block</i>	$d_{ij} = \sum_{k=1}^p w_k x_{ik} - x_{jk} $
Distância de Minkowski	$d_{ij} = \left(\sum_{k=1}^p w_k^r x_{ik} - x_{jk} ^r \right)^{1/r} \quad (r \geq 1)$
Distância de Canberra	$d_{ij} = \begin{cases} 0 & \text{for } x_{ik} = x_{jk} = 0 \\ \sum_{k=1}^p w_k x_{ik} - x_{jk} / (x_{ik} + x_{jk}) & \text{for } x_{ik} \neq 0 \text{ or } x_{jk} \neq 0 \end{cases}$

Fonte: (EVERITT et al., 2011)

Na **etapa D**, foi feita a aglomeração para o **conjunto I** de variáveis da família e para o **conjunto II** de variáveis da pessoa, utilizando tanto o método Ward quanto o centroide, utilizando filtros que captassem a ocorrência da família (F_FAM=1) ou da pessoa (F_PESS=1), respectivamente, para que não houvesse repetição indevida de famílias ou pessoas. Essas aglomerações foram feitas sem distinção dos anos e resultaram nos dendrogramas apresentados nas Figuras 10 e 11. Percebe-se que a forma do dendrograma difere muito pouco entre os métodos Ward e centroide para o mesmo conjunto de variáveis.

Considerando o **best cut** observa-se que os dendrogramas que partiram das variáveis de família, indicam 4 como sendo um número de *clusters* interessante de ser dado como

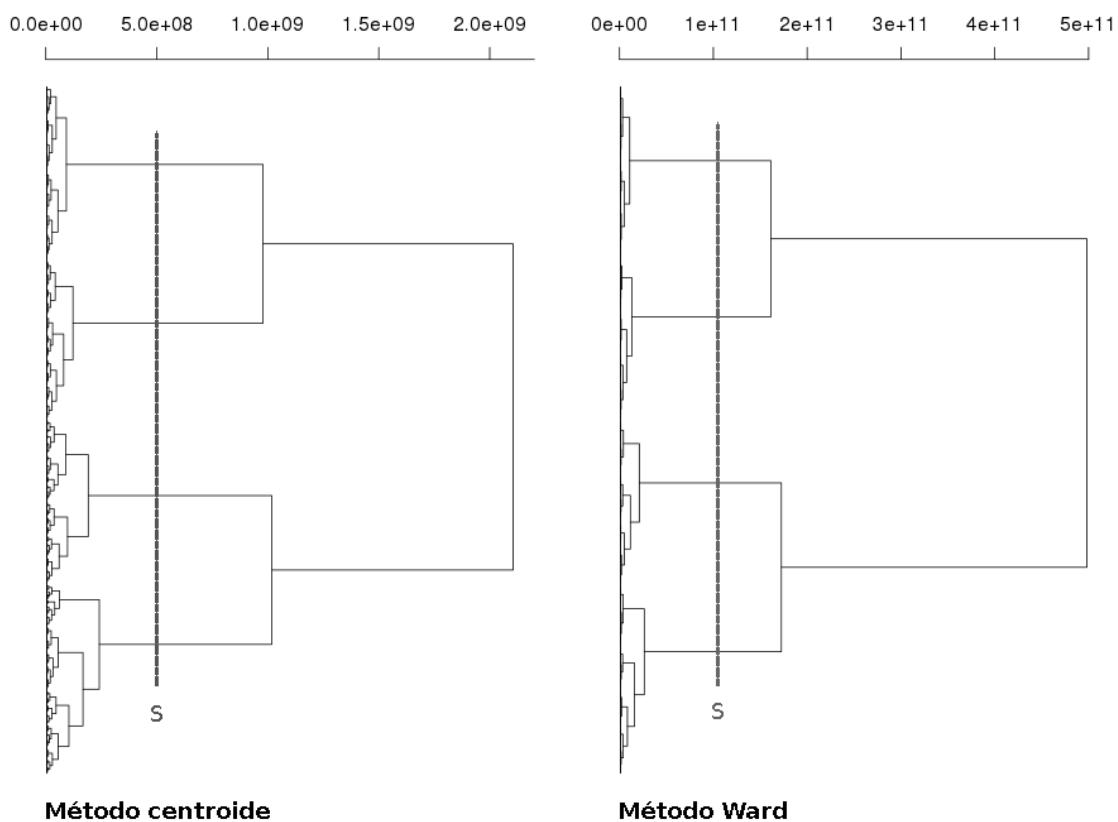
⁹ Os dendrogramas gerados admitiram escala não-monotônica.

input do método K-means - observar seções S da Figura 10. Os índices **RMSSTD** e R^2 ajustado, conforme pode ser visto no Gráfico 20, também corroboram para a divisão em 4 grupos, representando 90% da variância.

Partindo do conjunto de atributos de viagens relativas às pessoas, o **best cut** dos dendrogramas resultantes também indicam 4 *clusters*, seja pelo método Ward, seja pelo centroide - observar seções S da Figura 11. Os índices **RMSSTD** e R^2 ajustado, conforme pode ser visto no Gráfico 21, também corroboram para a divisão em 4 grupos, representando mais de 90% da variância.

Assim, na **etapa E**, definiu-se que seriam adotados 4 *clusters*, tanto para o conjunto de variáveis I (de família) como o II (de pessoas) no método de aglomeração K-means¹⁰ que, segundo Gouvea e La Plata (2006 apud FÁVERO et al., 2009), minimiza a variância interna aos grupos e maximiza a variância entre grupos.

Figura 10 – Dendrograma resultante da análise de conglomerados hierárquico, para o conjunto de atributos de viagens relativas às famílias



¹⁰ A função *kmeans* do pacote *stats* do software R implementa o algoritmo de Hartigan e Wong (1979) por padrão para sua execução, que utiliza também a distância euclidiana como medida de similaridade.

Gráfico 20 – Avaliação do número de *clusters* para o conjunto de atributos de viagens relativas às famílias

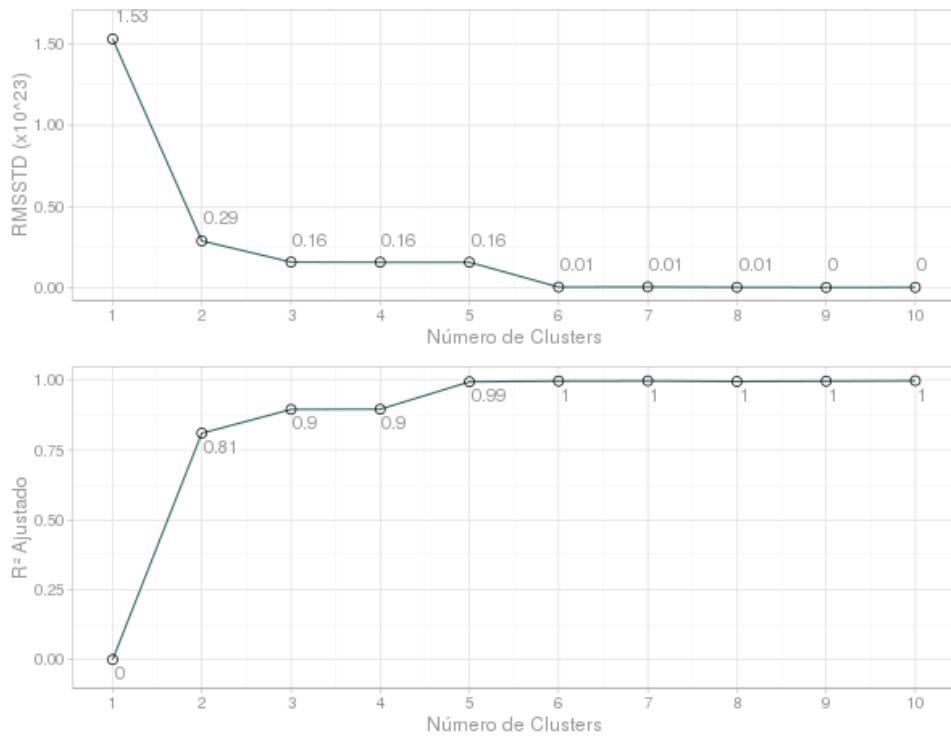


Figura 11 – Dendrograma resultante da análise de conglomerados hierárquico, para o conjunto de atributos de viagens relativas às pessoas

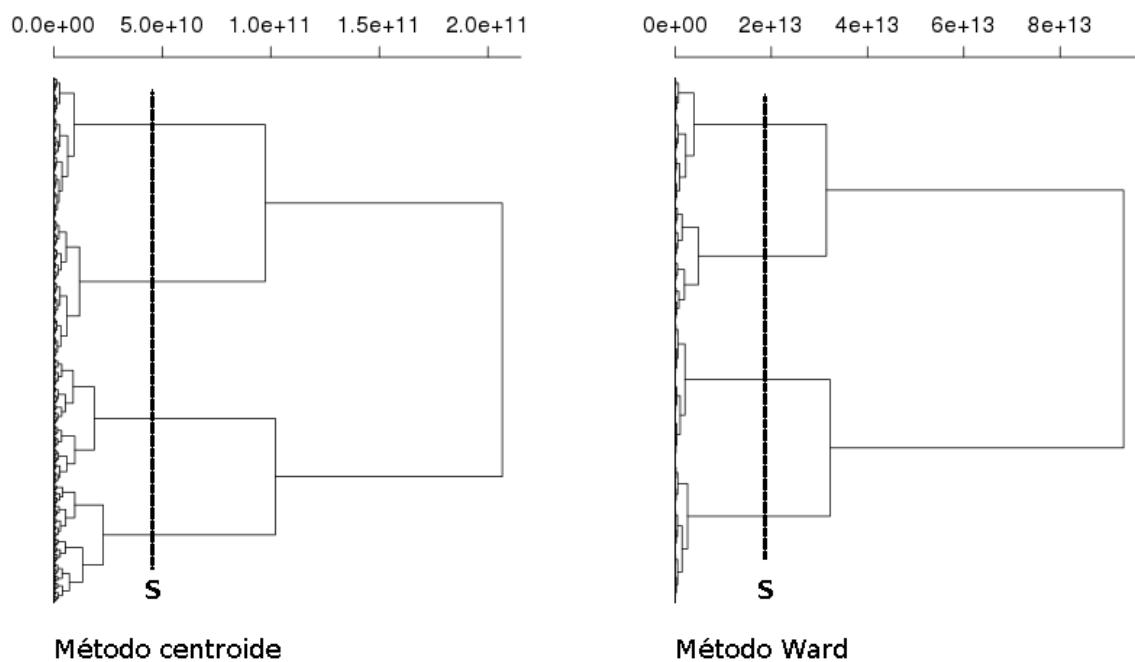
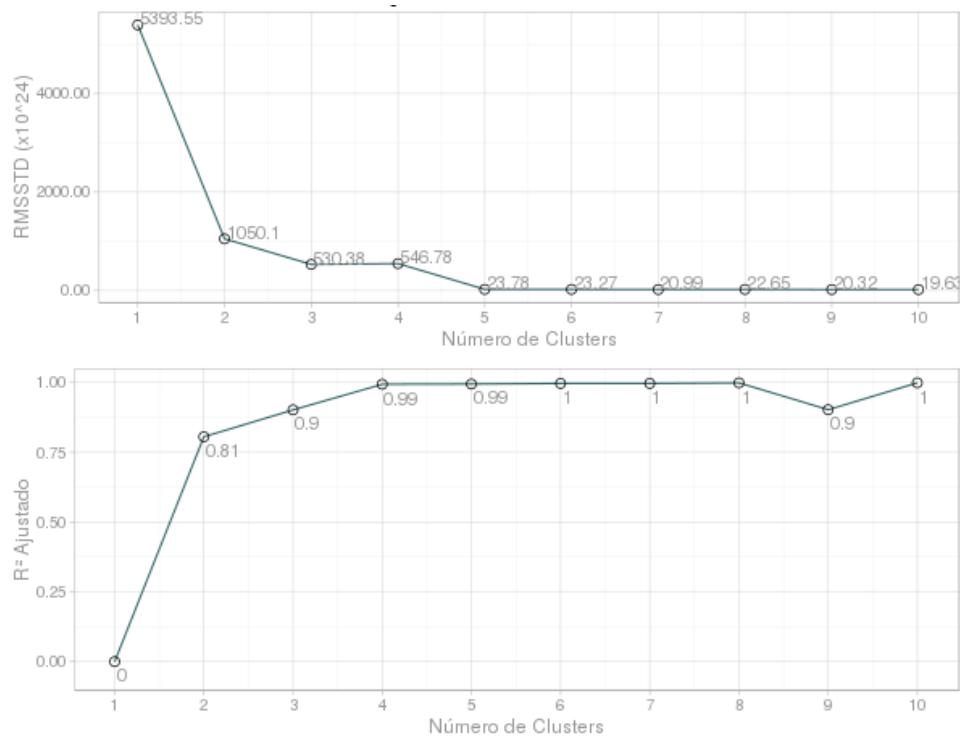


Gráfico 21 – Avaliação do número de *clusters* para o conjunto de atributos de viagens relativas às pessoas



Na **etapa F**, foram formados então quatro agrupamentos para atributos de viagens de **família**, pelo método Ward. Observa-se que os grupos formados correspondem exatamente às observações de cada ano. Ou seja, o *cluster* 1 agregou as observações de 1977, o *cluster* 2 agregou as observações de 1987, e assim por diante - ver Tabela 36. Entretanto, utilizando o método centroide, houve a união de 1997 e 2007, além da separação de 1987 em dois grupos distintos - ver Tabela 37. Ao analisar os quatro agrupamentos para atributos de viagens de **pessoas**, tanto com o método centroide quanto com o Ward, os grupos formados também correspondem exatamente às observações de cada ano.

Tabela 36 – Resultado do agrupamento de 4 *clusters*, por atributos de viagens de família - método Ward

Cluster n°	% de famílias de 1977	% de famílias de 1987	% de famílias de 1997	% de famílias de 2007
1	100	0	0	0
2	0	100	0	0
3	0	0	100	0
4	0	0	0	100

Tabela 37 – Resultado do agrupamento de 4 *clusters*, por atributos de viagens de família - método centroide

<i>Cluster</i> nº	% de famílias de 1977	% de famílias de 1987	% de famílias de 1997	% de famílias de 2007
1	100	0	0	0
2	0	100	0	0
3	0	100	0	0
4	0	0	51,94	48,06

Tomando três agrupamentos, tendo em foco a **família**, pelo método Ward os grupos que se unem são os anos de 1997 e 2007. Já pelo método centroide, 1997 destaca-se de 2007, enquanto 1987 se agrega completamente a 1977, conforme pode ser observado nas Tabelas 38 e 39. Ao focar os atributos de viagens das **pessoas**, agrupadas em três *clusters*, não houve diferença no resultado utilizando Ward ou centroide, tal como foi com quatro grupos.

Tabela 38 – Resultado do agrupamento de 3 *clusters*, por atributos de viagens de família - método Ward

<i>Cluster</i> nº	% de famílias de 1977	% de famílias de 1987	% de famílias de 1997	% de famílias de 2007
1	100	0	0	0
2	0	100	0	0
3	0	0	46,5	53,5

Tabela 39 – Resultado do agrupamento de 3 *clusters*, por atributos de viagens de família - método centroide

<i>Cluster</i> nº	% de famílias de 1977	% de famílias de 1987	% de famílias de 1997	% de famílias de 2007
1	48,1	51,9	0	0
2	0	0	100	0
3	0	0	0	100

Na Tabela 40 é possível observar que 1977 e 1987 se separam, e 1997 se une a 2007. Percebe-se que a variável ANO, embora não inserida nas análises de conglomerados, acabou sendo a grande diferenciadora dos grupos. Esses resultados levantam a questões como:

- (i) o tempo (em si ou como *proxy* de outras variáveis) é uma categoria de análise relevante;
- (ii) pode ter havido evolução no método de pesquisa e o agrupamento temporal esteja captando este efeito;
- (iii) parece haver semelhanças entre 2007 e 1997;
- (iv) parece haver semelhanças entre 1987 e 1977, talvez mais fracas que as entre 2007 e 1997;
- (v) não parece haver semelhanças tão fortes entre 1987 e 1997.

Tabela 40 – Resultado do agrupamento de 3 *clusters*, por atributos de viagens de pessoas - métodos Ward e centroide

<i>Cluster</i> nº	% de pessoas de 1977	% de pessoas de 1987	% de pessoas de 1997	% de pessoas de 2007
1	100	0	0	0
2	0	100	0	0
3	0	0	51,9	48,1

Sob a perspectiva de gênero, retomando os dados de participação na PEA apresentados no Gráfico 1, percebe-se o $\Delta_{1991-1980} = 6,3\%$, $\Delta_{2000-1991} = 11,2\%$ e $\Delta_{2010-2000} = 4,8\%$. Embora, os dados da PEA refiram-se aos anos 1980, 1991, 2000 e 2010, não coincidentes com os das Pesquisas OD (1977, 1987, 1997 e 2007), se tomarmos os anos mais próximos como referencial de análise, percebemos que o maior salto na participação feminina no mercado de trabalho ocorreu entre 2000 e 1991, e que as pesquisas que parecem apresentar a maior dissemelhança são as OD-1997 e OD-1987. Não é possível fazer muitas afirmações a partir somente deste paralelo, entretanto, isto pode ser um indicativo de que os padrões de mobilidade se alteraram sob efeito do tempo e também considerando o gênero como categoria de análise.

Para explorar melhor o que ocorre dentro de cada grupo e o que ocorre entre grupos foram analisadas as características de viagens, das pessoas e das famílias a partir da segregação dos 4 *clusters* resultantes - ver Anexo E.

No agrupamento feito por atributos de viagens da família pelo método Ward, observa-se que a maior diferença percentual entre valores mínimo e máximo entre grupos ocorreu com a variável “% de pessoas com superior completo” (76%) e a menor diferença

ocorreu com a variável “sexo” (4%). A Tabela 41 apresenta as variáveis que apresentaram as diferenças percentuais entre valores máximos e mínimos superiores a 50%. Para estas nove variáveis foi realizado teste qui-quadrado de Pearson (de independência) em relação à variável que indica os *clusters* e todas mostraram-se significativas considerando um intervalo de confiança de 99%.

Tabela 41 – Ordenamento das variáveis pelas maiores diferenças percentuais - para agrupamento FAM_CLUSTER_WARD4

Ranking	Variável	Diferença %
1º	% de pessoas com superior completo	76
2º	% de pessoas com situação familiar ‘outros’	71
3º	% de famílias na Classe A	67
4º	% de pessoas com médio completo ou superior incompleto	66
5º	% de pessoas que servem passageiro no destino	65
6º	% de famílias na Classe E	61
7º	% de famílias com presença de criança entre 0 e 4 anos	60
8º	% de pessoas empregadas	59
9º	% de famílias com presença de criança entre 5 e 9	56

No agrupamento feito por atributos de viagens da família pelo método centroide, observa-se que a maior diferença percentual entre valores mínimo e máximo entre grupos ocorreu com a variável “% de famílias na Classe A” (79%) e a menor diferença ocorreu com a variável “Média da quantidade de trabalhadores (as) na família” (2%). Vale mencionar que as diferenças percentuais entre valores mínimo e máximo ocorreram entre os grupos 2 (uma parte de 1987) e 3 (outra parte de 1987) em relação as seguintes características: (i) distância média por viagem da pessoa; (ii) distância média por pessoa da família; (iii) % de pessoas do sexo feminino e (iv) % de outros parentes / agregados.

A Tabela 42 apresenta as variáveis que apresentaram as diferenças percentuais entre valores máximos e mínimos superiores a 50%. Para estas nove variáveis foi realizado teste qui-quadrado de Pearson (de independência) em relação à variável que indica os *clusters* e todas mostraram-se significativas considerando um intervalo de confiança de 99%.

Oito dentre as nove variáveis listadas nas Tabelas 41 e 42 repetem-se, indicando mais consistência do que divergência entre os métodos utilizados. Ao se dar importância para a % de pessoas com superior completo ou % de pessoas com médio completo ou superior incompleto, na realidade, se está elencando a variável **grau de instrução** como

Tabela 42 – Ordenamento das variáveis pelas maiores diferenças percentuais - para agrupamento FAM_CLUSTER_CENTROIDE4

Ranking	Variável	Diferença %
1º	% de famílias na Classe A	79
2º	% de pessoas empregadas	76
3º	% de pessoas com superior completo	73
4º	% de pessoas com situação familiar ‘outros’	71
5º	% de famílias na Classe E	66
6º	% de pessoas com médio completo ou superior incompleto	65
7º	% de pessoas que servem passageiro no destino	61
8º	% de famílias na Classe E	57
9º	% de famílias com presença de criança entre 0 e 4 anos	51

relevante para explicar os agrupamentos. As porcentagens de famílias nas classes A, B ou E, colocam a **renda familiar** como outra variável relevante. Raciocínio análogo se aplica entre a % de pessoas com situação familiar ‘outros’ e a **situação familiar** e entre a % de pessoas que servem passageiro no destino e o **motivo no destino**. *Dummies* que indicam se a pessoa **trabalha ou não ou se há crianças até 9 anos** na família também são relevantes para explicar os agrupamentos.

Seguindo recomendação de Vespucci (2003) foi feita uma análise de conglomerados a partir da primeira análise de conglomerados. Ou seja, o procedimento de análise de *clusters* (hierárquico e não hierárquico) foi feito separadamente para 1977, 1987, 1997 e 2007, para os conjuntos de variáveis I (famílias) e II (indivíduos).

» INSERIR TABELAS AQUI

6.4 Regressão logística para investigar a formação de grupos

A regressão logística¹¹ (binária) é uma técnica estatística que busca investigar a relação entre uma variável dependente categórica (neste caso uma *dummy*) e variáveis explicativas métricas ou não métricas. Não se trata de um método de classificação, mas neste caso, é uma técnica que está sendo combinada após a utilização da análise de conglomerados, para tentar responder quais os pesos de cada variável na formação dos grupos (coincidente com os anos, na maior parte das aglomerações).

As probabilidades são estimadas usando a função logística definida conforme Equações (6.6) e (6.7). Z (logit) assume valores entre menos e mais infinito, levando $f(Z)$ a assumir valores entre 0 e 1, respectivamente. A ideia, analogamente à regressão linear, é construir uma função de predição que pondere as importâncias das variáveis explicativas na explicação de um determinado evento. Só é preciso ressaltar que na regressão linear, os estimadores correspondem às probabilidades, diretamente, já na regressão logística, o que se obtém, são scores do tipo $(\mathbf{X}_i, k) = \beta_k \cdot \mathbf{X}_i$, onde \mathbf{X}_i é o vetor das variáveis descritivas por observação i , k é o vetor de pesos (ou coeficientes da regressão) correspondente à escolha k . Nos modelos de escolha discreta, as observações correspondem às pessoas que escolhem entre k opções. No presente caso, as observações correspondem às famílias ou às pessoas, que foram agrupadas num determinado *cluster* k . O termo $(p/1-p)$ é que representa, de fato, a chance de ocorrência do evento.

$$f(Z) = \frac{1}{1 + e^{-Z}} \quad (6.6)$$

sendo:

$$Z = \ln \left(\frac{p}{1 - p} \right) = \alpha + \sum \beta_k \cdot X_i \quad (6.7)$$

Na regressão logística multinomial a variável dependente é categórica com duas ou mais categorias, de natureza ordinal ou nominal. Tomemos por linha de análise o caso intrigante em que os *clusters* coincidem com os anos. Assim, tomando 2007 (categoria 4) por referência temos as Expressões (6.8), (6.9) e (6.10).

$$Z = \ln \left(\frac{P(\text{cluster} = 1|X)}{P(\text{cluster} = 4|X)} \right) = \alpha_1 + \sum \beta_1 \cdot X_i \quad (6.8)$$

$$Z = \ln \left(\frac{P(\text{cluster} = 2|X)}{P(\text{cluster} = 4|X)} \right) = \alpha_2 + \sum \beta_2 \cdot X_i \quad (6.9)$$

$$Z = \ln \left(\frac{P(\text{cluster} = 3|X)}{P(\text{cluster} = 4|X)} \right) = \alpha_3 + \sum \beta_3 \cdot X_i \quad (6.10)$$

¹¹ Técnica desenvolvida inicialmente por ?? (??).

Foram utilizadas tanto a função *mlogit* da versão 11 do software estatístico STATA quanto a função *multinom* do pacote *nnet*¹² do R, para o cálculo da regressão logística multinomial de ANO, em função das variáveis de atributos de viagem, relativas à família.

Tabela 43 – Resultado da regressão logística multinomial com categoria de referência Ano=4 (correspondente a 2007)

Variáveis	Coeficientes	Desvio Padrão	z valor	Pr(> z)
ANO=1977				
intercepto	-0,576	0,0194	-29,67	0,000 (***)
FAM_VIAG_TOT	0,098	0,0031	31,59	0,000 (***)
FAM_DIST_TOT	3,01 E-06	0,0000	4,90	0,000 (***)
FAM_DIST_MED	3,35 E-06	0,0000	0,97	0,334 (-)
FAM_DURACAO_TOT	-0,002	0,0001	-11,89	0,000 (***)
FAM_DURACAO_MED	0,001	0,0007	0,91	0,364 (-)
ANO=1987				
intercepto	-0,431	0,0187	-23,10	0,000 (***)
FAM_VIAG_TOT	0,077	0,0031	24,99	0,000 (***)
FAM_DIST_TOT	4,18 E-07	0,0000	0,67	0,501 (-)
FAM_DIST_MED	3,19 E-06	0,0000	0,96	0,338 (-)
FAM_DURACAO_TOT	-0,001	0,0001	-9,91	0,000 (***)
FAM_DURACAO_MED	0,003	0,0007	5,07	0,000 (***)
ANO=1997				
intercepto	-0,328	0,0186	-17,64	0,000 (***)
FAM_VIAG_TOT	0,054	0,0032	16,94	0,000 (***)
FAM_DIST_TOT	-5,43 E-06	0,0000	-7,85	0,000 (***)
FAM_DIST_MED	-2,27 E-05	0,0000	-6,14	0,000 (***)
FAM_DURACAO_TOT	-2,60 E-04	0,0001	-1,98	0,048 (***)
FAM_DURACAO_MED	0,005	0,0007	7,58	0,000 (***)

Os resultados relativos aos coeficientes, apresentados na Tabela 43. A maior di-

¹² Documentação do pacote *nnet* disponível em <<https://cran.r-project.org/web/packages/nnet/nnet.pdf>> - acesso em 27 de janeiro de 2016.

ferença entre os grupos está no intercepto (provavelmente por englobar o termo de erro indicando a ausência de variável explicativa relevante) no modelo. O total de viagens da família é a variável com maior relevância, cujos coeficientes vão decrescendo quanto mais nos aproximamos do ano de referência, 2007. Não foram significativos os coeficientes relativos a distância média da família para 1977 e 1987, assim como a distância total da família para 1987 e a duração média da família para 1977. A duração total foi significante para todos anos, em relação a 2007. O grau de explicação do modelo, expresso pelo pseudo R^2 , foi de 0,0097, um valor baixo. Entretanto, o foco aqui não é criar um modelo preditivo, mas diagnosticar a influência de cada variável na formação dos agrupamentos.

» PARA CADA ANO INSERIR AQUI

Referências

- ABEP. *Critério Brasil*. São Paulo, 2009. 4 p. Citado na página 64.
- AHMED, Q. I.; LU, H.; YE, S. Urban transportation and equity: A case study of Beijing and Karachi. *Transportation Research Part A: Policy and Practice*, v. 42, n. 1, p. 125–139, jan 2008. ISSN 09658564. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0965856407000559>>. Citado na página 25.
- ALVA, E. N. *Metrópoles (in)sustentáveis*. Rio de Janeiro: Relume-Dumará, 1997. 152 p. Disponível em: <<http://livraria.folha.com.br/livros/urbanismo/metropoles-in-sustentaveis-eduardo-neira-alva-1043600.html>>. Citado na página 25.
- ALVES, J. E. D. *O crescimento da PEA e a redução do hiato de gênero nas taxas de atividade no mercado de trabalho*. 2013. 1–5 p. Disponível em: <<http://www.ie.ufrj.br/aparte/1>>. Citado na página 37.
- AURÉLIO. *Dicionário Online*. 2014. Disponível em: <<http://www.dicionariodoaurelio.com/genero>>. Citado na página 30.
- BANISTER, D. The trilogy of distance, speed and time. *Journal of Transport Geography*, Elsevier Ltd, v. 19, n. 4, p. 950–959, jul 2011. ISSN 09666923. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0966692310001973>>. Citado na página 25.
- BEAUVOIR, S. *O Segundo Sexo: a experiência vivida*. 2ª. ed. Rio de Janeiro: Difusão Europeia do Livro, 1967. 500 p. Citado na página 32.
- BEST, H.; LANZENDORF, M. Division of labour and gender differences in metropolitan car use: An empirical study in Cologne, Germany. *Journal of Transport Geography*, v. 13, n. 2, p. 109–121, jun 2005. ISSN 09666923. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0966692304000201>>. Citado 6 vezes nas páginas 25, 26, 27, 48, 50 e 51.
- BLAY, E. A. Um caminho ainda em construção: a igualdade de oportunidades para as mulheres. *Revista USP*, São Paulo, n. 49, p. 82–97, 2001. Disponível em: <<http://www.usp.br/revistausp/49/06-evablay.pdf>>. Citado 3 vezes nas páginas 35, 36 e 38.
- BRASIL. *Lei nº 12.587 de 03 de janeiro de 2012. Institui as diretrizes da Política Nacional de Mobilidade Urbana*. 2012. 1–11 p. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l125>. Citado na página 25.
- BRITO, M. N. C. Gênero e cidadania: referenciais analíticos. *Revista Estudos Feministas*, v. 9, n. 1, p. 291–298, 2001. Disponível em: <<http://www.redalyc.org/pdf/381/38109117.pdf>>. Citado na página 35.
- CAMERON, M. *Efficiency and fairness on the road*. Oakland, 1994. Citado na página 44.

CARRASCO, C. *Estatísticas sob suspeita: proposta de novos indicadores com base na experiência das mulheres*. 1. ed. São Paulo: SOF - Sempreviva Organização Feminista, 2012. 160 p. Disponível em: <<http://www.sof.org.br/wp-content/uploads/2015/07/Estatisticas.pdf>>. Citado na página 127.

CARVALHO, J. M. de. *Cidadania no Brasil - O longo Caminho*. Rio de Janeiro: Civilização Brasileira, 2002. Citado na página 35.

CHANT, S. *Gender and Migration in Developing Countries*. London: Belhaven Press, 1992. Citado na página 39.

CHU, Y.-L. Automobile Ownership Analysis Using Ordered Probit Models. *Transportation Research Record*, v. 1805, n. 1, p. 60–67, 2002. ISSN 0361-1981. Citado na página 97.

CRANE, R. Is There a Quiet Revolution in Women's Travel? Revisiting the Gender Gap in Commuting. *Journal of the American Planning Association*, v. 73, n. 3, p. 298–316, sep 2007. ISSN 0194-4363. Disponível em: <<http://www.tandfonline.com/doi/abs/10.1080/01944360708977979>>. Citado 5 vezes nas páginas 26, 48, 49, 52 e 66.

CRENSHAW, K. Documento para o encontro de especialistas em aspectos da discriminação racial relativos ao gênero. *Estudos Feministas*, v. 1, p. 171–188, 2002. Citado na página 34.

CRESSWELL, T.; UTENG, T. P. Gendered Mobilities: towards an holistic understanding. In: UTENG, T. P.; CRESSWELL, T. (Ed.). *Gendered Mobilities*. 1^a. ed. Hampshire: Ashgate Publishing Limited, 2008. cap. 1, p. 1–12. Citado 3 vezes nas páginas 26, 44 e 45.

CUSSET, J.-M. Mobility deux roues et politique de transport à Ouagadougou et à Hanoi. In: INRETS (Ed.). *Mobilité et politiques de transport dans les villes en développement: journées spécialisées INRETS*. [S.l.: s.n.], 1997. p. 87–104. Citado na página 53.

DALMASO, R. C. *Identificação e caracterização de grupos de indivíduos segundo padrões de seqüências de atividades multidimensionais*. 153 p. Tese (Dissertação de Mestrado) — Escola Politécnica da Universidade de São Paulo, 2009. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/3/3138/tde-21072009-144859/en.php>>. Citado na página 83.

DARGAY, J. M. The effect of income on car ownership: Evidence of asymmetry. *Transportation Research Part A: Policy and Practice*, v. 35, n. 9, p. 807–821, 2001. ISSN 09658564. Citado na página 97.

DARGAY, J. M.; VYTHOULKAS, P. C. Estimation of a dynamic car ownership model: A pseudo-panel approach. *Journal of Transport Economics and Policy*, v. 33, n. 3, p. 287–302, 1999. Citado na página 97.

DEAR, M.; SCOTT, A. J. *Urbanization and Urban Planning in Capitalist Societies*. Nova Iorque: Methuen, 1981. Citado na página 38.

D'INCAO, M. Â. Mulher e Família Burguesa. In: PRIORE, M. D.; PINSKY, C. B. (Ed.). *História das Mulheres no Brasil*. 10^a. ed. São Paulo: Contexto, 2012. cap. 7, p. 677. Citado na página 47.

- ENDERS, J. Academic Staff Mobility in the European Community: The ERASMUS Experience. *Comparative Education Review*, The University of Chicago Press, v. 42, n. 1, p. 46–60, 1998. Disponível em: <<http://www.jstor.org/stable/1188786>>. Citado na página 39.
- ENLOE, C. H. *Bananas, Beaches and Bases: Making Feminist Sense of International Politics*. London: Pandora, 1989. Citado na página 39.
- EVERITT, B. S. et al. *Cluster Analysis*. 5. ed. West Sussex: Wiley Series in probability and statistics, 2011. 330 p. Citado 3 vezes nas páginas 130, 131 e 132.
- FAGNANI, J. Women's commuting patterns in the Paris region. *Tijdschrift voor Economische en Sociale Geografie*, v. 24, p. 12–24, 1983. Citado 3 vezes nas páginas 26, 52 e 66.
- FARIA, P. N. *Avaliação de Métodos para Determinação do Número Ótimo de Clusters em Estudo de Divergência Genética Entre Acessos de Pimenta*. 54 p. Tese (Mestrado) — Universidade Federal de Viçosa, 2009. Disponível em: <http://www.tede.ufv.br/tedesimplificado/tde{_\}arquivos/41/TDE-2009-07-01T150546Z-1744/Publico/textocompleto>. Citado na página 130.
- FÁVERO, L. P. et al. *Análise de Dados: modelagem multivariada para tomada de decisões*. Rio de Janeiro: Elsevier, 2009. 650 p. Citado 5 vezes nas páginas 129, 130, 131, 132 e 133.
- FOX, M. B. Working Women and Travel: The Access of Women to Work and Community Facilities. *Journal of the American Planning Association*, v. 49, n. 2, p. 156–170, 1983. Citado 5 vezes nas páginas 25, 26, 51, 52 e 66.
- FRAISSE, G. *El concepto filosófico de género*. 2001. Disponível em: <http://www.europarl.europa.eu/transl{_\}es/plataforma/pagina/celter/art2fraise>. Citado 2 vezes nas páginas 30 e 32.
- FREITAG, B. *Teorias da Cidade*. 2. ed. Campinas: Papirus, 2007. 190 p. Citado 2 vezes nas páginas 25 e 40.
- FROELICH, J. M. et al. Êxodo seletivo, masculinização e envelhecimento da população rural na região central do RS. *Ciência Rural*, v. 41, n. 9, p. 1674–1680, 2011. Disponível em: <<http://www.scielo.br/pdf/cr/v41n9/a10411cr3002.pdf>>. Citado na página 40.
- FROHLICK, S. 'I'm More Sexy Here': Erotic Subjectivities of Female Tourists in the 'Sexual Paradise' of the Costa Rican Caribbean. In: UTENG, T. P.; CRESSWELL, T. (Ed.). *Gendered Mobilities*. 1. ed. Hampshire: Ashgate Publishing Limited, 2008. cap. 9, p. 129–142. Citado na página 39.
- Fundação Perseu Abramo. *Mulheres brasileiras e gênero nos espaços público e privado*. 2010. 300 p. Disponível em: <<http://www.fpa.org.br/sites/default/files/pesquisaintegra.pdf>>. Citado na página 37.
- FURTADO, C. *Desenvolvimento e subdesenvolvimento*. Rio de Janeiro: Centro Celso Furtado / Contraponto, 2009. 234 p. ISBN 978-85-7866-019-2. Disponível em: <http://www.centrocelfurtado.org.br/interna.php?ID{_\}M=>>. Citado na página 34.

GERMANI, E. B. *Análise do Comportamento da Demanda por Transportes Utilizando Métodos de Alinhamento de Sequências Multidimensionais: Uma Aplicação à Região Metropolitana de São Paulo.* 120 p. Tese (Mestrado) — Universidade de São Paulo, 2005. Citado na página 82.

GILBERT, M. R. 'Race', space and power: The survival strategies of working poor women. *Annals of the Association of American Geographers*, v. 88, n. 4, p. 595–621, 1998. Citado na página 48.

GNANADESIKAN, R.; KETTENRING, J. R.; TSAO, S. L. Weighting and selection of variables. *Journal of Classification*, v. 12, p. 113–136, 1995. Citado na página 130.

GODDARD, T. B. et al. Voyage of the SS Minivan: Women's Travel Behavior in Traditional and Suburban Neighborhoods. *Journal of the Transportation Research Board*, n. 1956, p. 141–148, 2006. Disponível em: <<http://trb.metapress.com/content/2731180920015rml/fulltext.pdf>>. Citado na página 51.

GOUVEA, M. A.; La Plata, J. P. F. *Segmentos Médicos para a categoria de produtos cirúrgicos no Brasil*. São Paulo: [s.n.], 2006. Citado na página 133.

GRABOIS, J. et al. O habitat e a questão social no Noroeste Fluminense. v. 2, n. 21, p. 55–71, 2001. Citado na página 39.

GRECO, M. A.; GODOI, M. S. *Solidariedade Social e Tributação*. São Paulo: Dialética, 2005. Citado na página 44.

HAGERSTRAND, T. What about people in regional science? *Papers, Regional Science Association*, v. 24, p. 7–21, 1970. Citado na página 42.

Hair Jr, J. F. et al. *Análise Multivariada de Dados*. 5. ed. Porto Alegre: Bookman, 2005. Citado na página 132.

HANSON, S. *Gender, work and space*. London: Routledge, 1995. Citado 2 vezes nas páginas 26 e 48.

HANSON, S. Getting There: Urban Transportation in Context. In: *The Geography of Urban Transportation*. 2^a. ed. Nova Iorque: The Guilford Press, 1995. p. 478. Citado 3 vezes nas páginas 41, 42 e 44.

HANSON, S. Gender and mobility: new approaches for informing sustainability. *Gender, Place & Culture*, v. 17, n. 1, p. 5–23, feb 2010. ISSN 0966-369X. Disponível em: <<http://www.tandfonline.com/doi/abs/10.1080/09663690903498225>>. Citado 6 vezes nas páginas 26, 45, 47, 48, 54 e 59.

HANSON, S.; JOHNSTON, I. Gender Differences in Work-Trip Length: Explanations and Implications. *Urban Geography*, v. 6, n. 3, p. 193–219, may 1985. ISSN 0272-3638. Disponível em: <<http://www.tandfonline.com/doi/abs/10.2747/0272-3638.6.3.193>>. Citado na página 26.

HARAWAY, D. "Gênero" para um dicionário marxista: a política sexual de uma palavra. *Cadernos Pagu*, v. 22, p. 201–246, 2004. Disponível em: <<http://www.scielo.br/pdf/cpa/n22/n22a09.pdf>>. Citado 3 vezes nas páginas 31, 32 e 33.

- HARTIGAN, J. A. Statistical Theory in Clustering. *Journal of Classification*, v. 2, n. 1, p. 63–76, 1985. Disponível em: <<http://link.springer.com/article/10.1007/BF01908064>>. Citado na página 131.
- HARTIGAN, J. A.; WONG, M. A. A K-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, v. 28, n. 1, p. 100–108, 1979. Citado na página 133.
- HEILBORN, M. L. Usos e Abusos da Categoria de Gênero. In: HOLLANDA, H. H. O. B. de (Ed.). *Y Nosotras latinoamericanas? Estudos sobre Gênero e Raça*. São Paulo: Fundação Memorial da América Latina, 1992. p. 39–44. Disponível em: <http://www.clam.org.br/bibliotecadigital/uploads/publicacoes/114{_}1042{_}usoseabusosdacategoriadegenero.p>. Citado na página 33.
- HIRATA, H.; GUIMARÃES, N. A. *Cuidado e cuidadoras: as várias faces do trabalho do care*. São Paulo: Atlas, 2012. 236 p. Citado na página 52.
- HJORTHOL, R. J. Same city, different options: An analysis of the work trips of married couples in the metropolitan area of Oslo. *Journal of Transport Geography*, v. 8, p. 213–220, 2000. Citado 5 vezes nas páginas 25, 26, 48, 51 e 54.
- HODGE, D. My fair share: equity issues in urban transportation. In: HANSON, S. (Ed.). *The Geography of Urban Transportation*. Nova Iorque: The Guilford Press, 1995. Citado na página 25.
- HOFFMAN, D. M. Changing Academic Mobility Patterns and International Migration: What Will Academic Mobility Mean in the 21st Century? *Journal of Studies in International Education*, v. 13, n. 3, p. 347–364, jul 2008. ISSN 1028-3153. Disponível em: <<http://jsi.sagepub.com/cgi/doi/10.1177/1028315308321374>>. Citado na página 39.
- HOOKS, B. *Yearning: Race, gender, and cultural politics*. Boston: South End Press, 1990. 248 p. Citado na página 34.
- HOWE, A.; O'CONNOR, K. Travel to work and labour force participation of men and women in an Australian metropolitan area. *Professional Geographer*, v. 34, p. 50–64, 1982. Citado 2 vezes nas páginas 26 e 53.
- IEMA, I. d. E. e. M. A. *A Bicicleta e as Cidades: como inserir a bicicleta na política de mobilidade urbana*. 2^a. ed. São Paulo: IEMA, 2010. 84 p. Disponível em: <<http://www.energiaeambiente.org.br/>>. Citado na página 41.
- JOHNSTON-ANUMONWO, I. The Influence of Household Type on Gender Differences in Work Trip Distance". *Professional Geographer*, v. 44, n. 2, p. 161–169, 1992. Citado 3 vezes nas páginas 26, 52 e 66.
- JONES, P. M. Activity approaches to understanding travel behavior. In: (eds) . Lexington Books, Cap. 13, p.253-266. In: STOPHER, P. R.; MEYBURG, A. H.; BRÖG, W. (Ed.). *New horizons in travel behavior*. Lexington Books, 1981. p. 253–266. Disponível em: <<http://trid.trb.org/view.aspx?id=320615>>. Citado na página 60.
- KARLAFTIS, M.; GOLIAS, J. Automobile Ownership, Households Without Automobiles, and Urban Traffic Parameters: Are They Related? *Transportation Research Record*, v. 1792, n. 1, p. 29–35, 2002. ISSN 0361-1981. Citado na página 97.

- KEHL, M. R. *Deslocamentos do Feminino- A Mulher Freudiana na Passagem para a Modernidade*. Rio de Janeiro: Imago, 1998. Citado na página 36.
- KERGOAT, D. Division sexuelle du travail et rapports sociaux de sexe. In: HIRATA, H. et al. (Ed.). *Dictionnaire critique du féminisme*. 2. ed. Paris: Presses Universitaires de France, 2004. p. 35–44. Citado 2 vezes nas páginas 33 e 36.
- KINGHAM, S.; DICKINSON, J.; COPSEY, S. Travelling to work: will people move out of their cars. *Transport Policy*, v. 8, p. 151–160, 2001. Citado na página 25.
- KOSTYNIUK, L. P.; KITAMURA, R. Trip chains and activity sequences: test of temporal stability. *Transportation Research Record*, v. 987, p. 29–39, 1984. Citado na página 60.
- KÜNZLER, J. *Familiale Arbeitsteilung: Die Beteiligung von Männern an der Hausarbeit*. Bielefeld: Kleine, 1994. Citado na página 48.
- LEE, B. S.; MCDONALD, J. Determinants of commuting time and distance for Seoul residents: the impact of family status on the commuting of women. *Urban Studies*, v. 40, n. 7, p. 1283–1302, jun 2003. ISSN 0042-0980. Disponível em: <<http://usj.sagepub.com/cgi/doi/10.1080/0042098032000084604>>. Citado 3 vezes nas páginas 26, 53 e 66.
- LÉVI-STRAUSS, C. *As Estruturas Elementares do Parentesco*. 6^a. ed. Petrópolis: Vozes, 2010. 544 p. Disponível em: <<http://classicos12011.files.wordpress.com/2011/03/lc3a9vi-strauss-claude-as-estruturas-elementares-do-parentesco.pdf>>. Citado na página 33.
- LEWIS, D. *Economic Perspectives on Transport and Equality*. Leipzig, 2011. 30 p. Disponível em: <<http://www.internationaltransportforum.org/>>. Citado na página 25.
- MAGIDSON, J. The CHAID Approach to Segmentation Modeling: Chi-squared Automatic Interaction Detection. In: BAGOZZI, R. (Ed.). *Advanced Methods of Marketing Research*. Cambridge: Blackwell Publishing, 1994. p. 118–159. Citado na página 60.
- MAHMASSANI, H. S. Some comments on activity-based approaches to the analysis and prediction of travel behavior. *Transportation*, v. 15, n. 1-2, p. 35–40, 1988. Citado na página 60.
- MALHOTRA, N. K. *Pesquisa de Marketing: Uma Orientação*. 3^a. ed. Porto Alegre: Bookman, 2001. 720 p. Citado 2 vezes nas páginas 57 e 58.
- MANDEL, J. Mobility matters: Women's livelihood strategies in Porto Novo, Benin. *Gender, Place & Culture*, v. 11, n. 2, p. 257–87, 2004. Disponível em: <<http://www.tandfonline.com/doi/pdf/10.1080/0966369042000218482>>. Citado na página 48.
- MARQUES, M. I. M. O conceito de espaço rural em questão. *Terra Livre*, v. 18, n. 19, p. 95–112, 2002. Citado na página 40.
- MAXWELL, A. E. *Multivariate Analysis in Behavioural Research*. 1. ed. London: Chapman & Hall, 1977. Citado na página 131.
- MCGUCKIN, N.; MURAKAMI, E. Examining Trip-Chaining Behavior. *Transportation Research Record*, v. 1693, n. 99, p. 79–85, 1995. Citado na página 54.

- MCGUCKIN, N.; ZMUD, J.; NAKAMOTO, Y. Trip-Chaining Trends in the United States: Understanding Travel Behavior for Policy Making. *Transportation Research Record: Journal of the Transportation Research Board*, v. 1917, p. 199–204, 2005. Citado 3 vezes nas páginas 26, 54 e 66.
- MCLAFFERTY, S.; PRESTON, V. Gender, race and commuting among service sector workers. *The Professional Geographer*, v. 43, p. 1–14, 1991. Citado 3 vezes nas páginas 26, 48 e 49.
- MCLAFFERTY, S.; PRESTON, V. Spatial Mismatch and Labor Market Segmentation for African-American and Latina Women. *Economic Geography*, v. 68, n. 4, p. 406–431, 1992. Citado 3 vezes nas páginas 26, 48 e 49.
- MEAD, M. *Sexo e temperamento em três sociedades primitivas*. 4^a. ed. São Paulo: Perspectiva, 2000. 316 p. Disponível em: <<http://www.livrariacultura.com.br/scripts/resenha/resenha.asp?nitem=72303><http://pt.scribd.com/doc/178229042/Resumo-Sexo-e-Temperamento-Margareth-Mead>>. Citado na página 31.
- METRÔ-SP. *Pesquisa Origem Destino 1977*. São Paulo, 1977. Citado 4 vezes nas páginas 62, 63, 64 e 81.
- METRÔ-SP. *Pesquisa Origem Destino 1987*. São Paulo, 1987. Citado 4 vezes nas páginas 62, 63, 64 e 81.
- METRÔ-SP. *Pesquisa Origem Destino 1997*. São Paulo, 1997. Citado 4 vezes nas páginas 62, 63, 64 e 81.
- METRÔ-SP. *Pesquisa Origem Destino 2007*. São Paulo, 2007. Citado 4 vezes nas páginas 62, 63, 64 e 81.
- METRÔ-SP. *Pesquisa Origem Destino 2007 - Região Metropolitana de São Paulo: Síntese das Informações*. São Paulo, 2008. 83 p. Citado 2 vezes nas páginas 61 e 62.
- METZ, D. Demographic determinants of daily travel demand. *Transport Policy*, Elsevier, v. 21, p. 20–25, may 2012. ISSN 0967070X. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0967070X1200008X>>. Citado 2 vezes nas páginas 25 e 39.
- MICHAELIS. *Dicionário Prático da Língua Portuguesa*. São Paulo: Melhoramentos, 2011. Disponível em: <<http://michaelis.uol.com.br/moderno/portugues/index.php>>. Citado na página 39.
- MINGIONE, E.; PUGLIESE, E. A difícil delimitação do "urbano" e do "rural": alguns exemplos e implicações teóricas. *Revista Crítica de Ciências Sociais*, n. 22, p. 83–99, 1987. Citado na página 40.
- MOJENA, R. Hierarchical Grouping Methods and Stopping Rules: an Evaluation. *The Computer Journal*, v. 20, n. 4, p. 359–363, 1977. Disponível em: <<http://comjnl.oxfordjournals.org/content/20/4/359.full.pdf+html>>. Citado na página 131.
- MORAES, M. L. Q. de. Usos e Limites da categoria gênero. *Cadernos Pagu*, Campinas, n. 11, p. 99–106, 1998. Disponível em: <www.bibliotecadigital.unicamp.br/document/?down=51203>. Citado 2 vezes nas páginas 30 e 36.

MÜLLNER, D. Modern hierarchical, agglomerative clustering algorithms. n. 1973, p. 29, 2011. Disponível em: <<http://arxiv.org/abs/1109.2378>>. Citado na página 129.

MÜLLNER, D. Fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, v. 53, n. 9, p. 1–18, 2013. Disponível em: <<http://www.jstatsoft.org/v53/i09/>>. Citado na página 129.

MURTAGH, F.; LEGENDRE, P. Ward’s Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward’s Criterion? *Journal of Classification*, v. 31, p. 274–295, 2014. Citado na página 129.

NELSON, C. C. *Literature of the women’s suffrage campaign in England*. [S.l.]: Broardview Press, 2002. Citado na página 31.

OLIVEIRA, A. G. de. *Efeitos das composições familiares na mobilidade dos idosos – uma análise multinível*. 128 p. Tese (Mestrado) — Universidade de Brasília, 2014. Citado na página 88.

ORTÚZAR, J. d. D.; WILLUMSEN, L. G. *Modelling transport*. New York: John Wiley and Sons, 1994. Citado 3 vezes nas páginas 60, 87 e 88.

PEIXOTO, N. M. O. *A evolução Temporal da Mobilidade da População na Região Metropolitana de Porto Alegre, entre 1986 e 1997*. 160 p. Tese (Mestrado) — Universidade Federal do Rio Grande do Sul, 2002. Citado na página 87.

PFEIFFER, L. M.; STRAMBI, O. Análise e modelagem da evolução temporal da posse de autos na Região Metropolitana de São Paulo. *Revista Transportes*, XIII, n. 1, p. 21–29, 2005. Citado 2 vezes nas páginas 97 e 98.

PINSKY, C. B.; PEDRO, J. M. Mulheres – Igualdade e Especificidade. In: PINSKY, J.; PINSKY, C. B. (Ed.). *História da Cidadania*. 2ª. ed. São Paulo: Contexto, 2003. Citado na página 35.

PINTO, L. P. Mulheres brasileiras na mídia portuguesa. *Cadernos Pagu*, v. 23, p. 229–257, 2004. Disponível em: <<http://www.scielo.br/pdf/cpa/n23/n23a08.pdf>>. Citado na página 34.

PISCITELLI, A. Gênero: a história de um conceito. In: ALMEIDA, H. B. de; SZWAKO, J. (Ed.). *Diferenças, Igualdade*. São Paulo: Berlendis & Vertecchia, 2009. cap. 4, p. 239. Citado 4 vezes nas páginas 31, 32, 33 e 35.

POLK, M. Are women potentially more accommodating than men to a sustainable transportation system in Sweden? *Transportation Research Part D: Transport and Environment*, v. 8, p. 75–95, 2003. Disponível em: <http://ac.els-cdn.com/S1361920902000342/1-s2.0-S1361920902000342-main.pdf?{|_}tid=74a6c252-78c6-11e4-a00c-00000aacb362{&}acdnat=1417375628{_|}1084fe2f2f94333a3b7d59a>. Citado 4 vezes nas páginas 25, 26, 51 e 52.

R Core Team. *R: A Language and Environment for Statistical Computing*. Viena: R Foundation for Statistical Computing, 2011. Citado na página 129.

- RAJU, S. Gender and empowerment: Creating “thus far and no further” supportive structures. A case from India. In: NELSON, L.; SEAGER, J. (Ed.). *A Companion to Feminist Geography*. Oxford: Blackwell Publishing, 2005. cap. 14. Disponível em: <<http://onlinelibrary.wiley.com/doi/10.1002/9780470996898.ch14/pdf>>. Citado 2 vezes nas páginas 26 e 48.
- RAY, P. O. Woman Suffrage in Foreign Countries. *American Political Science Review*, v. 12, n. 3, p. 469–474, 1918. Citado na página 31.
- RICHARDSON, B. C. Sustainable transport: analysis frameworks. *Journal of Transport Geography*, v. 13, n. 1, p. 29–39, mar 2005. ISSN 09666923. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0966692304000857>>. Citado na página 25.
- ROOT, A.; SCHINTLER, L. Women, motorization and the environment. *Transportation Research Part D: Transport and Environment*, v. 4, n. 5, p. 353–355, sep 1999. ISSN 13619209. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S1361920999000127>>. Citado 5 vezes nas páginas 26, 50, 52, 54 e 66.
- ROSENBLOOM, S. Editorial: The Need for Study of Women’s Travel Issues Recently. *Transportation*, v. 7, n. 1978, p. 347–350, 1978. Citado 3 vezes nas páginas 26, 47 e 48.
- ROSENBLOOM, S. Trends in women’s travel patterns. In: *Women’s Travel Issues Second National Conference*. [s.n.], 2000. p. 16–34. Disponível em: <<http://trid.trb.org/view.aspx?id=720092>>. Citado na página 52.
- ROSENBLOOM, S. The mobility needs of older Americans. *Taking the High Road: A Transportation Agenda of Strengthening Metropolitan Areas*, Washington DC, p. 227–54, 2003. Disponível em: <http://www.brookings.edu/~/media/research/files/reports/2003/7/transportationrosenbloom/20030807{_\}rosenbloo>. Citado na página 52.
- ROSENBLOOM, S. Understanding Women’s and Men’s Travel Patterns: The Research Challenge. In: TRANSPORTATION RESEARCH BOARD OF THE NATIONAL ACADEMIES. *Research on Women’s Issues in Transportation: Volume 1 - Conference Overview and Plenary Papers*. Washington DC: National Research Council, 2006. p. 7–28. ISBN 0309099560. Disponível em: <<http://onlinepubs.trb.org/Onlinepubs/conf/CP35v1.pdf>>. Citado na página 26.
- RUBIN, G. S. The Traffic in Women: Notes on the ‘Political Economy’ of Sex. In: REITER, R. (Ed.). *Toward an Anthropology of Women*. Nova Iorque: Monthly Review Press, 1975. p. 157–211. Disponível em: <<http://summermeetings2013.files.wordpress.com/2013/04/rubin-traffic.pdf>>. Citado 2 vezes nas páginas 32 e 33.
- RUEDA, S. et al. *Libro verde de medio ambiente urbano – Tomo I*. Barcelona, 2007. 473 p. Disponível em: <<http://bcnecologia.net/es/proyectos/libro-verde-de-medio-ambiente-urbano-tomo-i-y-ii>>. Citado na página 25.
- RYAN, J. M.; HAN, G. Vehicle-Ownership Model Using Family Structure and Accessibility Application to Honolulu, Hawaii. *Transportation Research Record*, v. 1676, n. 99, p. 1–10, 1999. ISSN 0361-1981. Citado na página 97.
- SÃO PAULO (ESTADO). *Lei nº 1.139, de 16 de junho de 2011. Reorganiza a Região Metropolitana da Grande São Paulo, cria o respectivo Conselho de Desenvolvimento e dá providências correlatas*. 2011. Citado na página 29.

SAFFIOTI, H. I. B. *A Mulher na Sociedade de Classes: Mito e Realidade*. 3^a. ed. São Paulo: Expressão Popular, 2013. 528 p. Citado na página 34.

SCHWANEN, T.; DIJST, M.; DIELEMAN, F. M. A microlevel analysis of residential context and travel time. *Environment and Planning A*, v. 34, n. 8, p. 1487–1507, 2002. ISSN 0308-518X. Disponível em: <<http://www.envplan.com/abstract.cgi?id=a34159>>. Citado 4 vezes nas páginas 26, 52, 53 e 66.

SCOTT, J. W. Gender: A Useful Category of Historical Analysis. *The American Historical Review*, IE/UFRGS, Porto Alegre, v. 91, n. 5, p. 1053–1075, 1986. Disponível em: <<http://www.jstor.org/stable/1864376>>. Citado 2 vezes nas páginas 34 e 47.

SHARMA, S. *Applied Multivariate Techniques*. New York: John Wiley, 1996. 493 p. Citado na página 131.

SHEARMUR, R. Travel from home: An economic geography of commuting distances in Montreal. *Urban Geography*, v. 27, n. 4, p. 330–59, 2006. Citado na página 63.

SILVEY, R.; ELMHIRST, R. Engendering Social Capital: Women Workers and Rural-Urban Networks in Indonesia's Crisis. *World Development*, v. 31, n. 5, p. 865–879, may 2003. ISSN 0305750X. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0305750X03000135>>. Citado 2 vezes nas páginas 26 e 48.

SILVEY, R. M. Stigmatized spaces: gender and mobility under crisis in South Sulawesi, Indonesia. *Gender, Place & Culture*, v. 7, n. 2, p. 143–61, 2000. Citado na página 39.

SOARES, V.; PINHEIRO, L. S. *Dados das desigualdades: gênero e raça*. [S.l.], 2003. 15 p. Disponível em: <http://www.planalto.gov.br/seppir/pesquisas{_\}indicadores/raca/presskit/unifem> Citado na página 38.

STEG, L. Car use: lust and must. Instrumental, symbolic and affective motives for car use. *Transportation Research Part A: Policy and Practice*, v. 39, n. 2-3, p. 147–162, feb 2005. ISSN 09658564. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0965856404001016>>. Citado na página 25.

STELLA, P. R. Palavra. In: BRAIT, B. (Ed.). *Bakthin - Conceitos-Chave*. São Paulo: Contexto, 2005. cap. 10. Citado na página 30.

STOLKE, V. La mujer es puro cuento: la cultura del género. *Revista Estudos Feministas*, v. 12, n. 2, p. 77–105, 2004. Citado na página 32.

STRAMBI, O.; BILT, K.-A. van de. Trip Generation Modeling Using CHAID, a Criterion-Based Segmentation Modeling Tool. *Transportation Research Record*, v. 1645, n. 98, p. 24–31, 1998. Citado 2 vezes nas páginas 53 e 60.

STRAMBI, O.; Van de Bilt, K. Mobility in São Paulo: a temporal perspective. *IRF ROAD WORLD CONGRESS*, 14, Paris, 2001. Citado na página 87.

TABAK, F. *A mulher brasileira no Congresso Nacional*. 1^a. ed. Brasília: Câmara dos Deputados, Centro de Documentação e Informação, Coordenação de Publicações, 1989. Citado na página 35.

- TERTOOLEN, G.; KREVELD, D. V.; VERSTRATEN, B. Psychological resistance against attempts to reduce private car use. *Transportation Research Part A: Policy and Practice*, v. 32, n. 3, p. 171–181, 1998. Citado na página 25.
- TREMBLAY, K. Academic Mobility and Immigration. *Journal of Studies in International Education*, v. 9, n. 3, p. 196–228, sep 2005. ISSN 1028-3153. Disponível em: <<http://jsi.sagepub.com/cgi/doi/10.1177/1028315305277618>>. Citado na página 39.
- URRY, J. Connections. *Environment and Planning D*, v. 22, p. 143–61, 2004. Citado 2 vezes nas páginas 44 e 45.
- Van de Bilt, K.-A. *Análise de taxas de produção de viagens urbanas utilizando modelagem de segmentação*. 131 p. Tese (Mestrado) — Universidade de São Paulo, 1997. Citado 2 vezes nas páginas 102 e 106.
- VANCE, C.; IOVANNA, R. Gender and the Automobile: Analysis of Nonwork Service Trips. *Transportation Research Record: Journal of the Transportation Research Board*, v. 2013, p. 54–61, dec 2007. ISSN 0361-1981. Disponível em: <<http://trb.metapress.com/openurl.asp?genre=article&id=doi:10.3141/2013>>. Citado 3 vezes nas páginas 50, 51 e 66.
- VASCONCELLOS, E. *Transporte Urbano, Espaço e Equidade: Análise das Políticas Públicas*. 1ª. ed. São Paulo: Annablume, 2001. 218 p. Citado 7 vezes nas páginas 25, 42, 44, 45, 53, 66 e 88.
- VASCONCELLOS, E. *Mobilidade Urbana e Cidadania*. Rio de Janeiro: Senac, 2012. 213 p. ISBN 978-85-7458-318-1. Citado 4 vezes nas páginas 25, 26, 41 e 100.
- VEIGA, J. E. da. *Cidades imaginárias: o Brasil é menos urbano do que se calcula*. Campinas: Autores, 2002. Citado na página 40.
- VESPUCCI, K. M. *Sequências de atividades e cadeias de viagens na Região Metropolitana de São Paulo – uma investigação comparativa do período 1987-1997*. 191 p. Tese (Mestrado) — Universidade de São Paulo, 2003. Citado 3 vezes nas páginas 81, 82 e 139.
- Ward Jr, J. H. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American statistical association*, v. 58, n. 301, p. 236–244, 1963. Disponível em: <<http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845>>. Citado na página 132.
- WILLARD, F. E. *Wheel Within a Wheel: How I Learned to Ride a Bicycle, With Some Reflections by The Way*. Chicago: Fleming H. Revell Company, 1895. 75 p. Disponível em: <<https://archive.org/details/wheelwithinwheel00williala>>. Citado 2 vezes nas páginas 46 e 47.

Anexos

ANEXO A – Correspondência entre Zonas
das Pesquisas Origem Destino por meio das
Unidades de Correspondência entre Zonas
(UCOD)

UCOD	Nome	Zoneamento da OD 1977										Zoneamento da OD 1987													
		1	6	7	9	10	24	5	8	17	18	19	20	21	22	23	39	1	2	3	4	5	6	7	8
1	Sé/República							44	65	66	89							9	10						
2	Liberdade/Bela Vista							25	26	27	28	45	46	47	67			11	12	13	14	15	16	17	
3	Consolação/Perdizes							2	3	11	12	13	29	30	31	32		18	19	20	21				
4	Barra Funda/Bom Retiro/Santa Cecília							14	15	33	34	35	54					22	23	24	25	26	27		
5	Brás/Pari/Belém							4	16	36	37	58	81					28	29	30					
6	Mooca/Água Rasa							38	40	41	60	61						31	32	33					
7	Cambuci/Ipiranga							59	83									34							
8	Vila Mariana							10	85	115	116							35	36	37					
9	Saúde							11	156	189								38	39						
10	Moema/Campo Belo							12	62	117								40	41	42	43	44			
11	Santo Amaro							13	42	43	63	64						45	46	47					
12	Itaim Bibi							14	86	87	88	121	122	125				48	49	50	51				
13	Jardim Paulista							15	90	91	123	124	126	127				52	53	54	55	56	57		
14	Pinheiros/Alto de Pinheiros							16	161	162	163	164						58	59						
15	Lapa/Leopoldina							17	92	128	129	165						60	61	62					
16	Pirituba/Jaguara/São Domingos							18	166	168								63	64						
17	Freguesia do Ó/Limão							19	48	68	69	70	93	94	95	130		65	66	67	68	69			
18	Brasilândia							20	131	167								70	71	72	73				
19	Santana/Casa Verde							21	96	97	132							74	75	76	77				
20	Mandaqui/Cachoeirinha							22	98	99	133							78							
21	Tremembé/Tucuruvi							23	49	50	51	71	72					79	80	81	82				
22	Jacanã/Vila Medeiros							24	52	53	73	74	103	143				83	84	85					
23	Vila Maria/Vila Guilherme							25	75	104	105	243						86	87						
24	Tatuapé/Carrão							26	55	56	57	78						88							
25	Vila Formosa/Aricanduva							27	76	77								89							
26	Vila Prudente/São Lucas							28	101	136	137	174						90	91						
29	Penha/Ponte Rasa							29	102	138	139	140	177					92	93	94					
30	Vila Matilde/Artur Alvim							31	141	142								95	96						
31	Cidade Líder/Parque do Carmo							32	144	145								97	98						
32	São Mateus/Guatemi/São Rafael							33	146									99							
33	Vila Jacuí/São Miguel Paulista							34	173	175								100	101						
35	Itaquera/José Bonifácio							35	176	178								102	103						
36	Jardim Helena/Vila Curuçá/Itaim Paulista							36	172	206	207							104	105	106					
37	Lajeado/Guaianases/Cidade Tiradentes							37	179									107	108	109					
38	Cursino/Sacomã							38	82	111	112							110	111						
39	Jabaquara							39	113	114	154							112	113						
40	Campo Grande/Cidade Ademar/Pedreira							40	155	187	188							114	115	116	117	118			
41	Socorro/Cidade Dutra							41	220									119	120						
42	Jd. São Luís/Capão Redondo/Jd. Ângela							42	190	191								121	122	123					
43	Campo Limpo/Vila Andrade							43	157									124	125	126					
44	Butantã/Morumbi/Vila Sônia							44	118	119	120	158						127	128	129	130				
45	Jaquaré/Rio Pequeno/Raposo Tavares							45	159	160	194							131	132	133					
46	Taboão da Serra							46	192	193								134							
47	Osasco(centro)							47	196	197								135	136						
48	Novo Osasco							48	195	198								137	138						
49	Mutitinga/Presidente Altino							49	199	200								139	140	141					
50	Carapicuíba							50	225	226								142	143	144					
51	Guarulhos(centro)							51	100	134	135							145	146	147					
52	F. de Vasconcelos/Poá/Itaquaquecetuba/Suzano							52	205	208	209	210	211	240				148	149	150	151	152	153	154	155
53	Mogi das Cruzes(centro)							53	237	238	239							157	158	159	160				
54	São Caetano do Sul							54	80	108	109	110						161	162						
55	Santo André(centro)							55	79	106	107	147	148	149	150			163	164	165	166				
56	Vila Pires/Pedroso							56	182	183	214							167	168						
57	Mauá							57	180	181								169	170	171	172				
58	Rudge Ramos							58	151	152								173							
59	São Bernardo do Campo(centro)							59	184	185	215	217						174	175	176					
60	Diadema							60	153	186								177	178	179					
61	Grajaú/Parelheiros/Marsilac							61	219									180	181	182					
62	Jaraguá/Perus/Anhangüera							62	201	202	231	233						183	184	185	186	187			
63	Macro-zona Norte (1)							63	203	232	234							188	189	190	191	192	193	194	195
64	Macro-zona Nordeste (2)							64	169	170	171	204						202	203	204	205	206	207	208	209
65	Macro-zona Leste (3)							65	235	236	241							214	215	216	217	218	219	220	221
66	Macro-zona Sudeste (4)							66	212	213	216	218	242					222	223	224	225	226	227	228	229
67	Macro_zona Sudoeste (5)							67	221	222	223							231	232	233	234	235	236	237	238
	Macro_zona Oeste (6)							67	224	227	228	229	230					240	241	242	243	244	245	246	247
																	248	249	250	251	252	253	254	255	

UCOD	Nome	Zoneamento da OD 1997					
		1	2	3	4	5	6
1	Sé/República	11	12	13	14	15	16
2	Liberdade/Bela Vista	17	36	37	74	76	
3	Consolação/Perdizes	7	18	19	38	39	40
4	Barra Funda/Bom Retiro/Santa Cecília	8	20	21	22	23	24
5	Brás/Pari/Belém	25	26	27	53	55	
6	Mococa/Águia Rasa	9	10	28	29	58	59
7	Cambuci/Ipiranga	30	31	32	62	64	
8	Vila Mariana	63	65	109	111		
9	Saúde	66	67	112	114	116	117
10	Moema/Campo Belo	190	191	192	193	118	119
11	Santo Amaro	69	121	122	123	124	
12	Itaim Bibi	33	34	35	68		
13	Jardim Paulista	70	71	72	73	75	130
14	Pinheiros/Alto de Pinheiros	77	78	79	132	133	
15	Lapa/Leopoldina	136	137	138	139	140	
16	Pirituba/Jaguara/São Domingos	80	81	141	142	143	
17	Freguesia do O/Limão	144	145	210	211		
18	Brasilândia	42	43	44	45	82	83
19	Santana/Casa Verde	146	147	148	149	212	213
20	Mandaqui/Cachoeirinha	86	150	151	152	153	155
21	Tremembé/Tucuruvi	144	145	210	211	214	215
22	Jaçanã/Vila Medeiros	87	88	89	154	156	
23	Vila Maria/Vila Guilherme	46	47	48	49	50	
24	Tatuapé/Carrão	51	52	95	96		
25	Vila Formosa/Aricanduva	54	97	169	171		
26	Vila Prudente/São Lucas	56	57	98	99	100	175
27	Sapopemba	173	174	176	235		
28	Cangaíba/Ermelino Matarazzo	92	159	160	161	218	
29	Penha/Ponte Rasa	91	93	162	163	164	
30	Vila Matilde/Artur Alvim	94	165	166	167		
31	Cidade Líder/Parque do Carmo	168	170	228	229	230	231
32	São Mateus/Iguatemi/São Rafael	172	232	233	234	236	294
33	Vila Jacuí/São Miguel Paulista	219	220	221	222	225	
34	Itaquera/José Bonifácio	223	224	226	227	289	292
35	Jardim Helena/Vila Curuçá/Itaim Paulista	279	280	281	282	283	284
36	Lajeado/Guaianases/Cidade Tiradentes	286	287	290	291	293	
37	Cursino/Sacomã	60	61	103	104	106	107
38	Jabaquara	110	113	115	182		
39	Campo Grande/Cidade Ademar/Pedreira	183	184	185	186	187	188
40	Socorro/Cidade Dutra	246	247	248	249	250	303
41	Jd. São Luís/Capão Redondo/Jd. Ângela	251	252	253	254	255	304
42	Campo Limpo/Vila Andrade	194	195	196	256	257	258
43	Butantã/Morumbi/Vila Sônia	125	126	127	128	129	131
44	Jacuá/Rio Pequeno/Raposo Tavares	134	135	201	202	262	263
45	Taboão da Serra	260	261				
46	Osasco(centro)	203	204				
47	Novo Osasco	265	266				
48	Mutinga/Presidente Altino	205	206	207			
49	Carapicuíba	267	313				
50	Guarulhos(centro)	90	157	158			
51	F. de Vasconcelos/Poá/Itaquaquecetuba/Suzano	288	329	331	332	334	335
52	Mogi das Cruzes(centro)	339	371	372			
53	São Caetano do Sul	102	105	178			
54	Santo André(centro)	177	179	237			
55	Vila Pires/Pedroso	239	298				
56	Mauá	238	297	342			
57	Rudge Ramos	180					
58	São Bernardo do Campo(centro)	240	241	299			
59	Diadema	181	242	243			
60	Grajaú/Parelheiros/Marsilac	301	302	347	348	349	382
61	Jaraguá/Perus/Anhanguera	208	209	270	271	272	
62	Macro-zona Norte (1)	273	274	322	323	324	325
63	Macro-zona Nordeste (2)	216	217	275	276	277	278
64	Macro-zona Leste (3)	333	341	369	370	373	374
65	Macro-zona Sudeste (4)	300	343	344	345	346	378
66	Macro_zona Sudoeste (5)	309	310	311	350	351	352
67	Macro_zona Oeste (6)	268	269	312	314	315	316

UCOD	Nome	Zoneamento da OD 2007																												
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	Sé/República	20	21	22	23	24	25	26	27	28	29																			
2	Liberdade/Bela Vista	30	31	32	33	34	83	84	85	86	87	88	89																	
3	Consolação/Perdizes	7	8	9	35	36	37	90	91	92	93																			
4	Barra Funda/Bom Retiro/Santa Cecília	10	11	12	13	14	15	16	17	38	39	40	41	42																
5	Brás/Pari/Belém	43	44	45	46	47	48																							
6	Mooca/Água Rasa	18	19	222	223	224	225	226	227	228																				
7	Cambuci/Ipiranga	49	50	51	52	53	54	55	56	57																				
8	Vila Mariana	58	59	60	61																									
9	Saúde																													
10	Moema/Campo Belo	62	63	64	65	66	251	252	253	254	255																			
11	Santo Amaro	281	282	283	284																									
12	Itaim Bibi	67	68	69	70	71	72																							
13	Jardim Paulista	73	74	75	76	77	78																							
14	Pinheiros/Alto de Pinheiros	79	80	81	82	94	95	96																						
15	Lapa/Leopoldina	97	98	99	100	101	102	103	104	105	106	107																		
16	Pirituba/Jaguara/São Domingos	108	109	110	111	112																								
17	Freguesia do Ó/Limão	123	124	125	136	137																								
18	Brasilândia	119	120	121	122																									
19	Santana/Casa Verde	126	127	128	129	130	131	132	133	134	135																			
20	Mandaqui/Cachoeirinha	138	139	140	141	142	143																							
21	Tremembé/Tucuruvi	144	145	146	147	148	149	150	151																					
22	Jaçanã/Vila Medeiros	152	153	154	155	156																								
23	Vila Maria/Vila Guilherme	157	158	159	160	161	162																							
24	Tatuapé/Carrão	163	164	165	166	197	198																							
25	Vila Formosa/Aricanduva	199	200	201	202																									
26	Vila Prudente/São Lucas	237	238	239	240	241	242																							
27	Sapopemba	243	244	245	246																									
28	Cangaíba/Ermelino Matarazzo	170	171	172	173	174	175	176																						
29	Penha/Ponte Rasa	167	168	169	177	178																								
30	Vila Matilde/Artur Alvim	203	204	205	206																									
31	Cidade Líder/Parque do Carmo	207	208	209	210	211	212																							
32	São Mateus/Iguatemi/São Rafael	218	219	220	221	247	248	249	250																					
33	Vila Jacuí/São Miguel Paulista	179	180	185	186	187																								
34	Itaquera/José Bonifácio	181	182	183	184	213	214																							
35	Jardim Helena/Vila Curuçá/Itaim Paulista	188	189	190	191	194	195	196																						
36	Lajeado/Guaianases/Cidade Tiradentes	192	193	215	216	217																								
37	Cursino/Sacomã	229	230	231	232	233	234	235	236																					
38	Jabotiguara	256	257	258	259																									
39	Campo Grande/Cidade Ademar/Pedreira	260	261	262	263	264	265	266	267	268																				
40	Socorro/Cidade Dutra	269	270	271	272	273	274																							
41	Jd. São Luís/Capão Redondo/Jd. Ângela	285	286	287	288	289	290	291	292	293	294																			
42	Campo Limpo/Vila Andrade	299	300	301	302	303	304	305																						
43	Butantã/Morumbi/Vila Sônia	295	296	297	298	306	307	308	309	317	318	319	320																	
44	Jaquaré/Rio Pequeno/Raposo Tavares	310	311	312	313	314	315	316																						
45	Taboão da Serra	411	412																											
46	Osasco (centro)	436	437	438	439																									
47	Novo Osasco	440	441	442																										
48	Mutings/Presidente Altino	443	444	445	446	447																								
49	Carapicuíba	432	433	434	435																									
50	Guarulhos (centro)	336	337	338	339	340	341																							
51	F. de Vasconcelos/Poá/Itaquaquecetuba/Suzano	354	355	356	357	358	359	360	361	362	363																			
52	Mogi das Cruzes (centro)	366	367	368	369	370																								
53	São Caetano do Sul	379	380	381	382																									
54	Santo André (centro)	383	384	385	386	387	388																							
55	Vila Pires/Pedroso	389	390																											
56	Mauá	392	393	394	395																									
57	Rudge Ramos	402																												
58	São Bernardo do Campo (centro)	401	403	404																										
59	Diadema	408	409	410																										
60	Grajaú/Parelheiros/Marsilac	275	276	277	278	279	280																							
61	Jaraguá/Perus/Anhanguera	113	114	115	116	117	118																							
62	Macro-zona Norte (1)	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335														
63	Macro-zona Nordeste (2)	342	343	344	345	346	347	348	349	350	351	352	353																	
64	Macro-zona Leste (3)	364	365	371	372	373	374	375	376	377	378																			
65	Macro-zona Sudeste (4)	391	396	397	398	399	400	405	406	407																				
66	Macro_zona Sudoeste (5)	413	414	415	416	417	418	419	420	421	422	423																		
67	Macro_zona Oeste (6)	424	425	426																										

ANEXO B – Mapas de Zonas e Subzonas
das Pesquisas Origem Destino (1977, 1987,
1997 e 2007)

ANEXO C – Layouts dos bancos de dados
das Pesquisas Origem-Destino do Metrô-SP
(1977, 1987, 1997 e 2007)

LAYOUT PESQUISA ORIGEM DESTINO 1977

Metrô - SP

Variável	Conteúdo	Início	Fim	Compr	Códigos
1	zona	Zona do Domicílio	1	3	3
2	subzona	Subzona do Domicílio	4	6	3
3	munires	Município do Domicílio	7	12	6
4	nota		13	13	1
5	conglome	Conglomerado	14	17	4
6	bolsao	Bolsão	18	18	1
7	setores	Setores	13	14	2
8	id_dom	Identifica Domicílio	15	34	20
9	f_dom	Identifica Primeiro Registro do Domicílio	35	35	1
					0 - Registros Complementares do Dom.
					1- Primeiro registro do domicílio
10	fe_dom	Fator de Expansão do Domicílio	36	43	8
11	domicil	Número do Domicílio	44	45	2
12	tipo_dom	Tipo de Domicílio	46	46	1
					1 - particular
					2 - Coletivo
13	pess_fam	Pessoa Residente da Família	47	48	2
14	não_res	Pessoas Não Residentes	49	50	2
15	condmora	Condição de Moradia	51	51	1
					1 - Própria paga
					2 - Própria em pagamento
					3 - Alugada
					4 - Cedida
					5 - Outro
16	aluguel	Prestação ou Aluguel	52	56	5
17	f_fam	Identifica Primeiro Registro da Família	57	57	1
					0 - Demais registros
					1- Primeiro registro da família
18	fe_fam	Fator de Expansão da Família	58	66	9
19	no_fam	Número da Família	67	68	2
20	tot_fam	Total de Famílias	69	70	2
21	qt_auto	Número de Veículos	71	71	1
22	codrendf	Código de renda familiar	72	72	1
					0 - Respondeu
					1 - Não sabe
					2 - Não respondeu
					3 - Não se aplica
23	redfam	Renda Familiar Mensal	73	77	5
24	f_pess	Identifica Primeiro Registro da Pessoa	78	78	1
					0 - Demais registros
					1- Primeiro registro da pessoa
25	fe_pess	Fator de Expansão da Pessoa	79	87	9
26	pessoa	Número da Pessoa	88	89	2
27	sit_fam	Situação Familiar	90	90	1
					1 - Chefe
					2 - Cônjugue
					3 - Filho(a)
					4 - Parente
					5 - Agregado
					6 - Empregado Residente
					7 - Visitante não Res.
28	idade	Idade	91	92	2
29	sexo	Sexo	93	93	1
					1 - Masculino
					2 - Feminino
30	grau_ins	Grau de Escolariedade	94	94	1
					1 - Sem instrução
					2 - Primário Incompleto
					3 - Primário Completo
					4 - Ginásial Incompleto
					5 - Ginásial Completo
					6 - Colegial Incompleto
					7 - Colegial Completo
					8 - Universitário Incompleto
					9 - Universitário Completo

LAYOUT PESQUISA ORIGEM DESTINO 1977

Metrô - SP

Variável	Conteúdo	Início	Fim	Compr	Códigos
31	ocupacao	95	96	2	1 - Estudante 2 - Prendas Domésticas 3 -Aposentado 4 - Sem Ocupação (nunca trabalhou) 5 - Desempregado 6 -Em Licença 11 - Serviços não Especializados 12 - Serviços semi-especializados 13 - Escriturários 14 - Empregados de Comércio/Vendedor/Corretor/Outros 15 - Chefe/Supervisor de Operários Especializados 16 - Empregado de Escritório c/ Nível Supervisor 17 - Empregados em Empresas c/ Nível Gerente 18 - Empregados c/ Nível Universitário 21 - Operários sem Especialização 22 - Operários semi-especializados 23 - Operários Especializados Trabalhando p/ Terceiros 24 - Operários Especializados Trabalhando em Fábricas 25 - Chefe/Supervisor de Operários Especializados 26 - Trabalhadores Rurais 27 - Trabalhadores Rurais c/ Nível de Capataz 31- Prestadores de Serviços Autônomos 32- Prestadores de Serviços por Cta Própria 33- Operários Especializados Trabalhando por Cta Própria 34- Prof.Lib.Trab. por Cta Própria c/ Trabalho Intelect/Mental 35- Prof.Lib.Trab. por Cta Própria c/ Nível Altam. Especializado 36- Prof.Lib.Trab. por Cta Própria c/ Nível Universitário 41- Func.Púb.Trab.p/o Governo em Serv.Não Especializados 42- Func.Púb.Trab.p/o Governo em Serv.semi-especializados 43 - Escriturários e Aux.de Escritório Trab.p/o Governo 44- Func.Púb.Trab.p/o Governo com Nível de Supervisor 45- Func.Púb.Trab.p/o Governo com Nível de Diretor 46 - Func.Púb.Trab.p/o Governo com Nível Universitário 51 - Sócios ou Donos de Pequeno Comércio 52 - Téc.,Prof., Artesões, Artífices, Sócios/Donos Firma Pequena 53 - Industriais e Comerciantes, Sócios ou Donos Firma Pequena 54 - Téc.,Prof., Artesões, Artífices, Sócios/Donos Firma Grande/Média 55 - Industriais e Comerciantes, Sócios ou Donos Firma Média 56 - Industriais e Comerciantes, Sócios ou Donos Firma Grande 57 - Agricultores Arrendários de Fazendas, Hortas, Granjas, etc. 58 - Agricultores Proprietários de Fazendas 59 - Grandes Fazendeiros
32	ramoativ	Ramo de Atividade	97	98	2
					1 -Agrícola 2 - Constr.Civil 3 - Industrial 4 - Comercial 5 - Func.Público 6 - Serv.Transportes 7 - Empr.Serviço 8 - Serv.Autônomos 9 - Outro 10 -Não se Aplica

LAYOUT PESQUISA ORIGEM DESTINO 1977

Metrô - SP

Variável	Conteúdo	Início	Fim	Compr	Códigos
33 co_ren_i	Código de Renda Individual	99	99	1	1 - Tem Renda 2 - Não Tem Renda 3 - Não Declarou
34 vl_ren_i	Renda Individual Mensal	100	104	5	
35 zonaesc1	Zona da Escola	105	107	3	1 a 243
36 subesc1	Subzona da Escola	108	110	3	1 a 633
37 muniesc1	Município da Escola 1	111	116	6	1 a 27
38 zonaesc2	Zona da 2 Escola	117	119	3	1 a 243
39 subesc2	Subzona da 2 Escola	120	122	3	1 a 633
40 zonatra1	Zona do Primeiro Trabalho	117	119	3	1 a 243
41 subtra1	Subzona do Primeiro Trabalho	120	122	3	1 a 633
42 munitra1	Município do Primeiro Trabalho	123	128	6	1 a 27
43 zonatra2	Zona do Segundo Trabalho	129	131	3	1 a 243
44 subtra2	Subzona do Segundo Trabalho	132	134	3	1 a 633
45 munitra2	Município do Segundo Trabalho	135	140	6	1 a 27
46 id_viage	Identificador do Registro de Viagem	141	141	1	0 - Não Tem Viagem 1 - Tem Viagem
47 fe_via	Fator de Expansão com Aferição	142	150	9	9 dígitos 2 casas decimais
48 zonaor	Zona de Origem	151	153	3	1 a 243
49 subor	Subzona de Origem	154	156	3	1 a 633
50 muniorig	Município de Origem	157	162	6	1 a 27
51 bolsaor		163	163	1	0 a 4
52 setoror		164	165	2	1 a 81
53 zonad	Zona de Destino	166	168	3	1 a 243
54 subdes	Subzona de Destino	169	171	3	1 a 633
55 munidest	Município de Destino	172	177	6	1 a 27
56 bolsaode		178	178	1	0 a 4
57 setorde		179	180	2	1 a 81
58 motivo_o	Motivo da Viagem na Origem	181	182	2	1 - Trabalho Indústria 2 - Trabalho Comércio 3 - Trabalho Serviços 4 - Escola/Educação 5 - Compras 6 - Negócios 7 - Médico/Dentista/Saúde 8 - Recreação/Visitas 9 - Servir Passageiro 10 - Residência
59 motivo_d	Motivo da Viagem no Destino	183	184	2	idem ao anterior
60 modo1	Primeiro Modo	185	186	2	1 - Ônibus Trólebus 2 - Ônibus Escolar/Empresa 3 - Dirigindo Automóvel 4 - Passageiro de Automóvel 5 - Táxi 6 - Lotação/Perua 7 - Metrô 8 - Trem 9 - Motocicleta 10 - Bicicleta 11 - A Pé 12 - Outros

LAYOUT PESQUISA ORIGEM DESTINO 1977

Metrô - SP

Variável	Conteúdo	Início	Fim	Compr	Códigos	
61	modo2	Segundo Modo	187	188	2	idem ao anterior
62	modo3	Terceiro Modo	189	190	2	idem ao anterior
63	modoprin	Modo principal	191	192	2	idem ao anterior
64	tipo_vg	Tipo de Viagem	193	193	1	1 - Coletivo 2 - individual 3 - A pé
65	hsaida	Hora da Saída	191	192	2	
66	minsaida	Minutos da Saída	193	194	2	
67	anda_o	Tempo Andando na Origem	195	196	2	
68	h_cheg	Hora da Chegada	197	198	2	
69	mincheg	Minutos da Chegada	199	200	2	
70	anda_d	Tempo Andando até o Destino	201	202	2	
71	duracao	Duração da Viagem (em minutos)	203	205	3	
72	tipoestc	Tipo de Estacionamento	206	206	1	1 - Zona Azul/Parquímetro 2 - Estacionamento Avulso 3 - Estacionamento Mensal 4 - Estacionamento Próprio 5 - Meio Fio/Logradouro 6 - Estacionamento Patrocinado 7 - Não Estacionou
73	custoest	Custo do Estacionamento	207	210	4	
74	pqnaocar	Código de Não Utilização do Carro	211	211	1	1 -Não Disponível 2 - Estacionamento Caro 3 - Difícil de Estacionar 4 - Condução mais Barata 5 -Condução mais Conveniente 6 - Outros
75	numviag	Número da Viagem	212	213	2	
76	ordem		214	221	8	1 a 230.606

LAYOUT PESQUISA ORIGEM DESTINO 1987

Metrô - SP

Variável	Conteúdo	Início	Fim	Compr	Códigos
1 ZONA	Zona do Domicílio	1	3	3	1 a 204
2 SZ	Subzona do Domicílio	4	4	1	1 a 9
3 SZSEQ	Subzona Sequencial do Domicílio	5	8	4	1 a 1.012
4 MUNI_DOM	Município do Domicílio	9	10	2	1 a 38
5 FAIXA	Faixa de Consumo de Energia Elétrica	11	11	1	
6 ID_DOM	Identifica Domicílio	12	29	18	
7 F_DOM	Identifica Primeiro Registro do Domicílio	30	30	1	0 - Registros Complementares do Dom. 1- Primeiro registro do domicílio
8 FE_DOM	Fator de Expansão do Domicílio	31	36	6	6 dígitos 2 casas decimais
9 DOMICIL	Número do Domicílio	37	40	4	
10 DATA	Data da Entrevista	41	46	6	
11 TIPO_DOM	Tipo de Domicílio	47	47	1	1 - Individual 2 - Coletivo
12 FAM_DOM	Número de Famílias no Domicílio	48	49	2	
13 F_FAM	Identifica Primeiro Registro da Família	50	50	1	0 - Demais registros 1- Primeiro registro da família
14 FE_FAM	Fator de Expansão da Família	51	56	6	6 dígitos 2 casas decimais
15 FAMILIA	Número da Família	57	58	2	
16 CD_ENTRE	Código da Entrevista	59	59	1	1 - Recusa Total 2 - Moradores Ausentes 3 - Domicílio Vago 4 - Incompleta 5 - Completa sem viagem 6 - Completa com viagem
17 TP_RBAIR	Tempo de Residência no Bairro	60	61	2	
18 CONDMORA	Condição de Moradia	62	62	1	1 - Não se Aplica 2 - Não respondeu 3 - Alugada 4 - Casa Própria
19 ALUG_SM	Valor do Aluguel em Salários Mínimos	63	65	3	
20 VALUGUEL	Valor do Aluguel Ajustado - set/87	66	73	8	
21 INS_CHEF	Grau de Instrução do Chefe da Família	74	74	1	1 - Analfabeto/4ª Série Incompleta 2 - 4ª Série Completa 3 - 1º Grau Completo 4 - Coelgial Completo 5 - Supreior Completo
22 QT_TV	Quantidade de Televisores	75	75	1	
23 QT_RADIO	Quantidade de Rádios	76	76	1	
24 QT_BANHO	Quantidade de Banheiros	77	77	1	
25 QT_AUTO	Quantidade de Automóveis	78	78	1	
26 QT_EMPRE	Quantidade de Empregados Domésticos	79	79	1	
27 QT_ASPIR	Quantidade de Aspiradores de Pó	80	80	1	
28 QT_MLAVA	Quantidade de Máquinas de Lavar	81	81	1	
29 ABAABIPE	Classificação Socioeconômica ABA-ABIPEME	82	82	1	1 - A 2 - B 3 - C 4 - D 5 - E
30 PONTOABA	Ponto-Aba - Total de Todos os Critérios ABA	83	84	2	
31 RENDA_FA	Renda Familiar Ajustado - set/87	85	92	8	
32 CD_RENDA	Código de Renda Familiar	93	93	1	1 - Não Tem Renda 2 - Renda Familiar Incompleta 3 - Renda Familiar Completa
33 RENDATRI	Renda Familiar Atribuída	94	103	10	10 dígitos 2 casas decimais
34 CD_ATRI	Código de Renda Familiar Atribuída	104	104	1	1 - Não Tem Renda

2 - Renda Familiar Atribuída pelo Crit.ABA-ABIPEME

3 - Renda Familiar Pesquisada

35	TOT_PESS	Total de Pessoas na Família	104	105	2	
36	TOT_V_FA	Total de Viagens na Família	106	107	2	
37	F_PESS	Identifica Primeiro Registro da Pessoa	108	108	1	0 - Demais registros 1- Primeiro registro da pessoa
38	FE_PESS	Fator de Expansão da Pessoa	109	114	6	6 dígitos 2 casas decimais
39	PESSOA	Número da Pessoa	115	116	2	
40	SIT_FAMI	Situação Familiar	117	117	1	1 - Chefe 2 - Cônjuge 3 - Filho(a) 4 - Parente 5 - Agregado 6 - Empregado Residente 7 - Visitante
41	IDADE	Idade	118	119	2	(anos)
42	FXETOD	Faixa Etária (anos)	120	121	2	1 - até 3 2 - 4 a 6 3 - 7 a 10 4 - 11 a 14 5 - 15 a 17 6 - 18 a 22 7 - 23 a 29 8 - 30 a 39 9 - 40 a 49 10 - 50 a 59 11 - 60 e mais
43	SEXO	Sexo	122	122	1	1 - Masculino 2 - Feminino
44	ESTUDA	Estuda Atualmente ?	123	123	1	1 - Sim 2 - Não
45	GRAU_INS	Grau de Instrução	124	124	1	0 - Não Declarou 1 - Não alfabetizado/4ª Série Incompleta 2 - 4ª Série Completa 3 - 1º Grau Completo 4 - Colegial Completo 5 - Superior Completo
46	CLAS_ATI	Classe de Ativ.da Empresa que Trabalha	125	126	2	0 - Não Declarou 1 - Agrícola 2 - Construção Civil 3 - Indústria 4 - Comércio 5 - Funcionalismo Público 6 - Serviços de Transporte 7 - Empresa de Serviços 8 - Serviços Autônomo 9 - Outros 10 - Não se Aplica
47	SET_ATIV	Setor de Atividade Agregado	127	127	1	1 - Indústria 2 - Comércio 3 - Serviços 4 - Outros
48	OCUPACAO	Ocupação Principal	128	129	2	
49	CD_REN_I	Código de Renda Individual	130	130	1	1 - Não Tem Renda 2 - Não Declarou 3 - Declarou
50	VL_REN_I	Renda Individual em Salários Mínimos	131	132	2	2 dígitos 1 casas decimais
51	FX_REN_I	Faixa de Renda Individual	133	134	2	0 - Não Declarou 1 - até 1 SM

2 - 1-2 SM

3 - 2-3 SM

4 - 3-4 SM

5 - 4-5 SM

6 - 5-6 SM

7 - 6-7 SM

8 - 7-8 SM

9 - 8-9 SM

10 - 9-10 SM

11 - 10-12 SM

12 - 12-15 SM

13 - 15-20 SM

14 - 20-25 SM

15 - 25-30 SM

16 - 30-40 SM

17 - 40-50 SM

18 - mais de 50 SM

52	VREN_IND	Valor da Renda Individual Ajustado-set/87	135	140	6	
53	ZONAESC	Zona da Escola	141	143	3	1 a 254
54	SZESCOLA	Subzona da Escola	144	144	1	1 a 9
55	SZESEQ	Subzona Sequencial da Escola	145	148	4	1 a 1.012
56	MUNIESC	Município da Escola	149	150	2	1 a 38
57	ZONATRA1	Zona do Primeiro Trabalho	151	153	3	1 a 254
58	SZTRAB1	Subzona do Primeiro Trabalho	154	154	1	1 a 9
59	SZT1SEQ	Subzona Sequencial do Primeiro Trabalho	155	158	4	1 a 1.012
60	MUNITRA1	Município do Primeiro Trabalho	159	160	2	1 a 38
61	ZONATRA2	Zona do Segundo Trabalho	161	163	3	1 a 254
62	SZTRAB2	Subzona do Segundo Trabalho	164	164	1	1 a 9
63	SZT2SEQ	Subzona Sequencial do Segundo Trabalho	165	168	4	1 a 1.012
64	MUNITRA2	Município do Segundo Trabalho	169	170	2	1 a 38
65	VIG_PESS	Número de Viagem da Pessoa	171	172	2	
66	F_VIA	Identifica Primeiro Registro da Viagem	173	173	1	0 - Demais registros 1- Primeiro registro da pessoa
67	FE_VIA	Fator de Expansão da Viagem	174	179	6	6 dígitos 2 casas decimais
68	DIA_SEM	Dia da Semana	180	180	1	2 - Segunda-feira 3 - Terça-feira 4 - Quarta-feira 5 - Quinta-feira 6 - Sexta-feira
69	ZONA_O	Zona de Origem	181	183	3	1 a 254
70	SZO	Subzona de Origem	184	184	1	1 a 9
71	SZOSEQ	Subzona Sequencial de Origem	185	188	4	1 a 1.012
72	MUNIORIG	Município de Origem	189	190	2	1 a 38
73	ZONA_D	Zona de Destino	191	193	3	1 a 254
74	SZD	Subzona de Destino	194	194	1	1 a 9
75	SZDSEQ	Subzona Sequencial de Destino	195	198	4	1 a 1.012
76	MUNIDEST	Município de Destino	199	200	2	1 a 38
77	MOTIVO_O	Motivo na Origem	201	201	1	1 - Trabalho Indústria 2 - Trabalho Comércio 3 - Trabalho Serviços 4 - Escola/Educação 5 - Compras 6 - Negócios 7 - Médico/Dentista/Saúde 8 - Recreação/Visitas 9 - Residência
78	MOTIVO_D	Motivo no Destino	202	202	1	idem ao anterior
79	MOT_SRES	Motivo de Destino sem Residência	203	203	1	1 - Trabalho Indústria 2 - Trabalho Comércio

							3 - Trabalho/Serviços
							4 - Escola/Educação
							5 - Compras
							6 - Negócios
							7 - Médico/Dentista/Saúde
							8 - Recreação/Visitas
80	MODO1	Modo 1	204	205	2	1	1 - Ônibus Diesel
						2	2 - Trólebus
						3	3 - Ônibus Fretado
						4	4 - Escolar
						5	5 - Dirigindo Automóvel
						6	6 - Passageiro de Automóvel
						7	7 - Táxi
						8	8 - Lotação/Perua
						9	9 - Metrô
						10	10 - Trem
						11	11 - Moto
						12	12 - Bicicleta
						13	13 - A Pé
						14	14 - Caminhão
						15	15 - Outros
81	MODO2	Modo 2	206	207	2		idem ao anterior
82	MODO3	Modo 3	208	209	2		idem ao anterior
83	H_SAIDA	Hora da Saída	210	211	2		
84	MINSAIDA	Minutos da Saída	212	213	2		
85	ANDA_O	Tempo Andando desde a Origem	214	215	2		
86	H_CHEG	Hora da Chegada	216	217	2		
87	MINCHEG	Minutos da Chegada	218	219	2		
88	ANDA_D	Tempo Andando até o Destino	220	221	2		
89	ESTAC	Tipo de Estacionamento	222	222	1	1	1 - Zona Azul/Parquímetro
						2	2 - Estacionamento Particular
						3	3 - Estacionamento Próprio
						4	4 - Estacionamento Patrocinado
						5	5 - Meio Fio
						6	6 - Não Estacionou
90	DURACAO	Duração da Viagem (em minutos)	223	225	3		
91	MODOPRIN	Modo Principal	226	227	2	1	1 - Ônibus Diesel
						2	2 - Trólebus
						3	3 - Ônibus Fretado
						4	4 - Escolar
						5	5 - Dirigindo Automóvel
						6	6 - Passageiro de Automóvel
						7	7 - Táxi
						8	8 - Lotação/Perua
						9	9 - Metrô
						10	10 - Trem
						11	11 - Moto
						12	12 - Bicicleta
						13	13 - A Pé
						14	14 - Caminhão
						15	15 - Outros
92	TIPO_VG	Tipo de Viagem	228	228	1	1	1 - Coletivo
						2	2 - Individual
						3	3 - A pé
93	ID_ORDEM	Número de Ordem do Registro	229	235	7	1 a 223.926	

LAYOUT PESQUISA ORIGEM DESTINO 1997 - ZONA

Metrô - SP

Variável	Conteúdo	Início	Fim	Compr	Códigos
1 ZONA	Zona do Domicílio	1	3	3	1 a 389
2 SZ	Subzona do Domicílio	4	4	1	1 a 9
3 SZSEQ	Subzona Sequencial do Domicílio	5	8	4	1 a 1.400
4 MUNI_DOM	Município do Domicílio	9	10	2	1 a 39
5 DISTRDOM	Distrito do Domicílio (São Paulo)	11	12	2	1 a 96
6 DTRMUN	Distrito+Município do Domicílio	13	15	3	1 a 134
7 ID_DOM	Identifica Domicílio	16	23	8	
8 F_DOM	Identifica Primeiro Registro do Domicílio	24	24	1	0 - Demais registros 1- Primeiro registro do domicílio
9 FE_DOM	Fator de Expansão do Domicílio	25	34	10	10 dígitos 5 casas decimais
10 DOMICIL	Número do Domicílio	35	38	4	
11 TIPO_DOM	Tipo de Domicílio	39	39	1	1 - Particular 2 - Coletivo 3 - Favela
12 TOT_FAM	Total de Famílias no Domicílio	40	41	2	
13 NO_MORAD	Número de Moradores do Domicílio	42	43	2	
14 RESULDOM	Resultado do Domicílio	44	44	1	6 - Completa sem Viagem 7 - Completa com Viagem
15 ID_FAM	Identifica Família	45	54	10	
16 F_FAM	Identifica Primeiro Registro da Família	55	55	1	0 - Demais registros 1- Primeiro registro da família
17 FE_FAM	Fator de Expansão da Família	56	65	10	10 dígitos 5 casas decimais
18 FAMILIA	Número da Família	66	67	2	
19 RESULFAM	Resultado da Família	68	68	1	6 - Completa sem Viagem 7 - Completa com Viagem
20 NO_MORAF	Número de Moradores da Família	69	70	2	
21 TP_RMUN	Tempo de Residência no Município	71	72	2	
22 TP_RBAIR	Tempo de Residência no Bairro	73	74	2	
23 CONDMORA	Condição de Moradia	75	75	1	1 - Alugada 2 - Própria 3 - Cedida 4 - Outros 5 - Não Respondeu
24 QT_RADIO	Rádios	76	76	1	
25 QT_GEL	Geladeiras	77	77	1	
26 QT_TV	Tv. a Cores	78	78	1	
27 QT_VIDEO	Vídeo Cassetes	79	79	1	
28 QT_BANHO	Banheiros	80	80	1	
29 QT_AUTO	Automóveis	81	81	1	
30 QT_ASPIR	Aspiradores de Pó	82	82	1	
31 QT_MLAVA	Máquinas de Lavar	83	83	1	
32 QT_EMPRE	Empregados Domésticos	84	84	1	
33 QT_MICRO	Microcomputadores	85	85	1	
34 QT_TEL	Telefones	86	86	1	
35 QT_CEL	Celulares	87	87	1	
36 DIA_SEMA	Dia da Semana	88	88	1	2 - Segunda-feira 3 - Terça-feira 4 - Quarta-feira 5 - Quinta-feira 6 - Sexta-feira
37 RENDA_FA	Renda Familiar	89	96	8	8 dígitos 2 casas decimais
38 CD_RENFA	Código de Renda Familiar	97	97	1	1 - Renda Familiar Completa 2 - Não Tem Renda 3 - Renda Familiar Incompleta
39 RENDATRI	Renda Familiar Atribuída	98	105	8	8 dígitos 2 casas decimais
40 ABIPEME	Classificação ABIPEME	106	106	1	1 - A 2 - B

						3 - C
						4 - D
						5 - E
41	ID_PESS	Identifica Pessoa	107	118	12	
42	F_PESS	Identifica Primeiro Registro da Pessoa	119	119	1	0 - Demais registros 1- Primeiro registro da pessoa
43	FE_PESS	Fator de Expansão da Pessoa	120	129	10	10 dígitos 5 casas decimais
44	PESSOA	Número da Pessoa	130	131	2	
45	SIT_FAM	Situação Familiar	132	132	1	1 - Chefe 2 - Cônjugue 3 - Filho(a) 4 - Parente/Agregado 5 - Empregado Residente 6 - Visitante não Residente na RMSP
46	IDADE	Idade	133	134	2	(anos)
47	SEXO	Sexo	135	135	1	1 - Masculino 2 - Feminino
48	SE_ESTUD	Estuda Atualmente ?	136	136	1	1 - Não 2 - Creche/Pré-Escola 3 - 1o./2o./3o. Graus 4 - Outros
49	GRAU_INS	Grau de Instrução	137	137	1	1 - Não-alfabetizado 2 - Pré-Escola 3 - 1o. Grau Incompleto 4 - 1o. Grau Completo 5 - 2o. Grau Incompleto 6 - 2o. Grau Completo 7 - Superior Incompleto 8 - Superior Completo
50	CD_ATIVI	Condição de Atividade	138	138	1	1 - Ocupado 2 - Ocupado Eventualmente 3 - Em Licença 4 - Não Ocupado 5 - Aposentado/Pensionista 6 - Nunca Trabalhou 7 - Dona de Casa 8 - Estudante
51	OCUP_PRI	Ocupação Principal	139	140	2	01 - Assalariado com Carteira 02 - Assalariado sem Carteira 03 - Funcionário Público 04 - Autônomo 05 - Empregador 06 - Profissional Liberal 07 - Trab. Doméstico com Carteira 08 - Trab. Doméstico sem Carteira 09 - Dono de Negócio Familiar 10 - Trabalhador Familiar 11 - Não se Aplica
52	SET_ATIV	Setor de Atividade	141	142	2	01 - Agrícola 02 - Construção Civil 03 - Indústria 04 - Comércio 05 - Serviços de Transp. Carga 06 - Serviços de Transp. de Passag. 07 - Serviços Creditícios/Financeiros 08 - Serviços Pessoais 09 - Serviços de Alimentação 10 - Serviços de Saúde 11 - Serviços de Educação 12 - Serviços Especializados

						13 - Serviços da Adm. Pública
						14 - Outros
						15 - Não se Aplica
53	CO_REN_I	Condição de Renda Individual	143	143	1	1 - Tem Renda
						2 - Não Tem Renda
						3 - Não Respondeu
54	VL_REN_I	Renda Individual	144	151	8	8 dígitos 2 casas decimais
55	USA_VTRA	Usa Vale-Transporte ?	152	152	1	1 - Sim
						2 - Não
56	ZONAESC	Zona da Escola	153	155	3	1 a 389
57	SZESCOLA	Subzona da Escola	156	156	1	1 a 9
58	SZESEQ	Subzona Sequencial da Escola	157	160	4	1 a 1.400
59	DISTRESC	Distrito da Escola (São Paulo)	161	162	2	1 a 96
60	MUNIESC	Município da Escola	163	164	2	1 a 39
61	DTRMUESC	Distrito+Município da Escola	165	167	3	1 a 134
62	ZONATRA1	Zona do Primeiro Trabalho	168	170	3	1 a 389
63	SZTRAB1	Subzona do Primeiro Trabalho	171	171	1	1 a 9
64	SZT1SEQ	Subzona Sequencial do Primeiro Trabalho	172	175	4	1 a 1.400
65	DISTRAB1	Distrito do Primeiro Trabalho (São Paulo)	176	177	2	1 a 96
66	MUNITRA1	Município do Primeiro Trabalho	178	179	2	1 a 39
67	DTRMUTR1	Distrito+Município do Primeiro Trabalho	180	182	3	1 a 134
68	TRAB1_SN	Primeiro Trabalho é igual a Residência ?	183	183	1	1 - Sim
						2 - Não
69	ZONATRA2	Zona do Segundo Trabalho	184	186	3	1 a 389
70	SZTRAB2	Subzona do Segundo Trabalho	187	187	1	1 a 9
71	SZT2SEQ	Subzona Sequencial do Segundo Trabalho	188	191	4	1 a 1.400
72	DISTRAB2	Distrito do Segundo Trabalho (São Paulo)	192	193	2	1 a 96
73	MUNTRA2	Município do Segundo Trabalho	194	195	2	1 a 39
74	DTRMUTR2	Distrito+Município do Segundo Trabalho	196	198	3	1 a 134
75	TRAB2_SN	Segundo Trabalho é igual a Residência ?	199	199	1	1 - Sim
						2 - Não
76	ID_VIAGE	Identifica Viagem	200	205	6	
77	FE_VIA	Fator de Expansão da Viagem	206	215	10	10 dígitos 5 casas decimais
78	ZONA_O	Zona de Origem	216	218	3	1 a 389
79	SZO	Subzona de Origem	219	219	1	1 a 9
80	SZOSEQ	Subzona Sequencial de Origem	220	223	4	1 a 1.400
81	DISTRORG	Distrito de Origem	224	225	2	1 a 96
82	MUNIORIG	Município de Origem	226	227	2	1 a 39
83	DTRMUNO	Distrito+Município de Origem	228	230	3	1 a 134
84	ID_POLOO	Pólo Gerador na Origem	231	231	1	
85	ZONA_D	Zona de Destino	232	234	3	1 a 389
86	SZD	Subzona de Destino	235	235	1	1 a 9
87	SZDSEQ	Subzona Sequencial de Destino	236	239	4	1 a 1.400
88	DISTRD	Distrito de Destino	240	241	2	1 a 96
89	MUNIDEST	Município de Destino	242	243	2	1 a 39
90	DTRMUND	Distrito+Município de Destino	244	246	3	1 a 134
91	ID_POLOD	Pólo Gerador no Destino	247	247	1	
92	MOTIVO_O	Motivo na Origem	248	248	1	1 - Trabalho Indústria 2 - Trabalho Comércio 3 - Trabalho Serviços 4 - Escola/Educação 5 - Compras 6 - Médico/Dentista/Saúde 7 - Recreação/Visitas 8 - Residência 9 - Outros
93	MOTIVO_D	Motivo no Destino	249	249	1	idem ao anterior
94	SERVIR_O	Servir Passageiro na Origem	250	250	1	1 - Sim 2 - Não
95	SERVIR_D	Servir Passageiro no Destino	251	251	1	1 - Sim

						2 - Não
96	MODO1	Modo 1	252	253	2	01 - Ônibus
						02 - Ônibus Fretado
						03 - Transporte Escolar
						04 - Dirigindo Automóvel
						05 - Passageiro de Automóvel
						06 - Táxi
						07 - Lotação/Perua
						08 - Metrô
						09 - Trem
						10 - Moto
						11 - Bicicleta
						12 - A Pé
						13 - Outros
97	MODO2	Modo 2	254	255	2	idem ao anterior
98	MODO3	Modo 3	256	257	2	idem ao anterior
99	MODO4	Modo 4	258	259	2	idem ao anterior
100	PQ_VIAPE	Motivo da Viagem a Pé	260	260	1	1 - Condução Cara 2 - Condução Desconfortável 3 - Ponto/Estação Distante 4 - Condução Demora a Passar 5 - Condução Lotada 6 - Viagem Demorada 7 - Horário Irregular (da condução) 8 - Pequena Distância 9 - Outros Motivos
101	H_SAIDA	Hora da Saída	261	262	2	
102	MINSAIDA	Minutos da Saída	263	264	2	
103	ANDA_O	Minutos Andando desde a Origem	265	266	2	
104	H_CHEG	Hora da Chegada	267	268	2	
105	MINCHEG	Minutos da Chegada	269	270	2	
106	ANDA_D	Minutos Andando até o Destino	271	272	2	
107	DURACAO	Duração da Viagem (em minutos)	273	275	3	
108	MODOPRIN	Modo Principal	276	277	2	01 - Ônibus 02 - Ônibus Fretado 03 - Transporte Escolar 04 - Dirigindo Automóvel 05 - Passageiro de Automóvel 06 - Táxi 07 - Lotação/Perua 08 - Metrô 09 - Trem 10 - Moto 11 - Bicicleta 12 - A Pé 13 - Outros
109	TIPO_VG	Tipo de Viagem	278	278	1	1 - Coletivo 2 - Individual 3 - A pé
110	ID_ORDEM	Número de Ordem do Registro	279	284	6	

LAYOUT PESQUISA ORIGEM DESTINO 2007

Variável	Conteúdo	Início	Fim	Compr	Códigos
1 ZONA	Zona do Domicílio	1	3	3	1 a 460
2 MUNI_DOM	Município de Domicílio	4	5	2	1 a 39
3 CO_DOM_X	Coordenada X Domicílio	6	17	12	12 dígitos 2 casas decimais
4 CO_DOM_Y	Coordenada Y Domicílio	18	29	12	12 dígitos 2 casas decimais
5 ID_DOM	Identifica Domicílio	30	36	7	
6 F_DOM	Identifica Primeiro Registro do Domicílio	37	37	1	0 - Demais Registros 1- Primeiro Registro do Domicílio
7 FE_DOM	Fator de Expansão do Domicílio	38	47	10	10 dígitos 5 casas decimais
8 DOM	Número do Domicílio	48	51	4	
9 CD_ENTRE	Código de Entrevista	52	52	1	5 - Completa sem Viagem 6 - Completa com Viagem
10 DATA	Data da Entrevista	53	60	8	
11 TIPO_DOM	Tipo de Domicílio	61	61	1	1 – Particular 2 – Coletivo 3 – Favela
12 NO_MORAD	Total de Moradores no Domicílio	62	63	2	
13 TOT_FAM	Total de Famílias no Domicílio	64	65	2	
14 ID_FAM	Identifica Família	66	74	9	
15 F_FAM	Identifica Primeiro Registro da Família	75	75	1	0 - Demais Registros 1- Primeiro Registro da Família
16 FE_FAM	Fator de Expansão da Família	76	85	10	10 dígitos 5 casas decimais
17 FAMILIA	Número da Família	86	87	2	
18 NO_MORAF	Total de Moradores na Família	88	89	2	
19 CONDMORA	Condição de Moradia	90	90	1	1 - Alugada 2 - Própria 3 - Cedida 4 - Outros 5 - Não Respondeu
20 QT_RADIO	Rádios	91	91	1	
21 QT_GEL1	Geladeiras de 1 porta	92	92	1	
22 QT_GEL2	Geladeiras de 2 portas	93	93	1	
23 QT_TVCOR	Tv. a Cores	94	94	1	
24 QT_FREEZ	Freezer	95	95	1	
25 QT_VIDEO	Vídeo Cassetes/DVD	96	96	1	
26 QT_BANHO	Banheiros	97	97	1	
27 QT_MOTO	Motos	98	98	1	
28 QT_AUTO	Automóveis	99	99	1	
29 QT_ASPIR	Aspiradores de Pó	100	100	1	
30 QT_MLAVA	Máquinas de Lavar	101	101	1	
31 QT_EMPRE	Empregados Domésticos	102	102	1	
32 QT_MICRO	Microcomputadores	103	103	1	
33 QT_BICICLE	Bicicletas	104	104	1	
34 NAO_DCL_IT	Código de Declaração de Itens de Conforto	105	105	1	
35 CRITERIO_B	Crítario de Classificação Econômica Brasil	106	106	1	1 – A1 2 – A2 3 – B1 4 – B2 5 – C1 6 – C2 7 – D 8 – E
36 ANO_AUTO1	Ano Fabricação - Auto 1	107	110	4	
37 ANO_AUTO2	Ano Fabricação - Auto 2	111	114	4	
38 ANO_AUTO3	Ano Fabricação - Auto 3	115	118	4	
39 RENDA_FA	Renda Familiar	119	126	8	8 dígitos 2 casa decimais
40 CD_RENFA	Código de Renda Familiar	127	127	1	1 - Renda Familiar Declarada e Maior que Zero 2 - Renda Familiar Declarada como Zero 3 - Renda Atribuída pelo Critério Brasil 4 - Renda Atribuída pela Média da Zona
41 ID_PESS	Identifica Pessoa	128	139	12	
42 F_PESS	Identifica Primeiro Registro da Pessoa	140	140	1	0 - Demais registros 1- Primeiro registro da pessoa
43 FE_PESS	Fator de Expansão da Pessoa	141	150	10	10 dígitos 5 casas decimais
44 PESSOA	Número da Pessoa	151	152	2	
45 SIT_FAM	Situação Familiar	153	153	1	1 – Pessoa Responsável 2 – Cônjuge/Companheiro(a)

						3 – Filho(a)/Enteado(a)
						4 - Outro Parente
						5 - Agregado
						6 - Empregado Residente
						7 - Parente do Empregado
46	IDADE	Idade	154	155	2	(anos)
47	SEXO	Gênero	156	156	1	1 - Masculino 2 - Feminino
48	ESTUDA	Estuda Atualmente?	157	157	1	1 - Não 2 - Creche/Pré-Escola 3 - 1º Grau /Fundamental 4 - 2º Grau/Médio 6 – Superior/Universitário 7 - Outros
49	GRAU_INS	Grau de Instrução	158	158	1	1 - Não-Alfabetizado/Primário Incompleto 2 - Primário Completo/Ginásio Incompleto 3 - Ginásio Completo/Colegial Incompleto 4 - Colegial Completo/Superior Incompleto 5 - Superior Completo
50	CD_ATIVI	Condição de Atividade	159	159	1	1 - Tem trabalho 2 - Faz bico 3 - Em Licença Médica 4 - Aposentado/Pensionista 5 - Sem Trabalho 6 - Nunca Trabalhou 7 - Dona de Casa 8 - Estudante
51	CO_REN_I	Condição de Renda Individual	160	160	1	1 - Tem Renda 2 - Não Tem Renda 3 - Não Respondeu
52	VL_REN_I	Renda Individual	161	168	8	8 dígitos
53	ZONA_ESC	Zona da Escola	169	171	3	1 a 460
54	MUNIESC	Município da Escola	172	173	2	
55	CO_ESC_X	Coordenada X Escola	174	185	12	12 dígitos 2 casas decimais
56	CO_ESC_Y	Coordenada Y Escola	186	197	12	12 dígitos 2 casas decimais
57	TIPO_ESC	Tipo de Escola	198	198	1	1 - Pública 2 - Particular
58	ZONATRA1	Zona do Primeiro Trabalho	199	201	3	1 a 460
59	MUNITRA1	Município do Primeiro Trabalho	202	203	2	1 a 39
60	CO_TR1_X	Coordenada X 1º Trabalho	204	215	12	12 dígitos 2 casas decimais
61	CO_TR1_Y	Coordenada Y 1º Trabalho	216	227	12	12 dígitos 2 casas decimais
62	TRAB1_RE	Primeiro Trabalho é igual a Residência ?	228	228	1	1 - Sim 2 - Não 3 - Sem endereço fixo
63	TRABEXT1	Realiza Trabalho Externo-1º Trabalho	229	229	1	1 - Sim 2 - Não
64	OCUP1	Ocupação do 1º Trabalho	230	232	3	
65	SETOR1	Setor de Atividade do 1º Trabalho	233	234	2	
66	VINC1	Vínculo Empregatício do 1º Trabalho	235	235	1	1 - Assalariado com carteira 2 - Assalariado sem carteira 3 - Funcionário Público 4 - Autônomo 5 - Empregador 6 - Profissional Liberal 7 – Dono de Negócio Familiar 8 - Trabalho Familiar
67	ZONATRA2	Zona do Segundo Trabalho	236	238	3	1 a 460
68	MUNITRA2	Município do Segundo Trabalho	239	240	2	1 a 39
69	CO_TR2_X	Coordenada X 2º Trabalho	241	252	12	12 dígitos 2 casas decimais
70	CO_TR2_Y	Coordenada Y 2º Trabalho	253	264	12	12 dígitos 2 casas decimais
71	TRAB2_RE	Segundo Trabalho é igual a Residência ?	265	265	1	1 - Sim 2 - Não 3 - Sem endereço fixo
72	TRABEXT2	Realiza Trabalho Externo 2º Trabalho	266	266	1	1 - Sim 2 - Não
73	OCUP2	Ocupação do 2º Trabalho	267	269	3	
74	SETOR2	Setor de Atividade do 2º Trabalho	270	271	2	
75	VINC2	Vínculo Empregatício do 2º Trabalho	272	272	1	idem ao 1º Trabalho
76	N_VIAG	Número da Viagem	273	274	2	

77	FE_VIA	Fator de Expansão da Viagem	275	284	10	10 dígitos 5 casas decimais
78	DIA_SEM	Dia da Semana	285	285	1	2 - Segunda-Feira 3 - Terça-Feira 4 - Quarta-Feira 5 - Quinta-Feira 6 - Sexta-Feira
79	TOT_VIAG	Total de Viagens da Pessoa	286	287	2	
80	ZONA_O	Zona de Origem	288	290	3	1 a 460
81	MUNI_O	Município de Origem	291	292	2	1 a 39
82	CO_O_X	Coordenada X Origem	293	304	12	12 dígitos 2 casas decimais
83	CO_O_Y	Coordenada Y Origem	305	316	12	12 dígitos 2 casas decimais
84	ZONA_D	Zona de Destino	317	319	3	1 a 460
85	MUNI_D	Município de Destino	320	321	2	1 a 39
86	CO_D_X	Coordenada X Destino	322	333	12	12 dígitos 2 casas decimais
87	CO_D_Y	Coordenada Y Destino	334	345	12	12 dígitos 2 casas decimais
88	ZONA_T1	Zona da 1ª Transferência	346	348	3	1 a 460
89	MUNI_T1	Município 1ª Transferência	349	350	2	1 a 39
90	CO_T1_X	Coordenada X 1ª Transferência	351	362	12	12 dígitos 2 casas decimais
91	CO_T1_Y	Coordenada Y 1ª Transferência	363	374	12	12 dígitos 2 casas decimais
92	ZONA_T2	Zona da 2ª Transferência	375	377	3	1 a 460
93	MUNI_T2	Município 2ª Transferência	378	379	2	1 a 39
94	CO_T2_X	Coordenada X 2ª Transferência	380	391	12	12 dígitos 2 casas decimais
95	CO_T2_Y	Coordenada Y 2ª Transferência	392	403	12	12 dígitos 2 casas decimais
96	ZONA_T3	Zona da 3ª Transferência	404	406	3	1 a 460
97	MUNI_T3	Município 3ª Transferência	407	408	2	1 a 39
98	CO_T3_X	Coordenada X 3ª Transferência	409	420	12	12 dígitos 2 casas decimais
99	CO_T3_Y	Coordenada Y 3ª Transferência	421	432	12	12 dígitos 2 casas decimais
100	MOTIVO_O	Motivo na Origem	433	434	2	1 - Trabalho/Indústria 2 - Trabalho/Comércio 3 - Trabalho/Serviços 4 - Educação 5 - Compras 6 - Saúde 7 - Lazer 8 - Residência 9 - Procurar Emprego 10 – Assuntos Pessoais
101	MOTIVO_D	Motivo no Destino	435	436	2	idem ao anterior
102	SERVIR_O	Servir Passageiro na Origem	437	437	1	1 - Sim 2 - Não
103	SERVIR_D	Servir Passageiro no Destino	438	438	1	1 - Sim 2 - Não
104	MODO1	Modo 1	439	440	2	01 – Ônibus Município S.Paulo 02 – Ônibus Outros Municípios 03 – Ônibus Metropolitano 04 – Ônibus Fretado 05 - Escolar 06 - Dirigindo Automóvel 07 - Passageiro de Automóvel 08 - Táxi 09 – Microônibus/Van Município de S.Paulo 10 – Microônibus/Van Outros Município 11 – Microônibus/Van Metropolitano 12 - Metrô 13 - Trem 14 - Moto 15 - Bicicleta 16 - A Pé 17 - Outros
105	MODO2	Modo 2	441	442	2	idem ao anterior
106	MODO3	Modo 3	443	444	2	idem ao anterior
107	MODO4	Modo 4	445	446	2	idem ao anterior
108	H_SAIDA	Hora Saída	447	448	2	Hora de Saída
109	MIN_SAIDA	Minuto Saída	449	450	2	Minuto de Saída
110	ANDA_O	Tempo Andando na Origem	451	452	2	Em minutos
111	H_CHEG	Hora Chegada	453	454	2	Hora de Chegada
112	MIN_CHEG	Minuto Chegada	455	456	2	Minuto de Chegada
113	ANDA_D	Tempo Andando no Destino	457	458	2	Em minutos
114	DURACAO	Duração da Viagem (em minutos)	459	461	3	

115 MODOPRIN	Modo Principal	462	463	2	01 – Ônibus Município S.Paulo 02 – Ônibus Outros Municípios 03 – Ônibus Metropolitano 04 - Ônibus Fretado 05 - Escolar 06 - Dirigindo Automóvel 07 - Passageiro de Automóvel 08 - Táxi 09 – Microônibus/Van Município de S.Paulo 10 – Microônibus/Van Outros Município 11 – Microônibus/Van Metropolitano 12 - Metrô 13 - Trem 14 - Moto 15 - Bicicleta 16 - A Pé 17 - Outros
116 TIPOVG	Tipo de Viagem	464	464	1	1 - Coletivo 2 - Individual 3 - A pé
117 PAG_VIAG	Quem Pagou a Viagem	465	465	1	1 – Você/Sua Família 2 – Empregador 3 - Isento 4 - Outros
118 TP_ESAUTO	Tipo de Estacionamento Automóvel	466	466	1	1 – Não Estacionou 2 – Zona azul/Zona marrom 3 – Patrocinado 4 – Proprio 5 – Meio-Fio 6 – Avulso 7 – Mensal 8 – Não respondeu
119 VL_EST	Valor do Estacionamento Automóvel	467	470	4	
120 PE_BICI	Por Que Viajou A Pé ou Bicicleta	471	471	1	1 – Pequena Distância 2 – Condução Cara 3 – Ponto/Estação Distante 4 – Condução Demora a Passar 5 – Viagem Demorada 6 – Condução Lotada 7 – Atividade Física 8 – Outros Motivos
121 TP_ESBICI	Estacionamento Bicicleta	472	472	1	1 – Bicicletário Gratuito 2 – Bicicletário Pago 3 – Local Privado 4 – Rua/Local Público 5 – Outros
122 ID_ORDEM	Número de Ordem do Registro	473	478	6	1 a 196.698

ANEXO D – Rotinas de Preparação das Bases de Dados das Pesquisas Origem-Destino do Metrô-SP (1977, 1987, 1997 e 2007)

rotina_final_1977

January 4, 2016

1 Setup Inicial

```
In [ ]: # coding: utf-8
        import math
        import logging
        import time
        import pandas as pd

        # Faz os gráficos de output ficarem um pouco melhores
        pd.set_option('display.mpl_style', 'default')
```

2 Definindo Loggers

Define loggers

Estes ‘loggers’ serão utilizados para salvar as saídas (outputs) em um arquivo de texto no diretório ‘outputs’.

ref: <http://stackoverflow.com/questions/17035077/python-logging-to-multiple-log-files-from-different-classes>

ATENÇÃO: RODAR O BLOCO ABAIXO APENAS UMA VEZ, SE ESTE BLOCO FOR EXECUTADO MAIS DE UMA VEZ OS LOGs SERÃO DUPLICADOS.

```
In [ ]: log_formatter = logging.Formatter('%(asctime)s | %(levelname)s: %(message)s')

log_output = logging.getLogger('log_output')
FH_output = logging.FileHandler(
            'outputs/1977_output.log', mode='w')
FH_output.setFormatter(log_formatter)
log_output.setLevel(logging.INFO)
log_output.addHandler(FH_output)
log_output.propagate = False

# Este logger (log_tela) loga na tela e também
# no arquivo de output, junto com o conteúdo do
# logger log_output
log_tela = logging.getLogger('log_tela')
SH_tela = logging.StreamHandler()
SH_tela.setFormatter(log_formatter)
log_tela.addHandler(SH_tela)
log_tela.addHandler(FH_output)
log_tela.setLevel(logging.INFO)
log_tela.propagate = False
```

3 Funções de 1977

3.1 Funções gerais assessórias

Função	Status
verifica_dummy	OK
verifica_range	OK
pre_processamento	OK
coordenadas	OK

```
In [ ]: log_tela.info('Definindo as funções gerais assessórias')
log_output.info('\n'=====\\n')

def verifica_dummy(df, variavel):
    """
    Verifica se uma variável, dummy, contém algum valor diferente de 0 ou de 1.
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados.
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_dummy(dataframe, 'coluna a ser verificada')
    """
    contador_de_erros = 0
    log_tela.info('Verificando a variável Dummy: ' + variavel)

    df_erros = df[(df[variavel] != 0) & (df[variavel] != 1)]
    if len(df_erros[variavel].value_counts()) > 0:
        log_tela.warning(variavel + ": " +
                          str(len(df_erros[variavel].value_counts())) +
                          " erros encontrados:\\n" +
                          str(df_erros[variavel].value_counts()))
        # Imprimindo as linhas com problemas no log_output
        #log_output.warning("Linhas com erro:\\n -----")
        #for index, linha in df_erros.iterrows():
        #    log_output.warning("Linha " + str(index+1) + ':\\n => ' +
        #                      str(linha))
    else:
        log_tela.info(variavel + ": Nenhum erro encontrado.")

def verifica_range(df, variavel, valor_menor, valor_maior):
    """
    Verifica se uma variável, do tipo número inteiro, contém algum valor menor
    que "valor_menor" ou maior que "valor_maior".
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados
    :param valor_menor: Valor mínimo esperado na variável (int ou float)
    :param valor_maior: Valor máximo esperado na variável (int ou float)
```

```

:rtype: Sem retorno, apenas salva as avaliações nos logs.

Uso:
    verifica_range(dataframe, 'nome_variavel', valor_menor, valor_maior)
"""

log_tela.info('Verificando Range da variável: ' + variavel)
# Obs: Registros inválidos: None (equivalente a NA)
log_output.info('\n\n' +
    ',      - ' + 'Mínimo esperado: ' + str(valor_menor) + '\n' +
    ',      - ' + 'Máximo esperado: ' + str(valor_maior) + '\n' +
    ',      - ' + 'Total de registros: ' + str(len(df[variavel])) +
    '\n' +
    ',      - ' + 'Registros nulos (NA): ' +
    str(df[variavel].isnull().sum()) + '\n'
    #'      - ' + 'Descrição da variável: \n' +
    #str(df[variavel].describe())
)

df_errores = df[(df[variavel] < valor_menor) | (df[variavel] > valor_maior)]

if len(df_errores[variavel].value_counts()) > 0:

    result = df_errores[variavel].value_counts().sort_index()
    #Verificando limite inferior
    if result.first_valid_index() < valor_menor:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro mínimo encontrado: ' +
            str(result.first_valid_index()) + ' - abaixo do esperado!')
    #Verificando limite superior
    if result.last_valid_index() > valor_maior:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro máximo encontrado: ' +
            str(result.last_valid_index()) + " - acima do esperado!")

    log_tela.warning(variavel + ': ' +
                    str(len(df_errores[variavel].value_counts())) +
                    ' valor(es) incorreto(s) ' +
                    'encontrado(s) nesta variável:\n' +
                    str(df_errores[variavel].value_counts()))
    #log_output.warning("Linhas com erro:\n-----")
    #for index, linha in df_errores.iterrows():
    #    log_output.warning("Linha " + str(index+1) + ': \n => ' +
    #                      str(linha))
else:
    log_tela.info(variavel + ": Nenhum erro encontrado.")

def pre_processamento(df):
    """
    Realiza algumas ações prévias ao processamento dos dados,
    removendo alguns registros e ajustando o dataframe.
    """

log_tela.info('Criando/Renomeando colunas no dataframe principal')

```

```

log_output.info('Renomeando coluna UCOD para UCOD_DOM')
df.rename(columns={'UCOD':'UCOD_DOM'}, inplace=True)

log_tela.info('Verificando se todas as variaveis esperadas'+
              'existem no dataframe.\n' +
              'Caso não exista alguma, ela é criada.')
variaveis = ['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
             'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
             'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
             'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
             'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
             'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
             'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
             'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
             'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
             'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
             'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
             'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
             'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
             'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
             'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
             'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
             'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
             'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
             'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
             'VALOR_EST_AUTO']

for variavel in variaveis:
    if variavel not in df.columns:
        # Se a variável não existe no dataframe então ela é criada
        # com valor padrão de NONE (NA)
        df[variavel] = None
    log_tela.info('Criando a variavel ' + variavel + ' no dataframe')

log_output.info('Reordenando as variáveis')
df = df[['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
          'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
          'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
          'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
          'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
          'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
          'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
          'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
          'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
          'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
          'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
          'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
          'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
          'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
          'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
          'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
          'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
          'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
          'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
          'VALOR_EST_AUTO']]

```

```

        'VALOR_EST_AUTO']]]

log_output.info('\'\n\n=====
log_output.info('\'\n\n=====\\n')

return df

def coordenadas(passo, df):
    """
    Itera sobre os registros do dataframe coords,
    a cada iteração, utiliza o valor da coluna ZONA
    para filtrar o dataframe df em cada um dos tipos de ZONA
    (DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui os valores
    das coordenadas X e Y de cada tipo (CO_DOM_X, CO_ESC_Y, etc) com
    os valores das colunas CO_X e CO_Y do dataframe coords.

    :param passo: número do passo para o log
    :param df: Dafaframe a ser modificado (ex.: od1977)
    :return: retorna o dataframe modificado com todas as coordenadas aplicadas
    """
    log_tela.info("### PASSO " + str(passo) + " - COORDENADAS")

    def coord_aux(row):
        """
        Esta função irá receber uma linha (row) do dataframe coords e
        utilizar os valores desta linha para realizar as alterações
        de coordenadas no dataframe df.
        """
        # Começando pelo tipo DOM, alterando CO_DOM_X e depois CO_DOM_Y
        df.loc[df['ZONA_DOM'] == row['ZONA'], 'CO_DOM_X'] = row['CO_X']
        df.loc[df['ZONA_DOM'] == row['ZONA'], 'CO_DOM_Y'] = row['CO_Y']
        # Agora com o tipo ESC, alterando CO_ESC_X e depois CO_ESC_Y
        df.loc[df['ZONA_ESC'] == row['ZONA'], 'CO_ESC_X'] = row['CO_X']
        df.loc[df['ZONA_ESC'] == row['ZONA'], 'CO_ESC_Y'] = row['CO_Y']
        # Agora com o tipo TRAB1, alterando CO_TRAB1_X e depois CO_TRAB1_Y
        df.loc[df['ZONA_TRAB1'] == row['ZONA'], 'CO_TRAB1_X'] = row['CO_X']
        df.loc[df['ZONA_TRAB1'] == row['ZONA'], 'CO_TRAB1_Y'] = row['CO_Y']
        # Agora com o tipo TRAB2, alterando CO_TRAB2_X e depois CO_TRAB2_Y
        df.loc[df['ZONA_TRAB2'] == row['ZONA'], 'CO_TRAB2_X'] = row['CO_X']
        df.loc[df['ZONA_TRAB2'] == row['ZONA'], 'CO_TRAB2_Y'] = row['CO_Y']
        # Agora com o tipo ORIG, alterando CO_ORIG_X e depois CO_ORIG_Y
        df.loc[df['ZONA_ORIG'] == row['ZONA'], 'CO_ORIG_X'] = row['CO_X']
        df.loc[df['ZONA_ORIG'] == row['ZONA'], 'CO_ORIG_Y'] = row['CO_Y']
        # Agora com o tipo DEST, alterando CO_DEST_X e depois CO_DEST_Y
        df.loc[df['ZONA_DEST'] == row['ZONA'], 'CO_DEST_X'] = row['CO_X']
        df.loc[df['ZONA_DEST'] == row['ZONA'], 'CO_DEST_Y'] = row['CO_Y']

    log_output.info('Lendo arquivo auxiliar de Coordenadas das subzonas')
    coords = pd.read_csv('auxiliares/coord_zonas_1977.csv', sep=';', decimal=',')

    # Esta variável out não é utilizada para nada além de evitar um
    # monte de output que não será utilizado e que é gerado pelo método apply.
    out = coords.apply(coord_aux, axis=1)

```

```
log_output.info('\'\n\n=====\\n')  
return df
```

3.2 Funções Gerais

Função/Variável	Status
passo_ano	OK
passo_dia_sem	OK
passo_ucods	OK

```
In [ ]: log_tela.info('Definindo as funções Gerais')
log_output.info('\n\n=====\\n')
```

```
def passo_ano(passo, df):
    """
    Preenche a coluna "ANO" com valor 1 em todas células
    Categorias:
    /valor/ano_correspondente/
    /-----/-----
    /1/1977/
    /2/1987/
    /3/1997/
    /4/2007/

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - ANO")

    # Definindo valor '1' para todas as células da coluna ANO
    df["ANO"] = 1

    return df

def passo_dia_sem(passo, df):
    """
    Não existe essa informação no banco de dados de 1977, logo, este campo será
    preenchido com 'None' (NA).

    # #####Categorias:
    # Valor/Descrição
    # -----/-----
    # 2/Segunda-Feira
    # 3/Terça-Feira
    # 4/Quarta-Feira
    # 5/Quinta-Feira
    # 6/Sexta-Feira

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
```

```

:return: retorna o dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - DIA_SEM")

# Atribuindo "None" (Nulo/NA) para todos os registros
df['DIA_SEM']= None

return df

def passo_ucods(passo, df):
    """
    Itera sobre os registros do dataframe ucods,
    a cada iteração utiliza o valor da coluna ZONA_REF
    para filtrar o dataframe df em cada um dos tipos de ZONA
    (DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui o valor
    da respectiva UCOD.

:param passo: número do passo para o log
:param df: Dafaframe a ser modificado (ex.: od1977)
:return: retorna o dataframe modificado com todas as UCODS aplicadas
"""
log_tela.info("### PASSO " + str(passo) + " - UCODS")

def ucod_aux(row):
    df.loc[df['ZONA_DOM']==row['ZONA_REF'], 'UCOD_DOM'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_ESC']==row['ZONA_REF'], 'UCOD_ESC'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_TRAB1']==row['ZONA_REF'], 'UCOD_TRAB1'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_TRAB2']==row['ZONA_REF'], 'UCOD_TRAB2'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_ORIG']==row['ZONA_REF'], 'UCOD_ORIG'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_DEST']==row['ZONA_REF'], 'UCOD_DEST'] = row['UCOD_BUSCADA']

log_output.info('Lendo arquivo auxiliar UCOD')
ucods = pd.read_csv('auxiliares/UCOD-1977.csv', sep=';', decimal=',')

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = ucods.apply(ucod_aux, axis=1)

# Verificando intervalo de valores - condições:
# "UCOD_XXX >= 1" E "UCOD_XXX <= 67"
for tipo in ['DOM', 'ESC', 'TRAB1', 'TRAB2', 'ORIG', 'DEST']:
    verifica_range(df, 'UCOD_' + tipo, 1, 67)
log_output.info('\n\n=====\\n')

return df

```

3.3 Funções do Domicílio

Função/Variável	Status
passo_zona_dom	OK
passo_subzona_dom	OK

Função/Variável	Status
passo_mun_dom	OK
passo_f_dom	OK
passo_fe_dom	OK
passo_tipo_dom	OK

```
In [ ]: log_tela.info('Definindo as funções do Domicílio')
log_output.info('\n=====\\n')

def passo_zona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 243

    [Teste: Checar se existe algum número < 1 ou > 243.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_DOM")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_DOM']==0,'ZONA_DOM'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_DOM >= 1" E "ZONA_DOM <= 243"
    verifica_range(df, 'ZONA_DOM', 1, 243)
    log_output.info('\n=====\\n')

    return df

def passo_subzona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 633

    [Teste: Checar se existe algum número < 1 ou > 633.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DOM")
```

```

# Substituindo valor 0 por None
df.loc[df['SUBZONA_DOM']==0,'SUBZONA_DOM'] = None

# Verificando intervalo de valores - condições:
# "SUBZONA_DOM >= 1" E "SUBZONA_DOM <= 633"
verifica_range(df, 'SUBZONA_DOM', 1, 633)
log_output.info('\n\n=====\\n')

return df

def passo_mun_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 27

    [Teste: Checar se existe algum número < 1 ou > 27.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_DOM")

    # Substituindo valor 0 por None
    df.loc[df['MUN_DOM']==0,'MUN_DOM'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_DOM >= 1" E "MUN_DOM <= 27"
    verifica_range(df, 'MUN_DOM', 1, 27)
    log_output.info('\n\n=====\\n')

    return df

def passo_f_dom(passo, df):
    """
    Checar se existe algum erro na coluna "F_DOM"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro do Domicilio

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - F_DOM")

```

```

verifica_dummy(df, 'F_DOM')
log_output.info('\n\n=====\\n')

return df

def passo_fe_dom(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_DOM"

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_DOM")
    log_output.info('\n\n=====\\n')

    return df

def passo_tipo_dom(passo, df):
    """
    Substituir **0** por **None (NA)**
    Substituir **2** por **0**

    # #####Categorias anteriores
    # Valor / Descrição
    # ----/----
    # 0/Não respondeu
    # 1/Particular
    # 2/Coletivo

    # #####Categorias novas
    # Valor / Descrição
    # ----/----
    # 0/Coletivo
    # 1/Particular

    [Teste: Checar se existe algum número < 0 ou > 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - TIPO_DOM")

    df.loc[df['TIPO_DOM']==0,'TIPO_DOM'] = None
    df.loc[df['TIPO_DOM']==2,'TIPO_DOM'] = 0

    # Verificando intervalo de valores - condições:
    # "TIPO_DOM >= 0" E "TIPO_DOM <= 1"
    verifica_dummy(df, 'TIPO_DOM')
    log_output.info('\n\n=====\\n')

```

```
return df
```

3.4 Funções da Família

Função/Variável	Status
passo_tot_fam	OK
passo_f_fam	OK
passo_fe_fam	OK
passo_cond_mora	OK
passo_qt_auto	OK
passo_qt_bici	OK
passo_qt_moto	OK
passo_ren_fam	OK
passo_cd_renfam	OK

```
In [ ]: log_tela.info('Definindo as funções da Família')
log_output.info('\n\n=====\\n')

def passo_tot_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "TOT_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - TOT_FAM")
    log_output.info('\\n\\n=====\\n')

    return df

def passo_f_fam(passo, df):
    """
    Checar se existe algum erro na coluna "F_FAM"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Família

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_FAM")

    verifica_dummy(df, 'F_FAM')
```

```

log_output.info('\n\n=====\\n')
return df

def passo_fe_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_FAM")
    log_output.info('\n\n=====\\n')

    return df

def passo_cond_mora(passo, df):
    """
    Substituir valores da coluna "COND_MORA"

    # * Substituir todos valores **1** por **2**
    # * Substituir todos valores **3** por **1**
    # * Substituir todos valores **4** por **3**
    # * Substituir todos valores **5** por **3**
    # * Substituir todos valores **9** por **None**
    # * Substituir todos valores **0** por **None**

    # #####Categorias anteriores
    # Valor/Descrição
    # ----/----
    # 0/Não respondeu
    # 1/Própria paga
    # 2/Própria em pagamento
    # 3/Alugada
    # 4/Cedida
    # 5/Outro
    # 9/Erro no preenchimento do banco de dados

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Alugada
    # 2/Própria
    # 3/Outros

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - COND_MORA")

```

```

# Substituindo valor 1 por 2
df.loc[df['COND_MORA']==1, 'COND_MORA'] = 2
# Substituindo valor 3 por 1
df.loc[df['COND_MORA']==3, 'COND_MORA'] = 1
# Substituindo valor 4 por 3
df.loc[df['COND_MORA']==4, 'COND_MORA'] = 3
# Substituindo valor 5 por 3
df.loc[df['COND_MORA']==5, 'COND_MORA'] = 3
# Substituindo valor 9 por None
df.loc[df['COND_MORA']==9, 'COND_MORA'] = None
# Substituindo valor 0 por None
df.loc[df['COND_MORA']==0, 'COND_MORA'] = None

# Verificando intervalo de valores - condições:
# "COND_MORA >= 1" E "COND_MORA <= 3"
verifica_range(df, 'COND_MORA', 1, 3)
log_output.info('\n\n=====\\n')

return df

def passo_qt_auto(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "QT_AUTO"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - QT_AUTO")
    log_output.info('\n\n=====\\n')

    return df

def passo_qt_bici(passo, df):
    """
    Não existe essa informação no banco de dados de 1977, logo,
    este campo será preenchido com 'None'.
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    df['QT_BICI'] = None

    # Contando "QT_BICI" para verificar os valores após o procedimento
    log_tela.info('### PASSO ' + str(passo) + ' - QT_BICI' + '\n'
                  'Total de registros: ' +
                  str(len(df['QT_BICI'])) + '\n' +
                  'Soma de bicicletas "nulas": ' +
                  str(df['QT_BICI'].isnull().sum()))

    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_qt_moto(passo, df):
    """
    Não existe essa informação no banco de dados de 1977, logo,
    este campo será preenchido com 'None'.
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    df['QT_MOTO'] = None

    # Contando "QT_MOTO" para verificar os valores após o procedimento
    log_tela.info('### PASSO ' + str(passo) + ' - QT_MOTO' + '\n'
                  , ' ' + 'Total de registros: ' +
                  str(len(df['QT_MOTO'])) + '\n' +
                  ' ' + 'Soma de motos "nulas": ' +
                  str(df['QT_MOTO'].isnull().sum()))

    log_output.info('\n\n=====\\n')

    return df

def passo_ren_fam(passo, df, deflator):
    """
    Corriga a renda familiar de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_FAM")

    df['REN_FAM'] = df['REN_FAM'] * deflator

    return df

def passo_cd_renfam(passo, df):
    """
    Substituir valores da coluna "CD_RENFAM"

    * Substituir todos valores **1** por **2**
    * Substituir todos valores **3** por **2**
        Quando categoria original for 0 (respondeu)
            checar no campo REN_FAM se o valor é nulo.
        Se for nulo, manter na categoria 0 (Renda Familiar Declarada como Zero),
        senão, mover para a categoria 1 (Renda Familiar Declarada e
        Maior que Zero).

    # #####Categorias anteriores
    # Valor/Descrição
    """

```

```

# ----/----
# 0/Respondeu
# 1/Não Sabe
# 2/Não Respondeu
# 3/Não se Aplica

# #####Categorias novas
# Valor/Descrição
# ----/----
# 0/Renda Familiar Declarada como Zero
# 1/Renda Familiar Declarada e Maior que Zero
# 2/Renda Atribuída

[Teste: Checar se existe algum número < 0 ou > 2.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe corrigido
"""
log_tela.info("### PASSO " + str(passo) + " - CD_RENFAM")

# Substituindo valor 1 (Não sabe) por 2 (Atribuída)
df.loc[df['CD_RENFAM']==1,'CD_RENFAM'] = 2

# Substituindo valor 3 (Não se Aplica) por 2 (Atribuída)
df.loc[df['CD_RENFAM']==3,'CD_RENFAM'] = 2

# Correção de REN_FAM
# REN_FAM só pode ser igual a zero se tiver sido declarada assim.
df.loc[df['REN_FAM']==0,'CD_RENFAM'] = 0

# Dividindo a categoria '0', "Respondeu", em:
#   - 0 "Renda Familiar Declarada como Zero" e
#   - 1 "Renda Familiar Declarada e Maior que Zero"
df.loc[(df['CD_RENFAM'] == 0) &
       (df['REN_FAM'] != 0) &
       (df['REN_FAM'].notnull()), 'CD_RENFAM'] = 1

# Verificando intervalo de valores - condições:
# "CD_RENFAM >= 0" E "CD_RENFAM <= 2"
verifica_range(df, 'CD_RENFAM', 0, 2)
log_output.info('\n\n=====\\n')

return df

```

3.5 Funções da pessoa

Função/Variável	Status
passo_f_pess	OK
passo_fe_pess	OK
passo_sit_fam	OK
passo_idade	OK
passo_sexo	OK
passo_grau_instr	OK
passo_ocup	OK
passo_setor_ativ	OK
passo_ren_ind	OK
passo_cd_renind	OK
passo_estuda	OK

```
In [ ]: log_tela.info('Definindo as funções da Pessoa')
log_output.info('\n\n=====\\n')

def passo_f_pess(passo, df):
    """
    Checar se existe algum erro na coluna "F_PESS"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Pessoa

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_PESS")

    verifica_dummy(df, 'F_PESS')
    log_output.info('\n\n=====\\n')

    return df

def passo_fe_pess(passo, df):
    """
    Nada há que se fazer em relação aos dados das colunas "FE_PESS"
    :param passo: Número do passo atual para registro/log
    :param df:
```

```

:return:
"""
log_tela.info("### PASSO " + str(passo) + " - FE_PESS")
log_output.info('\\n\\n=====\\n')

return df

def passo_sit_fam(passo, df):
"""
Substituir todos valores **5** por **4**
Substituir todos valores **6** por **5**
Substituir todos valores **7** por **6**
Substituir todos valores **9** por **6**
Substituir todos valores **0** por **6** quando DURACAO!=0

# #####Categorias anteriores
# Valor/Descrição
# ----/----
# 1/Chefe
# 2/Cônjuge
# 3/Filho(a)
# 4/Parente
# 5/Agregado
# 6/Empregado Residente
# 7/Visitante não Residente

# #####Categorias novas:
# Valor/Descrição
# ----/----
# 0/ Não Respondeu/Não fez viagem
# 1/ Pessoa Responsável
# 2/ Cônjuge/Companheiro(a)
# 3/ Filho(a)/Enteado(a)
# 4/ Outro Parente / Agregado
# 5/ Empregado Residente
# 6/ Outros (visitante não residente / parente do empregado)

[Teste: Checar se existe algum número < 0 ou > 6.
 Se encontrar, retornar erro indicando em qual linha.]"""

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - SIT_FAM")

# Substituindo valor 5 por 4
df.loc[df['SIT_FAM']==5,'SIT_FAM'] = 4
# Substituindo valor 6 por 5
df.loc[df['SIT_FAM']==6,'SIT_FAM'] = 5
# Substituindo valor 7 por 6
df.loc[df['SIT_FAM']==7,'SIT_FAM'] = 6
# Substituindo valor 9 por 6
df.loc[df['SIT_FAM']==9,'SIT_FAM'] = 6

```

```

# Substituindo valor 0 por 6 quando DURACAO != 0
df.loc[(df['SIT_FAM']==9) & (df['DURACAO']!=0), 'SIT_FAM'] = 6

# Verificando intervalo de valores - condições:
# "SIT_FAM >= 0" E "SIT_FAM <= 6"
verifica_range(df, 'SIT_FAM', 0, 6)
log_output.info('\n\n=====\\n')

return df

def passo_idade(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "IDADE"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - IDADE")
    log_output.info('\n\n=====\\n')

    return df

def passo_sexo(passo, df):
    """
    #####Categorias anteriores
    Valor/Descrição
    ----/----
    0/Não respondeu (-> NA)
    1/Masculino
    2/Feminino

    #####Categorias novas
    Valor/Descrição
    ----/----
    0/Masculino
    1/Feminino

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SEXO")

    # Substituindo valor 0 por None
    df.loc[df['SEXO']==0, 'SEXO'] = None
    # Substituindo valor 1 por 0
    df.loc[df['SEXO']==1, 'SEXO'] = 0
    # Substituindo valor 2 por 1
    df.loc[df['SEXO']==2, 'SEXO'] = 1

```

```

# Verificando intervalo de valores - condições:
# "SEXO >= 0" E "SEXO <= 1"
verifica_dummy(df, 'SEXO')
log_output.info('\'\n\'=====\'\n')

return df


def passo_grau_instr(passo, df):
    """
    Substituir valores da coluna "GRAU_INSTR"

    Substituir todos valores **2** por **1**
    Substituir todos valores **3** por **1**
    Substituir todos valores **4** por **1**
    Substituir todos valores **5** por **2**
    Substituir todos valores **6** por **2**
    Substituir todos valores **7** por **3**
    Substituir todos valores **8** por **3**
    Substituir todos valores **9** por **4**
    Substituir todos valores **0** por **None**

    #####Categorias anteriores:
    # Valor/Descrição
    # ----/----
    # 1/Sem Instrução
    # 2/Primário Incompleto
    # 3/Promocional Completo
    # 4/Ginasial Incompleto
    # 5/Ginasial Completo
    # 6/Colegial Incompleto
    # 7/Colegial Completo
    # 8/Universitário Incompleto
    # 9/Universitário Completo

    #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Não-Alfabetizado/Fundamental Incompleto
    # 2/Fundamental Completo/Médio Incompleto
    # 3/Médio Completo/Superior Incompleto
    # 4/Superior completo

[Teste: Checar se existe algum número < 1 ou > 4.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - GRAU_INSTR")

# Substituindo valor 2 por 1
df.loc[df['GRAU_INSTR']==2, 'GRAU_INSTR'] = 1
# Substituindo valor 3 por 1

```

```

df.loc[df['GRAU_INSTR']==3, 'GRAU_INSTR'] = 1
# Substituindo valor 4 por 1
df.loc[df['GRAU_INSTR']==4, 'GRAU_INSTR'] = 1
# Substituindo valor 5 por 2
df.loc[df['GRAU_INSTR']==5, 'GRAU_INSTR'] = 2
# Substituindo valor 6 por 2
df.loc[df['GRAU_INSTR']==6, 'GRAU_INSTR'] = 2
# Substituindo valor 7 por 3
df.loc[df['GRAU_INSTR']==7, 'GRAU_INSTR'] = 3
# Substituindo valor 8 por 3
df.loc[df['GRAU_INSTR']==8, 'GRAU_INSTR'] = 3
# Substituindo valor 9 por 4
df.loc[df['GRAU_INSTR']==9, 'GRAU_INSTR'] = 4
# Substituindo valor 0 por None
df.loc[df['GRAU_INSTR']==0, 'GRAU_INSTR'] = None

# Verificando intervalo de valores - condições:
# "GRAU_INSTR >= 1" E "GRAU_INSTR <= 4"
verifica_range(df, 'GRAU_INSTR', 1, 4)
log_output.info('\n\n=====\\n')

return df

def passo_ocup(passo, df):
    """
    Substituir valores da coluna "OCUP"

    Somar 10 em todos valores (para facilitar a lógica).
    Substituir todos valores **10** por **None**
    Substituir todos valores **11** por **7**
    Substituir todos valores **12** por **6**
    Substituir todos valores **13** por **3**
    Substituir todos valores **14** por **5**
    Substituir todos valores **15** por **4**
    Substituir todos valores **16** por **2**
    Substituir todos valores *maiores do que 16* por **1**

    #####Categorias anteriores:
    # Valor/Descrição
    # ----/----
    # 1/Estudante
    # 2/Prendas Domésticas
    # 3/Aposentado
    # 4/Sem Ocupação (nunca trabalhou)
    # 5/Desempregado
    # 6/Em licença
    # 7/em diante/diversas profissões

    #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Tem trabalho
    # 2/Em licença médica
    """

    if passo == 1:
        df['OCUP'] = df['OCUP'].replace(10, None)
        df['OCUP'] = df['OCUP'].replace(11, 7)
        df['OCUP'] = df['OCUP'].replace(12, 6)
        df['OCUP'] = df['OCUP'].replace(13, 3)
        df['OCUP'] = df['OCUP'].replace(14, 5)
        df['OCUP'] = df['OCUP'].replace(15, 4)
        df['OCUP'] = df['OCUP'].replace(16, 2)
        df['OCUP'] = df['OCUP'].replace(17, 1)

    elif passo == 2:
        df['OCUP'] = df['OCUP'].replace(1, None)
        df['OCUP'] = df['OCUP'].replace(2, 1)
        df['OCUP'] = df['OCUP'].replace(3, 1)
        df['OCUP'] = df['OCUP'].replace(4, 1)
        df['OCUP'] = df['OCUP'].replace(5, 1)
        df['OCUP'] = df['OCUP'].replace(6, 1)
        df['OCUP'] = df['OCUP'].replace(7, 1)

    return df

```

```

# 3/Aposentado / pensionista
# 4/Desempregado
# 5/Sem ocupação
# 6/Dona de casa
# 7/Estudante

[Teste: Checar se existe algum número < 1 ou > 7.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - OCUP")

df['OCUP'] = df['OCUP'] + 10
# Substituindo valor 10 por None
df.loc[df['OCUP']==10,'OCUP'] = None
# Substituindo valor 11 por 7
df.loc[df['OCUP']==11,'OCUP'] = 7
# Substituindo valor 12 por 6
df.loc[df['OCUP']==12,'OCUP'] = 6
# Substituindo valor 13 por 3
df.loc[df['OCUP']==13,'OCUP'] = 3
# Substituindo valor 14 por 5
df.loc[df['OCUP']==14,'OCUP'] = 5
# Substituindo valor 15 por 4
df.loc[df['OCUP']==15,'OCUP'] = 4
# Substituindo valor 16 por 2
df.loc[df['OCUP']==16,'OCUP'] = 2
# Substituindo valor >16 por 1
df.loc[df['OCUP']>16,'OCUP'] = 1

# Verificando intervalo de valores - condições:
# "OCUP >= 1" E "OCUP <= 7"
verifica_range(df, 'OCUP', 1, 7)
log_output.info('\n\n=====\\n')

return df

def passo_setor_ativ(passo, df):
"""
Substituir valores da coluna "SETOR_ATIV"

Na coluna "SETOR_ATIV", linha i,
ler o valor da linha i da coluna "SETOR_ATIV", daí,
buscar o mesmo valor na coluna "COD" do arquivo setor_ativ-1977.csv.
Ao achar, retornar o valor da mesma linha, só que da coluna "COD_UNIF"

####Categorias anteriores
> ver arquivo .csv

####Categorias novas
Valor/Descrição

```

```

----/----
1/Agrícola
2/Construção Civil
3/Indústria
4/Comércio
5/Administração Pública
6/Serviços de Transporte
7/Doutros serviços
8/Doutros
9/Não se aplica

[Teste: Checar se existe algum número < 1 ou > 9.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - SETOR_ATIV")

log_output.info('Lendo arquivo de referência externa setor_ativ-1977.csv')
df_setor = pd.read_csv('auxiliares/setor_ativ-1977.csv', sep=';', decimal=',')

def setor_aux(row):
    df.loc[df['SETOR_ATIV']==row['COD'], 'SETOR_ATIV'] = row['COD_UNIF']

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = df_setor.apply(setor_aux, axis=1)

# Substituindo valor 0 por None
df.loc[df['SETOR_ATIV']==0, 'SETOR_ATIV'] = None

# Verificando intervalo de valores - condições:
# "SETOR_ATIV >= 1" E "SETOR_ATIV <= 9"
verifica_range(df, 'SETOR_ATIV', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_ren_ind(passo, df, deflator):
    """
    Corriga a renda individual de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_IND")

    df['REN_IND'] = df['REN_IND'] * deflator

    return df

```

```

def passo_cd_renind(passo, df):
    """
        Substituir **2** por None (NA)
        Substituir **3** (erro de preenchimento) por None
        Substituir **1** por **2**
        Substituir **0** por **1**
        Substituir **2** por **0**

        ##### Categorias anteriores efetivas
        Valor Real/ Descrição /Valor Layout
        -----/-----/-----
        0      /Tem renda     /1
        1      /Não tem renda/2
        2      /Não declarou /3

        ##### Categorias novas
        Valor/Descrição
        -----/-----
        0      /Não tem renda
        1      /Tem renda

        [Teste: Checar se existe algum número < 0 ou > 1.
         Se encontrar, retornar erro indicando em qual linha.]
        :param passo: Número do passo atual para registro/log
        :param df:
        :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - CD_RENIND")

    df.loc[df['CD_RENIND']==2, 'CD_RENIND'] = None
    df.loc[df['CD_RENIND']==3, 'CD_RENIND'] = None
    df.loc[df['CD_RENIND']==1, 'CD_RENIND'] = 2
    df.loc[df['CD_RENIND']==0, 'CD_RENIND'] = 1
    df.loc[df['CD_RENIND']==2, 'CD_RENIND'] = 0

    # Verificando intervalo de valores - condições:
    # "CD_RENIND >= 0" E "CD_RENIND <= 1"
    verifica_dummy(df, 'CD_RENIND')
    log_output.info('\'\n\'=====\\n')

    return df


def passo_estuda(passo, df):
    """
        Considera-se que ninguém estuda, exceto quem forneceu informação
        de localização da Escola (Zona/SubZona)

        #####
        # ATENÇÃO #
        # ESTA FUNÇÃO SÓ DEVE SER #
        # EXECUTADA APÓS O PASSO #
        # ZONA_ESC      #
    """

```

```

#####
##### Categorias novas
Valor/Descrição
----/----
0/Não estuda
1/Estuda

[Teste: Checar se existe algum número diferente de 0 ou 1.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe com campo ESCOLA resolvido
"""
log_tela.info("### PASSO " + str(passo) + " - ESTUDA")

df['ESTUDA'] = 0

# Substituindo todos que declararam zona escola diferente de zero
# com campo ESTUDA igual a 1.
df.loc[(df['ZONA_ESC'] != 0)&
       (df['ZONA_ESC'].notnull()), 'ESTUDA'] = 1

verifica_dummy(df, 'ESTUDA')
log_output.info('\n\n=====\\n')

return df

```

3.6 Funções da Viagem

Função/Variável	Status
passo_f_vitag	OK
passo_fe_vitag	OK
passo_zona_esc	OK
passo_subzona_esc	OK
passo_mun_esc	OK
passo_zona_trab1	OK
passo_subzona_trab1	OK
passo_mun_trab1	OK
passo_zona_trab2	OK
passo_subzona_trab2	OK
passo_mun_trab2	OK
passo_zona_orig	OK
passo_subzona_orig	OK
passo_mun_orig	OK
passo_zona_dest	OK
passo_subzona_dest	OK
passo_mun_dest	OK
passo_serv_pas_orig	OK
passo_serv_pas_dest	OK
passo_motivo_orig	OK
passo_motivo_dest	OK
passo_modo1	OK
passo_modo2	OK
passo_modo3	OK
passo_modo4	OK
passo_modo_prin	OK
passo_tipo_est_auto	OK
passo_valor_est_auto	OK
passo_tot_vitag	OK

Obs: O passo_tot_vitag deve ser executado após os passos que geram id's.

```
In [ ]: log_tela.info('Definindo funções referentes às viagens')
log_output.info('\n=====\\n')
```

```
def passo_f_vitag(passo, df):
```

```

"""
Checar se existe algum erro na coluna "F_VIAG"

# #####Categorias
# Valor/Descrição
# ----/----
# 0/Não fez viagem
# 1/Fez viagem

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com F_VIAG modificado
"""
log_tela.info("### PASSO " + str(passo) + " - F_VIAG")

# Setando F_VIAG=0 se DURACAO==0
df.loc[df['DURACAO']==0,'F_VIAG'] = 0

verifica_dummy(df, 'F_VIAG')
log_output.info('\n\n=====\\n')

return df


def passo_fe_viag(passo, df):
"""
# Nada há que se fazer em relação aos dados da coluna "FE_VIAG"
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - FE_VIAG")
log_output.info('\\n\\n=====\\n')

return df


def passo_zona_esc(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 243

[Teste: Checar se existe algum número < 1 ou > 243.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: Sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - ZONA_ESC")

```

```

# Substituindo valor 0 por None
df.loc[df['ZONA_ESC']==0,'ZONA_ESC'] = None

# Verificando intervalo de valores - condições:
# "ZONA_ESC >= 1" E "ZONA_ESC <= 243"
verifica_range(df, 'ZONA_ESC', 1, 243)
log_output.info('\n\n=====\\n')

return df

def passo_subzona_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 633

    [Teste: Checar se existe algum número < 1 ou > 633.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ESC")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_ESC']==0,'SUBZONA_ESC'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_ESC >= 1" E "SUBZONA_ESC <= 633"
    verifica_range(df, 'SUBZONA_ESC', 1, 633)
    log_output.info('\\n\\n=====\\n')

    return df

def passo_mun_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 27

    [Teste: Checar se existe algum número < 1 ou > 27.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ESC")

    # Substituindo valor 0 por None
    df.loc[df['MUN_ESC']==0,'MUN_ESC'] = None

```

```

# Verificando intervalo de valores - condições:
# "MUN_ESC >= 1" E "MUN_ESC <= 27"
verifica_range(df, 'MUN_ESC', 1, 27)
log_output.info('\n\n=====\\n')

return df

def passo_zona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 243

    [Teste: Checar se existe algum número < 1 ou > 243.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """

    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB1']==0, 'ZONA_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_TRAB1 >= 1" E "ZONA_TRAB1 <= 243"
    verifica_range(df, 'ZONA_TRAB1', 1, 243)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 633

    [Teste: Checar se existe algum número < 1 ou > 633.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """

    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_TRAB1']==0, 'SUBZONA_TRAB1'] = None

    # Verificando intervalo de valores - condições:

```

```

# "SUBZONA_TRAB1 >= 1" E "SUBZONA_TRAB1 <= 633"
verifica_range(df, 'SUBZONA_TRAB1', 1, 633)
log_output.info('\n\n=====\\n')

return df

def passo_mun_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 27

    [Teste: Checar se existe algum número < 1 ou > 27.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB1']==0, 'MUN_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB1 >= 1" E "MUN_TRAB1 <= 27"
    # trips that are not work purposed
    verifica_range(df, 'MUN_TRAB1', 1, 27)
    log_output.info('\n\n=====\\n')

    return df

def passo_zona_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 243

    [Teste: Checar se existe algum número < 1 ou > 243.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB2']==0, 'ZONA_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_TRAB2 >= 1" E "ZONA_TRAB2 <= 243"

```

```

verifica_range(df, 'ZONA_TRAB2', 1, 243)
log_output.info('\n\n=====\\n')

return df

def passo_subzona_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 633

    [Teste: Checar se existe algum número < 1 ou > 633.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_TRAB2']==0, 'SUBZONA_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_TRAB2 >= 1" E "SUBZONA_TRAB2 <= 633"
    verifica_range(df, 'SUBZONA_TRAB2', 1, 633)
    log_output.info('\n\n=====\\n')

    return df

def passo_mun_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 27

    [Teste: Checar se existe algum número < 1 ou > 27.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB2']==0, 'MUN_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB2 >= 1" E "MUN_TRAB2 <= 27"
    verifica_range(df, 'MUN_TRAB2', 1, 27)
    log_output.info('\n\n=====\\n')

```

```

        return df

def passo_zona_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 243

    [Teste: Checar se existe algum número < 1 ou > 243.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """

    log_tela.info("### PASSO " + str(passo) + " - ZONA_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_ORIG']==0,'ZONA_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_ORIG >= 1" E "ZONA_ORIG <= 243"
    verifica_range(df, 'ZONA_ORIG', 1, 243)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 633

    [Teste: Checar se existe algum número < 1 ou > 633.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """

    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_ORIG']==0,'SUBZONA_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_ORIG >= 1" E "SUBZONA_ORIG <= 633"
    verifica_range(df, 'SUBZONA_ORIG', 1, 633)
    log_output.info('\n\n=====\\n')

    return df

```

```

def passo_mun_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 27

    [Teste: Checar se existe algum número < 1 ou > 27.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['MUN_ORIG']==0,'MUN_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_ORIG >= 1" E "MUN_ORIG <= 27"
    verifica_range(df, 'MUN_ORIG', 1, 27)
    log_output.info('\n\n=====\\n')

    return df


def passo_zona_dest(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 243

    [Teste: Checar se existe algum número < 1 ou > 243.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_DEST")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_DEST']==0,'ZONA_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_DEST >= 1" E "ZONA_DEST <= 243"
    verifica_range(df, 'ZONA_DEST', 1, 243)
    log_output.info('\n\n=====\\n')

    return df

```

```

def passo_subzona_dest(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 633

    [Teste: Checar se existe algum número < 1 ou > 633.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DEST")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_DEST']==0, 'SUBZONA_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_DEST >= 1" E "SUBZONA_DEST <= 633"
    verifica_range(df, 'SUBZONA_DEST', 1, 633)
    log_output.info('\n\n=====\\n')

    return df


def passo_mun_dest(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 27

    [Teste: Checar se existe algum número < 1 ou > 27.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_DEST")

    # Substituindo valor 0 por None
    df.loc[df['MUN_DEST']==0, 'MUN_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_DEST >= 1" E "MUN_DEST <= 27"
    verifica_range(df, 'MUN_DEST', 1, 27)
    log_output.info('\n\n=====\\n')

    return df


def passo_serv_pas_orig(passo, df):
    """

```

Atribuir valor à variável dummy que identifica se o motivo origem é servir passageiro(a)

```
#####
#      ATENÇÃO      #
#  ESTE PASSO DEVE  #
#  SER EXECUTADO   #
#  ANTES DO PASSO  #
#      MOTIVO_ORIG  #
#####

# Atribuir 1 quando "MOTIVO_ORIG" for igual a 9

####Categorias novas
Valor/Descrição
----/----
0/Não
1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_ORIG")

df['SERV_PAS_ORIG'] = 0

df.loc[df['MOTIVO_ORIG']==9, 'SERV_PAS_ORIG'] = 1

verifica_dummy(df, 'SERV_PAS_ORIG')
log_output.info('\n\n=====\\n')

return df


def passo_serv_pas_dest(passo, df):
    """
Atribuir valor à variável dummy que identifica se o motivo destino é servir passageiro(a)

#####
#      ATENÇÃO      #
#  ESTE PASSO DEVE  #
#  SER EXECUTADO   #
#  ANTES DO PASSO  #
#      MOTIVO_DEST   #
#####

# Atribuir 1 quando "MOTIVO_DEST" for igual a 9

####Categorias novas
Valor/Descrição
----/----
0/Não
```

```

1 /Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_DEST")

df['SERV_PAS_DEST'] = 0

df.loc[df['MOTIVO_DEST']==9, 'SERV_PAS_DEST'] = 1

verifica_dummy(df, 'SERV_PAS_DEST')
log_output.info('\n\n=====\\n')

return df

def passo_motivo_orig(passo, df):
"""
Substituir todos valores **6** por **11**
Substituir todos valores **7** por **6**
Substituir todos valores **8** por **7**
Substituir todos valores **10** por **8**
Substituir todos valores **11** por **9**
Substituir todos valores **0** por **None**

# #####Categorias anteriores
# Valor/Descrição
# ----/----
# 1/Trabalho/Indústria
# 2/Trabalho/Comércio
# 3/Trabalho/Serviços
# 4/Escola/Educação
# 5/Compras
# 6/Negócios
# 7/Médico/Dentista/Saúde
# 8/Recreação/Visitas
# 9/Servir Passageiro
# 10/Residência

# #####Categorias novas
# Valor/Descrição
# ----/----
# 1/Trabalho/Indústria
# 2/Trabalho/Comércio
# 3/Trabalho/Serviços
# 4/Educação
# 5/Compras
# 6/Saúde
# 7/Lazer
# 8/Residência
# 9/Outros

```

```

[Teste: Checar se existe algum número < 1 ou > 9.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - MOTIVO_ORIG")

# Substituindo valor 6 por 11
df.loc[df['MOTIVO_ORIG']==6,'MOTIVO_ORIG'] = 11
# Substituindo valor 7 por 6
df.loc[df['MOTIVO_ORIG']==7,'MOTIVO_ORIG'] = 6
# Substituindo valor 8 por 7
df.loc[df['MOTIVO_ORIG']==8,'MOTIVO_ORIG'] = 7
# Substituindo valor 10 por 8
df.loc[df['MOTIVO_ORIG']==10,'MOTIVO_ORIG'] = 8
# Substituindo valor 11 por 9
df.loc[df['MOTIVO_ORIG']==11,'MOTIVO_ORIG'] = 9
# Substituindo valor 0 por None
df.loc[df['MOTIVO_ORIG']==0,'MOTIVO_ORIG'] = None

# Verificando intervalo de valores - condições:
# "MOTIVO_ORIG >= 1" E "MOTIVO_ORIG <= 9"
verifica_range(df, 'MOTIVO_ORIG', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_motivo_dest(passo, df):
"""
Substituir todos valores **6** por **11**
Substituir todos valores **7** por **6**
Substituir todos valores **8** por **7**
Substituir todos valores **10** por **8**
Substituir todos valores **11** por **9**
Substituir todos valores **0** por **None**

# #####Categorias anteriores
# Valor/Descrição
# ----/
# 1/Trabalho/Indústria
# 2/Trabalho/Comércio
# 3/Trabalho/Serviços
# 4/Escola/Educação
# 5/Compras
# 6/Negócios
# 7/Médico/Dentista/Saúde
# 8/Recreação/Visitas
# 9/Servir Passageiro
# 10/Residência

# #####Categorias novas
# Valor/Descrição
"""

```

```

# ----/----
# 1/Trabalho/Indústria
# 2/Trabalho/Comércio
# 3/Trabalho/Serviços
# 4/Educação
# 5/Compras
# 6/Saúde
# 7/Lazer
# 8/Residência
# 9/Outros

[Teste: Checar se existe algum número < 1 ou > 9.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - MOTIVO_DEST")

# Substituindo valor 6 por 11
df.loc[df['MOTIVO_DEST']==6,'MOTIVO_DEST'] = 11
# Substituindo valor 7 por 6
df.loc[df['MOTIVO_DEST']==7,'MOTIVO_DEST'] = 6
# Substituindo valor 8 por 7
df.loc[df['MOTIVO_DEST']==8,'MOTIVO_DEST'] = 7
# Substituindo valor 10 por 8
df.loc[df['MOTIVO_DEST']==10,'MOTIVO_DEST'] = 8
# Substituindo valor 11 por 9
df.loc[df['MOTIVO_DEST']==11,'MOTIVO_DEST'] = 9
# Substituindo valor 0 por None
df.loc[df['MOTIVO_DEST']==0,'MOTIVO_DEST'] = None

# Verificando intervalo de valores - condições:
# "MOTIVO_DEST >= 1" E "MOTIVO_DEST <= 9"
verifica_range(df, 'MOTIVO_DEST', 1, 9)
log_output.info(' \n\n=====\\n')

return df

def passo_modo1(passo, df):
"""
Substituir todos valores **0** por **None**

# #####Categorias anteriores
# Valor/Descrição
# ----/----
# 1/Ônibus trólebus
# 2/Ônibus Escolar / Empresa
# 3/Dirigindo Automóvel
# 4/Passageiro de Automóvel
# 5/Táxi
# 6/Lotação / Perua
# 7/Metrô

```

```

# 8/Trem
# 9/Motocicleta
# 10/Bicicleta
# 11/A Pé
# 12/Outros

# #####Categorias novas
# Valor/Descrição
# ----/----
# 1/Ônibus
# 2/Ônibus Escolar / Empresa
# 3/Dirigindo Automóvel
# 4/Passageiro de Automóvel
# 5/Táxi
# 6/Lotação / Perua / Van / Microônibus
# 7/Metrô
# 8/Trem
# 9/Moto
# 10/Bicicleta
# 11/A Pé
# 12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```

log_tela.info("### PASSO " + str(passo) + " - MODO1")

# Substituindo valor 0 por None
df.loc[df['MODO1']==0,'MODO1'] = None

# Verificando intervalo de valores - condições:
# "MODO1 >= 1" E "MODO1 <= 12"
verifica_range(df, 'MODO1', 1, 12)
log_output.info('\'\n\'=====\\n')

return df
```

def passo_modo2(passo, df):

"""

Substituir todos valores **0** por **None**

```

# #####Categorias anteriores
# Valor/Descrição
# ----/----
# 1/Ônibus trólebus
# 2/Ônibus Escolar / Empresa
# 3/Dirigindo Automóvel
# 4/Passageiro de Automóvel
# 5/Táxi
# 6/Lotação / Perua
```

```

# 7/Metrô
# 8/Trem
# 9/Motocicleta
# 10/Bicicleta
# 11/A Pé
# 12/Outros

# #####Categorias novas
# Valor/Descrição
# ----/----
# 1/Ônibus
# 2/Ônibus Escolar / Empresa
# 3/Dirigindo Automóvel
# 4/Passageiro de Automóvel
# 5/Táxi
# 6/Lotação / Perua / Van / Microônibus
# 7/Metrô
# 8/Trem
# 9/Moto
# 10/Bicicleta
# 11/A Pé
# 12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```

log_tela.info("### PASSO " + str(passo) + " - MODO2")

# Substituindo valor 0 por None
df.loc[df['MODO2']==0,'MODO2'] = None

# Verificando intervalo de valores - condições:
# "MODO2 >= 1" E "MODO2 <= 12"
verifica_range(df, 'MODO2', 1, 12)
log_output.info(' \n\n=====\\n')

return df

def passo_modo3(passo, df):
    """
Substituir todos valores **0** por **None**

# #####Categorias anteriores
# Valor/Descrição
# ----/----
# 1/Ônibus trólebus
# 2/Ônibus Escolar / Empresa
# 3/Dirigindo Automóvel
# 4/Passageiro de Automóvel
# 5/Táxi
# 6/Lotação / Perua

```

```

# 7/Metrô
# 8/Trem
# 9/Motocicleta
# 10/Bicicleta
# 11/A Pé
# 12/Outros

# #####Categorias novas
# Valor/Descrição
# ----/----
# 1/Ônibus
# 2/Ônibus Escolar / Empresa
# 3/Dirigindo Automóvel
# 4/Passageiro de Automóvel
# 5/Táxi
# 6/Lotação / Perua / Van / Microônibus
# 7/Metrô
# 8/Trem
# 9/Moto
# 10/Bicicleta
# 11/A Pé
# 12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MODO3")

# Substituindo valor 0 por None
df.loc[df['MODO3']==0,'MODO3'] = None

# Verificando intervalo de valores - condições:
# "MODO3 >= 1" E "MODO3 <= 12"
verifica_range(df, 'MODO3', 1, 12)
log_output.info('\n\n=====\\n')

return df

def passo_modo4(passo, df):
"""
Nada há que se fazer em relação à coluna "MODO4" - não há dados de 1977,
coluna permanecerá vazia
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MODO4")

df['MODO4'] = None

log_output.info('\n\n=====\\n')

```

```

    return df

def passo_modo_prin(passo, df):
    """
    Substituir todos valores **0** por **None** 

    # #####Categorias anteriores
    # Valor/Descrição
    # ----/----
    # 1/Ônibus trólebus
    # 2/Ônibus Escolar / Empresa
    # 3/Dirigindo Automóvel
    # 4/Passageiro de Automóvel
    # 5/Táxi
    # 6/Lotação / Perua
    # 7/Metrô
    # 8/Trem
    # 9/Motocicleta
    # 10/Bicicleta
    # 11/A Pé
    # 12/Outros

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Ônibus
    # 2/Ônibus Escolar / Empresa
    # 3/Dirigindo Automóvel
    # 4/Passageiro de Automóvel
    # 5/Táxi
    # 6/Lotação / Perua / Van / Microônibus
    # 7/Metrô
    # 8/Trem
    # 9/Moto
    # 10/Bicicleta
    # 11/A Pé
    # 12/Outros

    [Teste: Checar se existe algum número < 1 ou > 12.
     Se encontrar, retornar erro indicando em qual linha.] 
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MODO_PRIN")

# Substituindo valor 0 por None
df.loc[df['MODO_PRIN']==0,'MODO_PRIN'] = None

# Verificando intervalo de valores - condições:
# "MODO_PRIN >= 1" E "MODO_PRIN <= 12"
verifica_range(df, 'MODO_PRIN', 1, 12)
log_output.info('\'\n\'=====\\n')

```

```

    return df

def passo_tipo_vitag(passo, df):
    """
    * Substituir os valores **0** por **None** 

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Coletivo
    # 2/Individual
    # 3/A pé

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - TIPO_VIAG")

    # Substituindo valor 0 por None
    df.loc[df['TIPO_VIAG']==0,'TIPO_VIAG'] = None

    # Verificando intervalo de valores - condições:
    # "MODO_PRIN >= 1" E "MODO_PRIN <= 3"
    verifica_range(df, 'TIPO_VIAG', 1, 3)

    log_output.info('\n\n=====\\n')
    return df


def passo_tipo_est_auto(passo, df):
    """
    Substituir valores da coluna "TIPO_EST_AUTO"

    Substituir todos valores **1** por **5**
    Substituir todos valores **2** por **2**
    Substituir todos valores **3** por **2**
    Substituir todos valores **4** por **3**
    Substituir todos valores **5** por **5**
    Substituir todos valores **6** por **4**
    Substituir todos valores **7** por **1**

    # #####Categorias anteriores
    # Valor/Descrição
    # ----/----
    # 1/Zona Azul / Parqímetro
    # 2/Estacionamento Avulso

```

```

# 3/Estacionamento Mensal
# 4/Estacionamento Próprio
# 5/Meio-Fio / Logradouro
# 6/Estacionamento Patrocinado
# 7/Não estacionou

# #####Categorias novas
# Valor/Descrição
# ----/----
# 0/Não Respondeu
# 1/Não Estacionou
# 2/Estacionamento Particular (Avulso / Mensal)
# 3/Estacionamento Próprio
# 4/Estacionamento Patrocinado
# 5/Rua (meio fio / zona azul / zona marrom / parquímetro)

[Teste: Checar se existe algum número < 0 ou > 5.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - TIPO_EST_AUTO")

# Substituindo valor 1 por 5
df.loc[df['TIPO_EST_AUTO']==1, 'TIPO_EST_AUTO'] = 5
# Substituindo valor 3 por 2
df.loc[df['TIPO_EST_AUTO']==3, 'TIPO_EST_AUTO'] = 2
# Substituindo valor 4 por 3
df.loc[df['TIPO_EST_AUTO']==4, 'TIPO_EST_AUTO'] = 3
# Substituindo valor 6 por 4
df.loc[df['TIPO_EST_AUTO']==6, 'TIPO_EST_AUTO'] = 4
# Substituindo valor 7 por 1
df.loc[df['TIPO_EST_AUTO']==7, 'TIPO_EST_AUTO'] = 1

# Verificando intervalo de valores - condições:
# "TIPO_EST_AUTO >= 0" E "TIPO_EST_AUTO <= 5"
verifica_range(df, 'TIPO_EST_AUTO', 0, 5)
log_output.info('\n\n=====\\n')

return df

def passo_valor_est_auto(passo, df, deflator):
"""
Corrigir o valor da coluna "VALOR_EST_AUTO" pelo
deflator passado como parâmetro

:param passo: Número do passo atual para registro/log
:param df:
:param deflator: Deflator a ser utilizado para correção
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - VALOR_EST_AUTO")

```

```

df['VALOR_EST_AUTO'] = df['VALOR_EST_AUTO'] * deflator

log_output.info('`n`n=====`n')

return df

def passo_tot_viag(passo, df):
    """
    #####
    #      ATENÇÃO      #
    # ESTA FUNÇÃO SÓ DEVE SER #
    # EXECUTADA APÓS A GERAÇÃO#
    # DE TODOS OS ID'S, POIS #
    # HÁ UMA DESCONFIANÇA   #
    # QUANTO À QUALIDADE   #
    #      DESTE DADO       #
    #####
    Calcula e confere o campo TOT_VIAG,
    baseado no maior valor de NO_VIAG para cada pessoa
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com o "TOT_VIAG" corrigido
"""

log_tela.info("### PASSO " + str(passo) + " - TOT_VIAG")
log_tela.warning('`n`n#####`n' +
                  'Este passo DEVE ser executado apenas após a `n' +
                  'execução dos passos que geram os novos IDs`n' +
                  '#####`n' )

# 'Calculando' o máximo de viagens para cada indivíduo
# Agrupa por ID_PESS e encontra o máximo para NO_VIAG.
# O resultado é um objeto do tipo "Series" cujo index é
# ID_PESS e a variável é NO_VIAG. Em seguida transforma
# esse objeto num DataFrame e depois renomeia a coluna
# NO_VIAG para MAX_VIAG.
df_max = pd.DataFrame(
    df.groupby('ID_PESS')['NO_VIAG'].max()).rename(
        columns={'NO_VIAG': 'MAX_VIAG'})
# Criando um novo dataframe auxiliar apenas com ID_PESS, apenas para ficar mais leve

# O passo seguinte é atribuir o MAX_VIAG à coluna TOT_VIAG
# do dataframe df, linkando por ID_PESS. Isso é feito usando o
# método 'merge' da biblioteca pandas.
df['TOT_VIAG'] = pd.merge(df, df_max, how='left', left_on='ID_PESS',
                           right_index=True)['MAX_VIAG']

log_output.info('TOT_VIAG:`n`n' +
                '      Situação final dos dados: `n' +
                str(df['TOT_VIAG'].describe()) + '\n' +
                '      TOT_VIAG: Situação final dos dados: `n' +
                str(df['TOT_VIAG'].value_counts()))

```

```

# Agora uma função que irá verificar se para todo "ID_PESS"
# o "TOT_VIAG" é igual ao 'NO_VIAG' máximo.
def verifica_no_viag_tot_viag(row):
    if row['NO_VIAG'] != row['TOT_VIAG']:
        log_output.warning('TOT_VIAG: Erro encontrado na linha '
                           + str(row) + ':\n'
                           + ' => ' + row
                           )
df.loc[:, ['ID_PESS', 'NO_VIAG', 'TOT_VIAG']]\
    .groupby('ID_PESS')\
    .agg({'NO_VIAG': 'max', 'ID_PESS': 'max', 'TOT_VIAG': 'max'})\
    .apply(verifica_no_viag_tot_viag, axis=1)
log_output.info('\n\n=====\\n')

return df

```

3.7 Funções que geram os “NO”s e os “ID”s

Função/Variável	Status
gera_nos_e_ids	OK

```
In [ ]: log_tela.info('Definindo funções que geram os "NO"s e os "ID"s')
log_output.info('\n\n=====\\n')
```

```
def gera_nos_e_ids(passo, df):
```

```
"""

```

Esta função gera todos os IDs e todos os NOs das variáveis:
 DOM (domicílio), FAM (família), PESS (pessoa) e VIAG (viagem)
 A ordem de geração é:

NO_DOM, ID_DOM, NO_FAM, ID_FAM, NO_PESS, ID_PESS, NO_VIAG, ID_VIAG
 Esta ordem é fixa pois cada uma dessas variáveis depende da geração
 da variável anterior.

Os NOs são gerados como se fossem subíndices.

No caso dos domicílios, eles são contabilizados dentro de cada zona.
 Assim, cada novo domicílio que aparece na listagem recebe um número
 que representa sua posição na sequência de domicílios dentro da zona
 na qual ele está contido.

As famílias seguem o mesmo raciocínio, com relação ao domicílio, as
 pessoas com relação às famílias e as viagens com relação às pessoas.

Deve-se apenas tomar cuidado com as viagens, pois existem pessoas que
 não realizaram viagens. Neste caso, estes registros devem contabilizar
 o NO_VIAG como zero, e não como 1. Para contemplar estes casos, vamos
 utilizar a variável F_VIAG, que representa quando há ou não viagem
 naquele registro (F_VIAG==1 tem viagem, F_VIAG==0 não tem viagem).
 Assim, o que precisamos fazer para calcular o NO_VIAG é agrupar os
 registros por pessoa (ID_PESS) e, dentro de cada agrupamento, somar
 o valor de F_VIAG registro a registro (linha por linha)
 cumulativamente. No final do processo, atribui-se zero a NO_VIAG
 quando F_VIAG for igual a zero.

```
"""

```

```
def gera_id_dom(row):
```

```
"""

```

Gera o ID_DOM baseado no 'ANO', na 'ZONA_DOM' e no 'NO_DOM'
 O argumento passado é a "linha".

Uso:

```
df['ID_DOM'] = df.apply(gera_id_dom, axis=1)
```

Retorna: ID_DOM da respectiva linha

```
"""

```

Formatando o ano para string

```
ano = str(row['ANO'])
```

Formatando a zona para ficar como string com 3 caracteres

```
zona = str('%03d' % row['ZONA_DOM'])
```

```

# Formatando no_dom para ficar como string com 4 caracteres
no_dom = str('%04d' % row['NO_DOM'])

# Retornando o número inteiro correspondente à string
# concatenada de ano + zona + no_dom
return ano + zona + no_dom


def gera_id_fam(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
    Uso:
        df['ID_FAM'] = df.apply(gera_ID_FAM, axis=1)
    Retorna: ID_FAM da respectiva linha
    """
    # Formatando id_dom como string
    id_dom = str(row['ID_DOM'])

    # Formatando no_fam como string para ficar com 2 caracteres
    no_fam = str('%02d' % row['NO_FAM'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_DOM com NO_FAM
    return id_dom + no_fam


def gera_id_pess(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
    Uso:
        df['ID_PESS'] = df.apply(gera_ID_PESS, axis=1)
    Retorna: ID_PESS da respectiva linha
    """
    # Formatando id_fam como string
    id_fam = str(row['ID_FAM'])

    # Formatando no_pess como string para ficar com 2 caracteres
    no_pess = str('%02d' % row['NO_PESS'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_FAM com NO_PESS
    return id_fam + no_pess


def gera_id_viag(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
    Uso:
        df['ID_VIAG'] = df.apply(gera_ID_VIAG, axis=1)
    Retorna ID_VIAG da respectiva linha
    """
    # Formatando id_pess como string
    id_pess = str(row['ID_PESS'])

```

```

# Formatando no_viag como string para ficar com 2 caracteres
no_viag = str('%02d' % row['NO_VIAG'])

# Retornando o número inteiro correspondente à string
# concatenada de ID_PESS com NO_VIAG
return id_pess + no_viag

#gera NO_DOM
log_tela.info("Gerando NO_DOM")
df['NO_DOM'] = df.groupby('ZONA_DOM', sort=False)['F_DOM'].cumsum()
log_tela.info("NO_DOM gerado")
log_tela.info("Gerando ID_DOM")
#gera ID_DOM
df['ID_DOM'] = df.apply(gera_id_dom, axis=1)
log_tela.info("ID_DOM gerado")

#gera NO_FAM
log_tela.info("Gerando NO_FAM")
df['NO_FAM'] = df.groupby('ID_DOM', sort=False)['F_FAM'].cumsum()
log_tela.info("NO_FAM gerado")
log_tela.info("Gerando ID_FAM")
#gera ID_FAM
df['ID_FAM'] = df.apply(gera_id_fam, axis=1)
log_tela.info("ID_FAM gerado")

#gera NO_PESS
log_tela.info("Gerando NO_PESS")
df['NO_PESS'] = df.groupby('ID_FAM', sort=False)['F_PESS'].cumsum()
log_tela.info("NO_PESS gerado")
log_tela.info("Gerando ID_PESS")
#gera ID_PESS
df['ID_PESS'] = df.apply(gera_id_pess, axis=1)
log_tela.info("ID_PESS gerado")

#gera NO_VIAG
log_tela.info("Gerando NO_VIAG")
df['NO_VIAG'] = df.groupby('ID_PESS', sort=False)['F_VIAG'].cumsum()
# Verificando se F_VIAG == 0 e zerando NO_VIAG nesse caso
df.loc[df['F_VIAG'] == 0, 'NO_VIAG'] = 0
log_tela.info("NO_VIAG gerado")
log_tela.info("Gerando ID_VIAG")
#gera ID_VIAG
df['ID_VIAG'] = df.apply(gera_id_viag, axis=1)
log_tela.info("ID_VIAG gerado")

return df

```

3.8 Função relativa à entrevista

Função/Variável	Status
passo_cd_entre	OK

Obs: o passo_cd_entre deve ser executado após o passo_tot_vitag.

```
In [ ]: log_tela.info('Definindo função relativa à entrevista')
log_output.info('\n\n=====\\n')

def passo_cd_entre(passo, df):
    """
    -----
    Substituir valores da coluna "CD_ENTRE"
    Todas entrevistas são consideradas "completas", segundo informações do Metrô

    * sem viagem: se TOT_VIAG == 0
    * com viagem: se TOT_VIAG != 0

    # #####Categorias novas
    # | Valor | Descrição |
    # | ----- | ----- |
    # | 0 | Completa sem viagem |
    # | 1 | Completa com viagem |

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
    log_tela.info("### PASSO " + str(passo) + " - CD_ENTRE")
    log_tela.warning('\n\n#####\\n' +
                     'Este passo DEVE ser executado apenas após a \\n' +
                     'execução do passo que geram o TOT_VIAG.\\n' +
                     '#####\\n' )

    # Definindo 'CD_ENTRE' baseado no valor de 'TOT_VIAG'
    df.loc[df['TOT_VIAG'] == 0, 'CD_ENTRE'] = 0
    df.loc[df['TOT_VIAG'] != 0, 'CD_ENTRE'] = 1

    verifica_dummy(df, 'CD_ENTRE')
    log_output.info('\n\n=====\\n')

    return df

In [ ]: def passo_correcoes(passo, df):
    log_tela.info("### PASSO " + str(passo) + " - Correções")

    log_tela.info("Corrigindo 'SERV_PAS' caso TOT_VIAG seja nulo.")
    df.loc[df['TOT_VIAG'] == 0, 'SERV_PAS_ORIG'] = 0
```

```
df.loc[df['TOT_VIAG'] == 0, 'SERV_PAS_DEST'] = 0

log_tela.info("Corrigindo motivo_origem, provável domicílio")
df.loc[(df['MOTIVO_ORIG']==0) &
       (df['MOTIVO_DEST']!=0) &
       (df['ZONA_ORIG']==df['ZONA_DOM']) &
       (df['SUBZONA_ORIG']==df['SUBZONA_DOM']), 'MOTIVO_ORIG'] == 8

log_tela.info("Corrigindo motivo_destino, provável domicílio")
df.loc[(df['MOTIVO_ORIG']!=0) &
       (df['MOTIVO_DEST']==0) &
       (df['ZONA_DEST']==df['ZONA_DOM']) &
       (df['SUBZONA_DEST']==df['SUBZONA_DOM']), 'MOTIVO_DEST'] == 8

return df
```

4 Defining the main function

This function will load all needed files, then call all other step functions defined above. On the end this main function will save the transformed data into an csv file.

```
In [ ]: def main():
    start_time = time.time()

    log_tela.info("Horário de início da execução: %s" %
                  time.strftime("%H:%M", time.localtime(start_time)))

    # ----

    log_tela.info('Lendo arquivos CSV')

    log_output.info('Lendo CSV da base original da OD.')
    od = pd.read_csv('bases/OD_1977.csv', sep=';', decimal=',')

    # Definindo variável de deflação de renda:
    deflator = 0.422345901

    # Filtrando dataframe para pegar apenas uma amostra
    # logging.info('\nFiltering the main dataframe to get just a sample')
    #od = od[:1000].copy()

    log_output.info('\n\n=====\\n')
    log_tela.info('Pré-Processamento.')
    od = pre_processamento(od)

    log_tela.info('Imprimindo a lista de variáveis do dataframe da OD.')
    log_tela.debug(od.columns.tolist())

    #Contador de 'PASSO'
    passo = 1

    # ----
    # ##Passo: "ANO"
    od = passo_ano(passo, od)
    passo += 1

    # ----
    # ##Passo: "DIA_SEM"
    od = passo_dia_sem(passo, od)
    passo += 1

    # ----
    # ##Passo: "UCODs"
    od = passo_ucods(passo, od)
    passo += 1

    # ----
    # ##Passo: "ZONA_DOM"
    od = passo_zona_dom(passo, od)
    passo += 1
```

```

# -----
# ##Passo: "SUBZONA_DOM"
od = passo_subzona_dom(passo, od)
passo += 1

# -----
# ##Passo: "MUN_DOM"
od = passo_mun_dom(passo, od)
passo += 1

# -----
# ##Passo: "F_DOM"
od = passo_f_dom(passo, od)
passo += 1

# -----
# ##Passo: "FE_DOM"
od = passo_fe_dom(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_DOM"
od = passo_tipo_dom(passo, od)
passo += 1

# -----
# ##Passo: "TOT_FAM"
od = passo_tot_fam(passo, od)
passo += 1

# -----
# ##Passo: "F_FAM"
od = passo_f_fam(passo, od)
passo += 1

# -----
# ##Passo: "FE_FAM"
od = passo_fe_fam(passo, od)
passo += 1

# -----
# ##Passo: "COND_MORA"
od = passo_cond_mora(passo, od)
passo += 1

# -----
# ##Passo: "QT_AUTO"
od = passo_qt_auto(passo, od)
passo += 1

# -----
# ##Passo: "QT_BICI"
od = passo_qt_bici(passo, od)
passo += 1

```

```

# -----
# ##Passo: "QT_MOTO"
od = passo_qt_moto(passo, od)
passo += 1

# -----
# ##PASSO: "REN_FAM"
od = passo_ren_fam(passo, od, deflator)
passo += 1

# -----
# ##Passo: "CD_RENFAM"
od = passo_cd_renfam(passo, od)
passo += 1

# -----
# ##Passo: "F_PESS"
od = passo_f_pess(passo, od)
passo += 1

# -----
# ##Passo: "FE_PESS"
od = passo_fe_pess(passo, od)
passo += 1

# -----
# ##Passo: "SIT_FAM"
od = passo_sit_fam(passo, od)
passo += 1

# -----
# ##Passo: "IDADE"
od = passo_idade(passo, od)
passo += 1

# -----
# ##Passo: "SEXO"
od = passo_sexo(passo, od)
passo += 1

# -----
# ##Passo: "GRAU_INSTR"
od = passo_grau_instr(passo, od)
passo += 1

# -----
# ##Passo: "OCUP"
od = passo_ocup(passo, od)
passo += 1

# -----
# ##Passo: "SETOR_ATIV"
od = passo_setor_ativ(passo, od)

```

```

passo += 1

# -----
# ##Passo: "REN_IND"
od = passo_ren_ind(passo, od, deflator)
passo += 1

# -----
# ##Passo: "CD_RENIND"
od = passo_cd_renind(passo, od)
passo += 1

## O Passo estuda encontra-se mais abaixo, após MUN_ESC

# -----
# ##Passo: "F_VIAG"
od = passo_f_viag(passo, od)
passo += 1

# -----
# ##Passo: "FE_VIAG"
od = passo_fe_viag(passo, od)
passo += 1

# -----
# ##Passo: "ZONA_ESC"
od = passo_zona_esc(passo, od)
passo += 1

# -----
# ##Passo: "SUBZONA_ESC"
od = passo_subzona_esc(passo, od)
passo += 1

# -----
# ##Passo: "MUN_ESC"
od = passo_mun_esc(passo, od)
passo += 1

# -----
# ##Passo: "ESTUDA"
# Este passo deve vir após os passos de localização da escola
od = passo_estuda(passo, od)
passo += 1

# -----
# ##Passo: "ZONA_TRAB1"
od = passo_zona_trab1(passo, od)
passo += 1

# -----
# ##Passo: "SUBZONA_TRAB1"
od = passo_subzona_trab1(passo, od)
passo += 1

```

```

# -----
# ##Passo: "MUN_TRAB1"
od = passo_mun_trab1(passo, od)
passo += 1

# -----
# ##Passo: "ZONA_TRAB2"
od = passo_zona_trab2(passo, od)
passo += 1

# -----
# ##Passo: "SUBZONA_TRAB2"
od = passo_subzona_trab2(passo, od)
passo += 1

# -----
# ##Passo: "MUN_TRAB2"
od = passo_mun_trab2(passo, od)
passo += 1

# -----
# ##Passo: "ZONA_ORIG"
od = passo_zona_orig(passo, od)
passo += 1

# -----
# ##Passo: "SUBZONA_ORIG"
od = passo_subzona_orig(passo, od)
passo += 1

# -----
# ##Passo: "MUN_ORIG"
od = passo_mun_orig(passo, od)
passo += 1

# -----
# ##Passo: "ZONA_DEST"
od = passo_zona_dest(passo, od)
passo += 1

# -----
# ##Passo: "SUBZONA_DEST"
od = passo_subzona_dest(passo, od)
passo += 1

# -----
# ##Passo: "MUN_DEST"
od = passo_mun_dest(passo, od)
passo += 1

# -----
# ##Passo: "SERV_PAS_ORIG"
od = passo_serv_pas_orig(passo, od)

```

```

passo += 1

# -----
# ##Passo: "SERV_PAS_DEST"
od = passo_serv_pas_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_ORIG"
od = passo_motivo_orig(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_DEST"
od = passo_motivo_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOD01"
od = passo_modo1(passo, od)
passo += 1

# -----
# ##Passo: "MOD02"
od = passo_modo2(passo, od)
passo += 1

# -----
# ##Passo: "MOD03"
od = passo_modo3(passo, od)
passo += 1

# -----
# ##Passo: "MOD04"
od = passo_modo4(passo, od)
passo += 1

# -----
# ##Passo: "MODO_PRIN"
od = passo_modo_prin(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_VIAG"
od = passo_tipo_viag(passo, od)
passo += 1

# -----
# ##"H_SAIDA"; "MIN_SAIDA"; "ANDA_ORIG"; "H_CHEG"; "MIN_CHEG";
#     "ANDA_DEST"; "DIST_VIAG" e "DURACAO"
# Nada há que se fazer em relação aos dados das colunas acima mencionadas

# -----
# ##Passo: "TIPO_EST_AUTO"

```

```

od = passo_tipo_est_auto(passo, od)
passo += 1

# -----
# ##Passo: "VALOR_EST_AUTO"
od = passo_valor_est_auto(passo, od, deflator)
passo += 1

# -----
# ##Passo: Coordenadas
od = coordenadas(passo, od)
passo += 1

## O passo TOT_VIAG apenas é chamado após a geração dos IDs.

# -----
# ##Passo: Gerando NO's e ID's
od = gera_nos_e_ids(passo, od)
passo += 1

# -----
# ##Passo: "TOT_VIAG"
od = passo_tot_viag(passo, od)
passo += 1

# -----
# ##Passo: "CD_ENTRE"
od = passo_cd_entre(passo, od)
passo += 1

# -----
# ##Passo: Correções
od = passo_correcoes(passo, od)
passo += 1

log_tela.info('Salvando dataframe como arquivo CSV')
# -----
# ## Salvando o dataframe num arquivo local
od.to_csv('outputs/1977_od.csv', sep=';', decimal=',', index=False)

log_tela.info("Output gerado. Arquivo: outputs/1977_od.csv")

log_tela.info("Tempo total de execução: %s segundos" %
              (time.time() - start_time))

log_tela.info("Horário de finalização: %s" %
              (time.strftime("%H:%M", time.localtime(time.time()))))

log_tela.info("Terminou o main")

```

5 RUN the main() function....

```
In [ ]: if __name__ == "__main__":
          main()
```

rotina_final_1987

January 4, 2016

1 Setup Inicial

```
In [ ]: # coding: utf-8
        import math
        import logging
        import time
        import pandas as pd

        # Faz os gráficos um pouco mais bonitos
        pd.set_option('display.mpl_style', 'default')
```

2 Definindo Loggers

Define os loggers

Estes ‘loggers’ serão utilizados para salvar as saídas (outputs) em um arquivo de texto no diretório ‘outputs’.

ref: <http://stackoverflow.com/questions/17035077/python-logging-to-multiple-log-files-from-different-classes>

ATENÇÃO: RODAR O BLOCO ABAIXO APENAS UMA VEZ, SE ESTE BLOCO FOR EXECUTADO MAIS DE UMA VEZ OS LOGs SERÃO DUPLICADOS.

```
In [ ]: log_formatter = logging.Formatter('%(asctime)s | %(levelname)s: %(message)s')

log_output = logging.getLogger('log_output')
FH_output = logging.FileHandler(
            'outputs/1987_output.log', mode='w')
FH_output.setFormatter(log_formatter)
log_output.setLevel(logging.INFO)
log_output.addHandler(FH_output)
log_output.propagate = False

# Este logger (log_tela) loga na tela e também
# no arquivo de output, junto com o conteúdo do
# logger log_output
log_tela = logging.getLogger('log_tela')
SH_tela = logging.StreamHandler()
SH_tela.setFormatter(log_formatter)
log_tela.addHandler(SH_tela)
log_tela.addHandler(FH_output)
log_tela.setLevel(logging.INFO)
log_tela.propagate = False
```

3 Funções de 1987

3.1 Funções gerais assessórias

Função	Status
verifica_dummy	OK
verifica_range	OK
pre_processamento	OK
coordenadas	OK

```
In [ ]: log_tela.info('Definindo as funções gerais assessórias')
log_output.info('\n'=====\\n')

def verifica_dummy(df, variavel):
    """
    Verifica se uma variável, dummy, contém algum valor diferente de 0 ou de 1.
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados.
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_dummy(dataframe, 'coluna a ser verificada')
    """
    contador_de_erros = 0
    log_tela.info('Verificando a variável Dummy: ' + variavel)

    df_erros = df[(df[variavel] != 0) & (df[variavel] != 1)]
    if len(df_erros[variavel].value_counts()) > 0:
        log_tela.warning(variavel + ": " +
                          str(len(df_erros[variavel].value_counts())) +
                          " erros encontrados:\\n" +
                          str(df_erros[variavel].value_counts()))
    else:
        log_tela.info(variavel + ": Nenhum erro encontrado.")

def verifica_range(df, variavel, valor_menor, valor_maior):
    """
    Verifica se uma variável, do tipo número inteiro, contém algum valor menor
    que "valor_menor" ou maior que "valor_maior".
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados
    :param valor_menor: Valor mínimo esperado na variável (int ou float)
    :param valor_maior: Valor máximo esperado na variável (int ou float)
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_range(dataframe, 'nome_variavel', valor_menor, valor_maior)
    """
    log_tela.info('Verificando Range da variável: ' + variavel)
```

```

# Obs: Registros inválidos: None (equivalente a NA)
nulos = df[variavel].isnull().sum()
nulos = nulos if nulos else 0
log_output.info('\n\n' +
    ',       - ' + 'Mínimo esperado: ' + str(valor_menor) + '\n' +
    ',       - ' + 'Máximo esperado: ' + str(valor_maior) + '\n' +
    ',       - ' + 'Total de registros: ' + str(len(df[variavel])) +
    '\n' +
    ',       - ' + 'Registros nulos (NA): ' +
    str(df[variavel].isnull().sum()) + '\n'
)
df_errores = df[(df[variavel] < valor_menor) | (df[variavel] > valor_maior)]

if len(df_errores[variavel].value_counts()) > 0:

    result = df_errores[variavel].value_counts().sort_index()
    # Verificando limite inferior
    if result.first_valid_index() < valor_menor:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro mínimo encontrado: ' +
            str(result.first_valid_index()) + ' - abaixo do esperado!')
    # Verificando limite superior
    if result.last_valid_index() > valor_maior:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro máximo encontrado: ' +
            str(result.last_valid_index()) + " - acima do esperado!")

    log_tela.warning(variavel + ': ' +
        str(len(df_errores[variavel].value_counts())) +
        ' valor(es) incorreto(s) ' +
        'encontrado(s) nesta variável:\n' +
        str(df_errores[variavel].value_counts()))
else:
    log_tela.info(variavel + ": Nenhum erro encontrado.")

def pre_processamento(df):
    """
    Realiza algumas ações prévias ao processamento dos dados,
    removendo alguns registros e ajustando o dataframe.
    """
    log_tela.info('Criando/Renomeando colunas no dataframe principal')

    log_output.info('Renomeando coluna UCOD para UCOD_DOM')
    df.rename(columns={'UCOD': 'UCOD_DOM'}, inplace=True)

    log_output.info('Renomeando coluna ANDA_CHEG para ANDA_DEST')
    df.rename(columns={'ANDA_CHEG': 'ANDA_DEST'}, inplace=True)

    log_tela.info('Verificando se todas as variáveis esperadas'+
        'existem no dataframe.\n' +
        'Caso não exista alguma, ela é criada.')

```

```

variaveis = ['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
             'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
             'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
             'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
             'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
             'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
             'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
             'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
             'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
             'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
             'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
             'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
             'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
             'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
             'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
             'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
             'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
             'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
             'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
             'VALOR_EST_AUTO']

for variavel in variaveis:
    if variavel not in df.columns:
        # Se a variável não existe no dataframe então ela é criada
        # com valor padrão de NONE (NA)
        df[variavel] = None
    log_tela.info('Criando a variavel ' + variavel + ' no dataframe')

log_output.info('Reordenando as variáveis')
df = df[['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
          'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
          'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
          'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
          'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
          'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
          'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
          'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
          'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
          'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
          'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
          'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
          'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
          'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
          'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
          'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
          'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
          'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
          'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
          'VALOR_EST_AUTO']]

log_output.info('\n\n=====\\n')
log_output.info('\\n=====\\n')

return df

```

```

def coordenadas(passo, df):
    """
    Itera sobre os registros do dataframe coords,
    a cada iteração, utiliza o valor das colunas ZONA e SUBZONA
    para filtrar o dataframe df em cada um dos tipos de ZONA E
    SUBZONA (DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui os valores
    das coordenadas X e Y de cada tipo (CO_DOM_X, CO_ESC_Y, etc) com
    os valores das colunas CO_X e CO_Y do dataframe coords.

    :param passo: número do passo para o log
    :param df: Dafaframe a ser modificado (ex.: od1987)
    :return: retorna o dataframe modificado com todas as coordenadas aplicadas
    """
    log_tela.info("### PASSO " + str(passo) + " - COORDENADAS")

    def coord_aux(row):
        """
        Esta função irá receber uma linha (row) do dataframe coords e
        utilizar os valores desta linha para realizar as alterações
        de coordenadas no dataframe df.
        """
        # Começando pelo tipo DOM, alterando CO_DOM_X e depois CO_DOM_Y
        df.loc[(df['ZONA_DOM'] == row['ZONA']) &
               (df['SUBZONA_DOM'] == row['SUBZONA']), 'CO_DOM_X'] = row['CO_X']
        df.loc[(df['ZONA_DOM'] == row['ZONA']) &
               (df['SUBZONA_DOM'] == row['SUBZONA']), 'CO_DOM_Y'] = row['CO_Y']
        # Agora com o tipo ESC, alterando CO_ESC_X e depois CO_ESC_Y
        df.loc[(df['ZONA_ESC'] == row['ZONA']) &
               (df['SUBZONA_ESC'] == row['SUBZONA']), 'CO_ESC_X'] = row['CO_X']
        df.loc[(df['ZONA_ESC'] == row['ZONA']) &
               (df['SUBZONA_ESC'] == row['SUBZONA']), 'CO_ESC_Y'] = row['CO_Y']
        # Agora com o tipo TRAB1, alterando CO_TRAB1_X e depois CO_TRAB1_Y
        df.loc[(df['ZONA_TRAB1'] == row['ZONA']) &
               (df['SUBZONA_TRAB1'] == row['SUBZONA']), 'CO_TRAB1_X'] = row['CO_X']
        df.loc[(df['ZONA_TRAB1'] == row['ZONA']) &
               (df['SUBZONA_TRAB1'] == row['SUBZONA']), 'CO_TRAB1_Y'] = row['CO_Y']
        # Agora com o tipo TRAB2, alterando CO_TRAB2_X e depois CO_TRAB2_Y
        df.loc[(df['ZONA_TRAB2'] == row['ZONA']) &
               (df['SUBZONA_TRAB2'] == row['SUBZONA']), 'CO_TRAB2_X'] = row['CO_X']
        df.loc[(df['ZONA_TRAB2'] == row['ZONA']) &
               (df['SUBZONA_TRAB2'] == row['SUBZONA']), 'CO_TRAB2_Y'] = row['CO_Y']
        # Agora com o tipo ORIG, alterando CO_ORIG_X e depois CO_ORIG_Y
        df.loc[(df['ZONA_ORIG'] == row['ZONA']) &
               (df['SUBZONA_ORIG'] == row['SUBZONA']), 'CO_ORIG_X'] = row['CO_X']
        df.loc[(df['ZONA_ORIG'] == row['ZONA']) &
               (df['SUBZONA_ORIG'] == row['SUBZONA']), 'CO_ORIG_Y'] = row['CO_Y']
        # Agora com o tipo DEST, alterando CO_DEST_X e depois CO_DEST_Y
        df.loc[(df['ZONA_DEST'] == row['ZONA']) &
               (df['SUBZONA_DEST'] == row['SUBZONA']), 'CO_DEST_X'] = row['CO_X']
        df.loc[(df['ZONA_DEST'] == row['ZONA']) &
               (df['SUBZONA_DEST'] == row['SUBZONA']), 'CO_DEST_Y'] = row['CO_Y']

```

```
log_output.info('Lendo arquivo auxiliar de Coordenadas das subzonas')
coords = pd.read_csv('auxiliares/coord_subzonas_1987.csv', sep=';', decimal=',')

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = coords.apply(coord_aux, axis=1)

log_output.info('\n\n=====\\n')

return df
```

3.2 Funções Gerais

Função/Variável	Status
passo_ano	OK
passo_dia_sem	OK
passo_ucods	OK

```
In [ ]: log_tela.info('Definindo as funções gerais')
log_output.info('\n'=====\\n')

def passo_ano(passo, df):
    """
    Preenche a coluna "ANO" com valor 2 em todas células
    Categorias:
    /valor/ano_correspondente/
    /-----/-----
    /1/1977/
    /2/1987/
    /3/1997/
    /4/2007/

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
    :return: retorna o dataframe modificado
    """

    log_tela.info("### PASSO " + str(passo) + " - ANO")

    # Definindo valor '2' para todas as células da coluna ANO
    df["ANO"] = 2

    return df

def passo_dia_sem(passo, df):
    """
    Assumindo que as respostas iguais a **1** são
    referentes à segunda-feira (2) e
    valores iguais a **7** são referentes à
    sexta-feira (6)

    # #####Categorias:
    # Valor/Descrição
    # -----/-----
    # 2/Segunda-Feira
    # 3/Terça-Feira
    # 4/Quarta-Feira
    # 5/Quinta-Feira
    # 6/Sexta-Feira
```

```

:param passo: Número do passo atual para registro/log
:param df: dataframe a ser modificado
:return: retorna o dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - DIA_SEM")

# Substituindo **0** por **None**
df.loc[df['DIA_SEM']==0,'DIA_SEM'] = None

# Alocando que respondeu "1" para Segunda-feira (2)
df.loc[df['DIA_SEM']==1,'DIA_SEM'] = 2

# Alocando quem respondeu "7" para Sexta-feira (6)
df.loc[df['DIA_SEM']==7,'DIA_SEM'] = 6

# Verificando intervalo de valores - condições:
# "DIA_SEM >= 2" E "DIA_SEM <= 6"
verifica_range(df, 'DIA_SEM', 2, 6)
log_output.info('\n\n=====\\n')

return df

def passo_ucods(passo, df):
"""
Itera sobre os registros do dataframe ucods,
a cada iteração, utiliza o valor da coluna ZONA_REF
para filtrar o dataframe df em cada um dos tipos de ZONA
(DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui o valor
da respectiva UCOD.

:param passo: número do passo para o log
:param df: Dafaframe a ser modificado (ex.: od1987)
:return: retorna o dataframe modificado com todas as UCODS aplicadas
"""
log_tela.info("### PASSO " + str(passo) + " - UCODS")

def ucod_aux(row):
    df.loc[df['ZONA_DOM']==row['ZONA_REF'], 'UCOD_DOM'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_ESC']==row['ZONA_REF'], 'UCOD_ESC'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_TRAB1']==row['ZONA_REF'], 'UCOD_TRAB1'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_TRAB2']==row['ZONA_REF'], 'UCOD_TRAB2'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_ORIG']==row['ZONA_REF'], 'UCOD_ORIG'] = row['UCOD_BUSCADA']
    df.loc[df['ZONA_DEST']==row['ZONA_REF'], 'UCOD_DEST'] = row['UCOD_BUSCADA']

log_output.info('Lendo arquivo auxiliar UCOD')
ucods = pd.read_csv('auxiliares/UCOD-1987.csv', sep=';', decimal=',')

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = ucods.apply(ucod_aux, axis=1)

# Verificando intervalo de valores - condições:
# "UCOD_XXX >= 1" E "UCOD_XXX <= 67"

```

```
for tipo in ['DOM', 'ESC', 'TRAB1', 'TRAB2', 'ORIG', 'DEST']:
    verifica_range(df, 'UCOD_' + tipo, 1, 67)
log_output.info('\n\n=====\\n')

return df
```

3.3 Funções do Domicílio

Função/Variável	Status
passo_zona_dom	OK
passo_subzona_dom	OK
passo_mun_dom	OK
passo_f_dom	OK
passo_fe_dom	OK
passo_tipo_dom	OK

```
In [ ]: log_tela.info('Definindo as funções do domicílio')
log_output.info('\n\n=====\\n')

def passo_zona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 254

    [Teste: Checar se existe algum número < 1 ou > 254.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_DOM")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_DOM']==0,'ZONA_DOM'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_DOM >= 1" E "ZONA_DOM <= 254"
    verifica_range(df, 'ZONA_DOM', 1, 254)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
```

```

:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DOM")

# Substituindo valor 0 por None
df.loc[df['SUBZONA_DOM']==0,'SUBZONA_DOM'] = None

# Verificando intervalo de valores - condições:
# "SUBZONA_DOM >= 1" E "SUBZONA_DOM <= 9"
verifica_range(df, 'SUBZONA_DOM', 1, 9)
log_output.info('\n\n=====\\n')

return df


def passo_mun_dom(passo, df):
"""
Checar se existe algum erro

# #####Categorias
# > 1 a 38

[Teste: Checar se existe algum número < 1 ou > 38.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MUN_DOM")

# Substituindo valor 0 por None
df.loc[df['MUN_DOM']==0,'MUN_DOM'] = None

# Verificando intervalo de valores - condições:
# "MUN_DOM >= 1" E "MUN_DOM <= 38"
verifica_range(df, 'MUN_DOM', 1, 38)
log_output.info('\n\n=====\\n')

return df


def passo_f_dom(passo, df):
"""
Checar se existe algum erro na coluna "F_DOM"

# #####Categorias
# Valor/Descrição
# ----/----
# 0/Demais registros
# 1/Primeiro Registro do Domicílio

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]

```

```

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - F_DOM")

verifica_dummy(df, 'F_DOM')
log_output.info('\n\n=====\\n')

return df

def passo_fe_dom(passo, df):
"""
Nada há que se fazer em relação aos dados da coluna "FE_DOM"

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - FE_DOM")
log_output.info('\n\n=====\\n')

return df

def passo_tipo_dom(passo, df):
"""
Substituir **0** por **None (NA)**
Substituir **2** por **0**

# #####Categorias anteriores
# Valor / Descrição
# ----/----
# 1/Individual
# 2/Coletivo

# #####Categorias novas
# Valor / Descrição
# ----/----
# 0/Coletivo
# 1/Particular

[Teste: Checar se existe algum número < 0 ou > 1.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
log_tela.info("### PASSO " + str(passo) + " - TIPO_DOM")

df.loc[df['TIPO_DOM']==0, 'TIPO_DOM'] = None
df.loc[df['TIPO_DOM']==2, 'TIPO_DOM'] = 0

```

```
# Verificando intervalo de valores - condições:  
# "TIPO_DOM >= 0" E "TIPO_DOM <= 1"  
verifica_dummy(df, 'TIPO_DOM')  
log_output.info('\\n\\n=====\\n')  
  
return df
```

3.4 Funções da Família

Função/Variável	Status
passo_tot_fam	OK
passo_f_fam	OK
passo_fe_fam	OK
passo_cond_mora	OK
passo_qt_auto	OK
passo_qt_bici	OK
passo_qt_moto	OK
passo_ren_fam	OK
passo_cd_renfam	OK

```
In [ ]: log_tela.info('Definindo as funções da família')
log_output.info('\n\n=====\\n')

def passo_tot_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "TOT_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - TOT_FAM")
    log_output.info('\\n\\n=====\\n')

    return df

def passo_f_fam(passo, df):
    """
    Checar se existe algum erro na coluna "F_FAM"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Família

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_FAM")

    verifica_dummy(df, 'F_FAM')
```

```

log_output.info('\n\n=====\\n')
return df

def passo_fe_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_FAM")
    log_output.info('\n\n=====\\n')

    return df

def passo_cond_mora(passo, df):
    """
    Substituir valores da coluna "COND_MORA"

    * Substituir todos valores **2** por **0**
    * Substituir todos valores **0** por **None**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **3** por **5**
    * Substituir todos valores **1** por **3**
    * Substituir todos valores **5** por **1**

    ##### Categorias anteriores

    Valor/Descrição
    ----/----
    1/Não se aplica
    2/Não respondeu
    3/Alugada
    4/Casa Própria

    ##### Categorias novas

    Valor/Descrição
    ----/----
    1/Alugada
    2/Própria
    3/Outros

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - COND_MORA")

```

```

# Substituindo valor 2 por 0
df.loc[df['COND_MORA']==2,'COND_MORA'] = 0
# Substituindo valor 0 por None
df.loc[df['COND_MORA']==0,'COND_MORA'] = None
# Substituindo valor 4 por 2
df.loc[df['COND_MORA']==4,'COND_MORA'] = 2
# Substituindo valor 3 por 5
df.loc[df['COND_MORA']==3,'COND_MORA'] = 5
# Substituindo valor 1 por 3
df.loc[df['COND_MORA']==1,'COND_MORA'] = 3
# Substituindo valor 5 por 1
df.loc[df['COND_MORA']==5,'COND_MORA'] = 1

# Verificando intervalo de valores - condições:
# "COND_MORA >= 1" E "COND_MORA <= 3"
verifica_range(df, 'COND_MORA', 1, 3)
log_output.info('\'\n\'=====\\n')

return df

def passo_qt_auto(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "QT_AUTO"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - QT_AUTO")
    log_output.info('\'\n\'=====\\n')

    return df

def passo_qt_bici(passo, df):
    """
    Não existe essa informação no banco de dados de 1987, logo,
    este campo será preenchido com 'None'.
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    df['QT_BICI'] = None

    log_tela.info('### PASSO ' + str(passo) + ' - QT_BICI')
    log_output.info('\'\n\'=====\\n')

    return df

def passo_qt_moto(passo, df):
    """
    Não existe essa informação no banco de dados de 1987, logo,
    este campo será preenchido com 'None'.
    """

```

```

:param passo: Número do passo atual para registro/log
:param df:
:return: dataframe modificado
"""
df['QT_MOTO'] = None

log_tela.info('### PASSO ' + str(passo) + ' - QT_MOTO')
log_output.info('\n\n=====\\n')

return df

def passo_ren_fam(passo, df, deflator):
    """
    Corrigir a renda familiar de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_FAM")

    df['REN_FAM'] = df['REN_FAM'] * deflator

    return df

def passo_cd_renfam(passo, df):
    """
    Substituir valores da coluna "CD_RENFAM"

    * Substituir todos valores **1** por **0**
    * Substituir todos valores **3** por **1**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Não Tem Renda
    2/Renda Familiar Incompleta
    3/Renda Familiar Completa

    ##### Categorias novas
    Valor/Descrição
    ----/----
    0/Renda Familiar Declarada como Zero
    1/Renda Familiar Declarada e Maior que Zero
    2/Renda Atribuída

    [Teste: Checar se existe algum número < 0 ou > 2.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe corrigido
    """

```

```

log_tela.info("### PASSO " + str(passo) + " - CD_RENFAM")

# Substituindo valor 1 por 0
df.loc[df['CD_RENFAM']==1,'CD_RENFAM'] = 0
# Substituindo valor 3 por 1
df.loc[df['CD_RENFAM']==3,'CD_RENFAM'] = 1

# Dividindo a categoria '0', "Respondeu", em:
# - 0 "Renda Familiar Declarada como Zero" e
# - 1 "Renda Familiar Declarada e Maior que Zero"
df.loc[(df['CD_RENFAM'] == 0) &
       (df['REN_FAM'] != 0) &
       (df['REN_FAM'].notnull()), 'CD_RENFAM'] = 1

# Verificando intervalo de valores - condições:
# "CD_RENFAM >= 0" E "CD_RENFAM <= 2"
verifica_range(df, 'CD_RENFAM', 0, 2)
log_output.info('\'\n\'=====\\n')

return df

```

3.5 Funções da pessoa

Função/Variável	Status
passo_f_pess	OK
passo_fe_pess	OK
passo_sit_fam	OK
passo_idade	OK
passo_sexo	OK
passo_grau_instr	OK
passo_ocup	OK
passo_setor_ativ	OK
passo_ren_ind	OK
passo_cd_renind	OK
passo_estuda	OK

Obs.: o passo estuda deve ser executado após o passo ZONA_ESC.

```
In [ ]: log_tela.info('Definindo as funções da pessoa')
log_output.info('\n\n=====\\n')

def passo_f_pess(passo, df):
    """
    Checar se existe algum erro na coluna "F_PESS"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Pessoa

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_PESS")

    verifica_dummy(df, 'F_PESS')
    log_output.info('\n\n=====\\n')

    return df

def passo_fe_pess(passo, df):
    """
    Nada há que se fazer em relação aos dados das colunas "FE_PESS"
```

```

:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
log_tela.info("### PASSO " + str(passo) + " - FE_PESS")
log_output.info('\n\n=====\\n')

return df

def passo_sit_fam(passo, df):
    """
    * Substituir todos valores **5** por **4**
    * Substituir todos valores **6** por **5**
    * Substituir todos valores **7** por **6**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Chefe
    2/Cônjuge
    3/Filho(a)
    4/Parente
    5/Agregado
    6/Empregado Residente
    7/Visitante

    ##### Categorias novas:
    Valor/Descrição
    ----/----
    0/ Não Respondeu/Não fez viagem
    1/ Pessoa Responsável
    2/ Cônjuge/Companheiro(a)
    3/ Filho(a)/Enteado(a)
    4/ Outro Parente / Agregado
    5/ Empregado Residente
    6/ Outros (visitante não residente / parente do empregado)

[Teste: Checar se existe algum número < 0 ou > 6.
Se encontrar, retornar erro indicando em qual linha.]]

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - SIT_FAM")

# Substituindo valor 5 por 4
df.loc[df['SIT_FAM']==5,'SIT_FAM'] = 4
# Substituindo valor 6 por 5
df.loc[df['SIT_FAM']==6,'SIT_FAM'] = 5
# Substituindo valor 7 por 6
df.loc[df['SIT_FAM']==7,'SIT_FAM'] = 6

# Verificando intervalo de valores - condições:

```

```

        # "SIT_FAM >= 0" E "SIT_FAM <= 6"
verifica_range(df, 'SIT_FAM', 0, 6)
log_output.info('\n\n=====\\n')

return df

def passo_idade(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "IDADE"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - IDADE")
    log_output.info('\n\n=====\\n')

return df

def passo_sexo(passo, df):
    """
    #####Categorias anteriores
    # Valor/Descrição
    # ---/---
    # 0/Não Respondeu (-> None)
    # 1/Masculino
    # 2/Feminino

    #####Categorias novas
    # Valor/Descrição
    # ---/---
    # 0/Masculino
    # 1/Feminino

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SEXO")

    # Substituindo valor 0 por None
    df.loc[df['SEXO']==0,'SEXO'] = None
    # Substituindo valor 1 por 0
    df.loc[df['SEXO']==1,'SEXO'] = 0
    # Substituindo valor 2 por 1
    df.loc[df['SEXO']==2,'SEXO'] = 1

    # Verificando intervalo de valores - condições:
    # "SEXO >= 0" E "SEXO <= 1"
    verifica_dummy(df, 'SEXO')
    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_grau_instr(passo, df):
    """
    Substituir valores da coluna "GRAU_INSTR"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **3**
    * Substituir todos valores **5** por **4**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores:
    Valor/Descrição
    ----/----
    0/Não declarou
    1/Não-alfabetizado/4ª série Incompleta
    2/4ª Série Completa
    3/1º Grau completo
    4/Colegial completo
    5/Superior Completo

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Não-Alfabetizado/Fundamental Incompleto
    2/Fundamental Completo/Médio Incompleto
    3/Médio Completo/Superior Incompleto
    4/Superior completo

    [Teste: Checar se existe algum número < 1 ou > 4.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - GRAU_INSTR")

    # Substituindo valor 2 por 1
    df.loc[df['GRAU_INSTR']==2, 'GRAU_INSTR'] = 1
    # Substituindo valor 3 por 2
    df.loc[df['GRAU_INSTR']==3, 'GRAU_INSTR'] = 2
    # Substituindo valor 4 por 3
    df.loc[df['GRAU_INSTR']==4, 'GRAU_INSTR'] = 3
    # Substituindo valor 5 por 4
    df.loc[df['GRAU_INSTR']==5, 'GRAU_INSTR'] = 4
    # Substituindo valor 0 por None
    df.loc[df['GRAU_INSTR']==0, 'GRAU_INSTR'] = None

    # Verificando intervalo de valores - condições:
    # "GRAU_INSTR >= 1" E "GRAU_INSTR <= 4"
    verifica_range(df, 'GRAU_INSTR', 1, 4)

```

```

log_output.info('\n\n=====\\n')
return df

def passo_ocup(passo, df):
    """
    Substituir valores da coluna "OCUP"

    Somar 10 em todos valores.
    Substituir todos valores **10** por **None**
    Substituir todos valores **11** por **7**
    Substituir todos valores **12** por **6**
    Substituir todos valores **13** por **3**
    Substituir todos valores **14** por **5**
    Substituir todos valores **15** por **4**
    Substituir todos valores **16** por **2**
    Substituir todos valores *maiores do que 16* por **1**

    # #####Categorias anteriores:
    # Valor/Descrição
    # ----/----
    # 1/Estudante
    # 2/Prendas Domésticas
    # 3/Aposentado
    # 4/Sem Ocupação (nunca trabalhou)
    # 5/Desempregado
    # 6/Em licença
    # 7 em diante/diversas profissões

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Tem trabalho
    # 2/Em licença médica
    # 3/Aposentado / pensionista
    # 4/Desempregado
    # 5/Sem ocupação
    # 6/Dona de casa
    # 7/Estudante

    [Teste: Checar se existe algum número < 1 ou > 7.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - OCUP")

df['OCUP'] = df['OCUP'] + 10
# Substituindo valor 10 por None
df.loc[df['OCUP']==10,'OCUP'] = None
# Substituindo valor 11 por 7
df.loc[df['OCUP']==11,'OCUP'] = 7

```

```

# Substituindo valor 12 por 6
df.loc[df['OCUP']==12,'OCUP'] = 6
# Substituindo valor 13 por 3
df.loc[df['OCUP']==13,'OCUP'] = 3
# Substituindo valor 14 por 5
df.loc[df['OCUP']==14,'OCUP'] = 5
# Substituindo valor 15 por 4
df.loc[df['OCUP']==15,'OCUP'] = 4
# Substituindo valor 16 por 2
df.loc[df['OCUP']==16,'OCUP'] = 2
# Substituindo valor >16 por 1
df.loc[df['OCUP']>16,'OCUP'] = 1

# Verificando intervalo de valores - condições:
# "OCUP >= 1" E "OCUP <= 7"
verifica_range(df, 'OCUP', 1, 7)
log_output.info(' \n\n===== \n')

return df


def passo_setor_ativ(passo, df):
    """
    Substituir valores da coluna "SETOR_ATIV"

    Na coluna "SETOR_ATIV", linha i,
        ler o valor da linha i da coluna "SETOR_ATIV", daí,
        buscar o mesmo valor na coluna "COD" do arquivo setor_ativ-1987.csv.
    Ao achar, retornar o valor da mesma linha, só que da coluna "COD_UNIF"

    #####Categorias anteriores
    > ver arquivo .csv

    #####Categorias novas
    Valor/Descrição
    ----/-----
    1/Agrícola
    2/Construção Civil
    3/Indústria
    4/Comércio
    5/Administração Pública
    6/Serviços de Transporte
    7/Outros serviços
    8/Outros
    9/Não se aplica

    [Teste: Checar se existe algum número < 0 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SETOR_ATIV")

```

```

log_output.info('Lendo arquivo de referência externa setor_ativ-1987.csv')
df_setor = pd.read_csv('auxiliares/setor_ativ-1987.csv', sep=';', decimal=',')

def setor_aux(row):
    df.loc[df['SETOR_ATIV']==row['COD'], 'SETOR_ATIV'] = row['COD_UNIF']

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = df_setor.apply(setor_aux, axis=1)

# Substituindo valor 0 por None
df.loc[df['SETOR_ATIV']==0, 'SETOR_ATIV'] = None

# Verificando intervalo de valores - condições:
# "SETOR_ATIV >= 0" E "SETOR_ATIV <= 9"
verifica_range(df, 'SETOR_ATIV', 0, 9)
log_output.info('\n\n=====\\n')

return df

def passo_ren_ind(passo, df, deflator):
    """
    Corriga a renda individual de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_IND")
    df['REN_IND'] = df['REN_IND'] * deflator
    return df

def passo_cd_renind(passo, df):
    """
    Substituir valores da coluna "CD_RENIND"

    * Substituir todos valores **2** por None
    * Substituir todos valores **1** por **0**
    * Substituir todos valores **3** por **1**

    ##### Categorias anteriores:
    Valor/Descrição
    ----/----
    1/Não tem renda
    2/Não Declarou
    3/Declarou

    ##### Categorias novas
    Valor/Descrição
    ----/-----
    """

```

```

0      /Não tem renda
1      /Tem renda

[Teste: Checar se existe algum número < 0 ou > 1.
   Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```

log_tela.info("### PASSO " + str(passo) + " - CD_RENIND")

df.loc[df['CD_RENIND']==2,'CD_RENIND'] = None
df.loc[df['CD_RENIND']==1,'CD_RENIND'] = 0
df.loc[df['CD_RENIND']==3,'CD_RENIND'] = 1

# Verificando intervalo de valores - condições:
# "CD_RENIND >= 0" E "CD_RENIND <= 1"
verifica_dummy(df, 'CD_RENIND')
log_output.info('\n\n=====\\n')

return df
```

def passo_estuda(passo, df):

"""

Se zona da escola for zero (0) então não estuda (0), senão, estuda (1)

#####
ATENÇÃO
ESTA FUNÇÃO SÓ DEVE SER
EXECUTADA APÓS A GERAÇÃO#
DE TODOS OS ID'S, POIS
HÁ UMA DESCONFIANÇA
QUANTO À QUALIDADE
DESTE DADO
#####

* Substituir todos valores **2** por **0**

Categorias anteriores

Valor/Descrição

----/----

1/Sim

2/Não

Categorias novas

Valor/Descrição

----/----

0/Não estuda

1/Estuda

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]

:param passo: Número do passo atual para registro/log

```

:param df:
:return: retorna dataframe com campo ESCOLA resolvido
"""

log_tela.info("### PASSO " + str(passo) + " - ESTUDA")

# Substitui 2 por 0
df.loc[df['ESTUDA'] == 2, 'ESTUDA'] = 0

# Substituindo todos que declararam zona escola diferente de zero
# com campo ESTUDA igual a 1.
df.loc[(df['ZONA_ESC'] != 0)&
       (df['ZONA_ESC'].notnull()), 'ESTUDA'] = 1

verifica_dummy(df, 'ESTUDA')
log_output.info('\n\n=====\\n')

return df

```

3.6 Funções referentes da Viagem

Função/Variável	Status
passo_f_vitag	OK
passo_fe_vitag	OK
passo_zona_esc	OK
passo_subzona_esc	OK
passo_mun_esc	OK
passo_zona_trab1	OK
passo_subzona_trab1	OK
passo_mun_trab1	OK
passo_zona_trab2	OK
passo_subzona_trab2	OK
passo_mun_trab2	OK
passo_zona_orig	OK
passo_subzona_orig	OK
passo_mun_orig	OK
passo_zona_dest	OK
passo_subzona_dest	OK
passo_mun_dest	OK
passo_serv_pas_orig	OK
passo_serv_pas_dest	OK
passo_motivo_orig	OK
passo_motivo_dest	OK
passo_modo1	OK
passo_modo2	OK
passo_modo3	OK
passo_modo4	OK
passo_modo_prin	OK
passo_tipo_est_auto	OK
passo_valor_est_auto	OK
passo_tot_vitag	OK

Obs: O passo_tot_vitag só deve ser executado após a produção dos ID's e NO's.

```
In [ ]: log_tela.info('Definindo funções referentes às viagens')
log_output.info('\n'=====\\n')
```

```
def passo_f_vitag(passo, df):
```

```

"""
Checar se existe algum erro na coluna "F_VIAG"

# #####Categorias
# Valor/Descrição
# ----/----
# 0/Não fez viagem
# 1/Fez viagem

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - F_VIAG")

verifica_dummy(df, 'F_VIAG')
log_output.info('\n\n=====\\n')

return df


def passo_fe_viag(passo, df):
"""
# Nada há que se fazer em relação aos dados da coluna "FE_VIAG"
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - FE_VIAG")
log_output.info('\n\n=====\\n')

return df


def passo_zona_esc(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 254

[Teste: Checar se existe algum número < 1 ou > 254.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: Sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - ZONA_ESC")

# Substituindo valor 0 por None
df.loc[df['ZONA_ESC']==0, 'ZONA_ESC'] = None

```

```

# Verificando intervalo de valores - condições:
# "ZONA_ESC >= 1" E "ZONA_ESC <= 254"
verifica_range(df, 'ZONA_ESC', 1, 254)
log_output.info('\'\n\'=====\'\n')

return df


def passo_subzona_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ESC")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_ESC']==0,'SUBZONA_ESC'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_ESC >= 1" E "SUBZONA_ESC <= 9"
    verifica_range(df, 'SUBZONA_ESC', 1, 9)
    log_output.info('\'\n\'=====\'\n')

    return df


def passo_mun_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 38

    [Teste: Checar se existe algum número < 1 ou > 38.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ESC")

    # Substituindo valor 0 por None
    df.loc[df['MUN_ESC']==0,'MUN_ESC'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_ESC >= 1" E "MUN_ESC <= 38"

```

```

verifica_range(df, 'MUN_ESC', 1, 38)
log_output.info('\n\n=====\\n')

return df

def passo_zona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 254

    [Teste: Checar se existe algum número < 1 ou > 254.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB1']==0,'ZONA_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_TRAB1 >= 1" E "ZONA_TRAB1 <= 254"
    verifica_range(df, 'ZONA_TRAB1', 1, 254)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_TRAB1']==0,'SUBZONA_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_TRAB1 >= 1" E "SUBZONA_TRAB1 <= 9"
    verifica_range(df, 'SUBZONA_TRAB1', 1, 9)
    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_mun_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 38

    [Teste: Checar se existe algum número < 1 ou > 38.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB1']==0,'MUN_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB1 >= 1" E "MUN_TRAB1 <= 38"
    verifica_range(df, 'MUN_TRAB1', 1, 38)
    log_output.info('\n\n=====\\n')

    return df

def passo_zona_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 254

    [Teste: Checar se existe algum número < 1 ou > 254.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB2']==0,'ZONA_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_TRAB2 >= 1" E "ZONA_TRAB2 <= 254"
    verifica_range(df, 'ZONA_TRAB2', 1, 254)
    log_output.info('\n\n=====\\n')

    return df

```

```

def passo_subzona_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_TRAB2']==0, 'SUBZONA_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_TRAB2 >= 1" E "SUBZONA_TRAB2 <= 9"
    verifica_range(df, 'SUBZONA_TRAB2', 1, 9)
    log_output.info('\n\n=====\\n')

    return df


def passo_mun_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 38

    [Teste: Checar se existe algum número < 1 ou > 38.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB2']==0, 'MUN_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB2 >= 1" E "MUN_TRAB2 <= 38"
    verifica_range(df, 'MUN_TRAB2', 1, 38)
    log_output.info('\n\n=====\\n')

    return df


def passo_zona_orig(passo, df):

```

```

"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 254

[Teste: Checar se existe algum número < 1 ou > 254.
 Se encontrar, retornar erro indicando em qual linha.] 
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - ZONA_ORIG")

# Substituindo valor 0 por None
df.loc[df['ZONA_ORIG']==0,'ZONA_ORIG'] = None

# Verificando intervalo de valores - condições:
# "ZONA_ORIG >= 1" E "ZONA_ORIG <= 254"
verifica_range(df, 'ZONA_ORIG', 1, 254)
log_output.info('\n\n=====\\n')

return df


def passo_subzona_orig(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 9

[Teste: Checar se existe algum número < 1 ou > 9.
 Se encontrar, retornar erro indicando em qual linha.] 
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ORIG")

# Substituindo valor 0 por None
df.loc[df['SUBZONA_ORIG']==0,'SUBZONA_ORIG'] = None

# Verificando intervalo de valores - condições:
# "SUBZONA_ORIG >= 1" E "SUBZONA_ORIG <= 9"
verifica_range(df, 'SUBZONA_ORIG', 1, 9)
log_output.info('\n\n=====\\n')

return df


def passo_mun_orig(passo, df):
"""
Checar se existe algum erro

```

```

# #####Categorias
# > 1 a 38

[Teste: Checar se existe algum número < 1 ou > 38.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MUN_ORIG")

# Substituindo valor 0 por None
df.loc[df['MUN_ORIG']==0,'MUN_ORIG'] = None

# Verificando intervalo de valores - condições:
# "MUN_ORIG >= 1" E "MUN_ORIG <= 38"
verifica_range(df, 'MUN_ORIG', 1, 38)
log_output.info('\n\n=====\\n')

return df


def passo_zona_dest(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 254

[Teste: Checar se existe algum número < 1 ou > 254.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - ZONA_DEST")

# Substituindo valor 0 por None
df.loc[df['ZONA_DEST']==0,'ZONA_DEST'] = None

# Verificando intervalo de valores - condições:
# "ZONA_DEST >= 1" E "ZONA_DEST <= 254"
verifica_range(df, 'ZONA_DEST', 1, 254)
log_output.info('\n\n=====\\n')

return df


def passo_subzona_dest(passo, df):
"""
Checar se existe algum erro

# #####Categorias:

```

```

# > 1 a 9

[Teste: Checar se existe algum número < 1 ou > 9.
 Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```
log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DEST")
```

Substituindo valor 0 por None

```
df.loc[df['SUBZONA_DEST']==0,'SUBZONA_DEST'] = None
```

Verificando intervalo de valores - condições:

"SUBZONA_DEST >= 1" E "SUBZONA_DEST <= 9"

```
verifica_range(df, 'SUBZONA_DEST', 1, 9)
```

```
log_output.info('\n\n=====\\n')
```

```
return df
```

def passo_mun_dest(passo, df):

"""

Checar se existe algum erro

#####Categorias

> 1 a 38

[*Teste: Checar se existe algum número < 1 ou > 38.*
 Se encontrar, retornar erro indicando em qual linha.]

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```
log_tela.info("### PASSO " + str(passo) + " - MUN_DEST")
```

Substituindo valor 0 por None

```
df.loc[df['MUN_DEST']==0,'MUN_DEST'] = None
```

Verificando intervalo de valores - condições:

"MUN_DEST >= 1" E "MUN_DEST <= 38"

```
verifica_range(df, 'MUN_DEST', 1, 38)
```

```
log_output.info('\n\n=====\\n')
```

```
return df
```

def passo_serv_pas_orig(passo, df):

"""

*Substituir **0** por None*

*Substituir **2** por **0***

#####Categorias antigas

Valor/Descrição

```

-----/-----
0/Não Respondido
1/Sim
2/Não

#####Categorias novas
Valor/Descrição
-----/-----
0/Não
1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_ORIG")

df.loc[df['SERV_PAS_ORIG']==0, 'SERV_PAS_ORIG'] = None
df.loc[df['SERV_PAS_ORIG']==2, 'SERV_PAS_ORIG'] = 0

verifica_dummy(df, 'SERV_PAS_ORIG')
log_output.info('\n\n=====\\n')

return df


def passo_serv_pas_dest(passo, df):
"""
Substituir **0** por None
Substituir **2** por **0**

#####Categorias antigas
Valor/Descrição
-----/-----
0/Não Respondido
1/Sim
2/Não

#####Categorias novas
Valor/Descrição
-----/-----
0/Não
1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_DEST")

df.loc[df['SERV_PAS_DEST']==0, 'SERV_PAS_DEST'] = None
df.loc[df['SERV_PAS_DEST']==2, 'SERV_PAS_DEST'] = 0

verifica_dummy(df, 'SERV_PAS_DEST')

```

```

log_output.info('\n\n=====\\n')

return df

def passo_motivo_orig(passo, df):
    """
    * Substituir todos valores **6** por **10**
    * Substituir todos valores **7** por **6**
    * Substituir todos valores **8** por **7**
    * Substituir todos valores **9** por **8**
    * Substituir todos valores **10** por **9**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Trabalho/Indústria
    2/Trabalho/Comércio
    3/Trabalho/Serviços
    4/Escola/Educação
    5/Compras
    6/Negócios
    7/Médico/Dentista/Saúde
    8/Recreação/Visitas
    9/Residência

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Trabalho/Indústria
    2/Trabalho/Comércio
    3/Trabalho/Serviços
    4/Educação
    5/Compras
    6/Saúde
    7/Lazer
    8/Residência
    9/Outros

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - MOTIVO_ORIG")

# Substituindo valor 6 por 10
df.loc[df['MOTIVO_ORIG']==6, 'MOTIVO_ORIG'] = 10
# Substituindo valor 7 por 6
df.loc[df['MOTIVO_ORIG']==7, 'MOTIVO_ORIG'] = 6
# Substituindo valor 8 por 7
df.loc[df['MOTIVO_ORIG']==8, 'MOTIVO_ORIG'] = 7

```

```

# Substituindo valor 9 por 8
df.loc[df['MOTIVO_ORIG']==9,'MOTIVO_ORIG'] = 8
# Substituindo valor 10 por 9
df.loc[df['MOTIVO_ORIG']==10,'MOTIVO_ORIG'] = 9
# Substituindo valor 0 por None
df.loc[df['MOTIVO_ORIG']==0,'MOTIVO_ORIG'] = None

# Verificando intervalo de valores - condições:
# "MOTIVO_ORIG >= 1" E "MOTIVO_ORIG <= 9"
verifica_range(df, 'MOTIVO_ORIG', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_motivo_dest(passo, df):
"""
Substituir todos valores **6** por **10**
Substituir todos valores **7** por **6**
Substituir todos valores **8** por **7**
Substituir todos valores **9** por **8**
Substituir todos valores **10** por **9**
Substituir todos valores **0** por **None**

#### Categorias anteriores
Valor/Descrição
----/----
1/Trabalho/Indústria
2/Trabalho/Comércio
3/Trabalho/Serviços
4/Escola/Educação
5/Compras
6/Negócios
7/Médico/Dentista/Saúde
8/Recreação/Visitas
9/Residência

#### Categorias novas
Valor/Descrição
----/----
1/Trabalho/Indústria
2/Trabalho/Comércio
3/Trabalho/Serviços
4/Educação
5/Compras
6/Saúde
7/Lazer
8/Residência
9/Outros

[Teste: Checar se existe algum número < 1 ou > 9.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:

```

```

: return: dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - MOTIVO_DEST")

# Substituindo valor 6 por 10
df.loc[df['MOTIVO_DEST']==6,'MOTIVO_DEST'] = 10
# Substituindo valor 7 por 6
df.loc[df['MOTIVO_DEST']==7,'MOTIVO_DEST'] = 6
# Substituindo valor 8 por 7
df.loc[df['MOTIVO_DEST']==8,'MOTIVO_DEST'] = 7
# Substituindo valor 9 por 8
df.loc[df['MOTIVO_DEST']==9,'MOTIVO_DEST'] = 8
# Substituindo valor 10 por 9
df.loc[df['MOTIVO_DEST']==10,'MOTIVO_DEST'] = 9
# Substituindo valor 0 por None
df.loc[df['MOTIVO_DEST']==0,'MOTIVO_DEST'] = None

# Verificando intervalo de valores - condições:
# "MOTIVO_DEST >= 1" E "MOTIVO_DEST <= 9"
verifica_range(df, 'MOTIVO_DEST', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_modo1(passo, df):
    """
    Substituir valores da coluna "MOD01"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **5** por **3**
    * Substituir todos valores **6** por **4**
    * Substituir todos valores **7** por **5**
    * Substituir todos valores **8** por **6**
    * Substituir todos valores **9** por **7**
    * Substituir todos valores **10** por **8**
    * Substituir todos valores **11** por **9**
    * Substituir todos valores **12** por **10**
    * Substituir todos valores **13** por **11**
    * Substituir todos valores **14** por **12**
    * Substituir todos valores **15** por **12**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Ônibus diesel
    2/Trólebus
    3/Ônibus Fretado
    4/Escolar
    5/Dirigindo Automóvel
    6/Passageiro de Automóvel

```

```

7/Táxi
8/Lotação/Perua
9/Metrô
10/Trem
11/Moto
12/Bicicleta
13/A Pé
14/Caminhão
15/Outros

##### Categorias novas
Valor/Descrição
----/----
1/Ônibus
2/Ônibus Escolar / Empresa
3/Dirigindo Automóvel
4/Passageiro de Automóvel
5/Táxi
6/Lotação / Perua / Van / Microônibus
7/Metrô
8/Trem
9/Moto
10/Bicicleta
11/A Pé
12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```

log_tela.info("### PASSO " + str(passo) + " - MODO1")

df.loc[df['MODO1']==2,'MODO1'] = 1
df.loc[df['MODO1']==3,'MODO1'] = 2
df.loc[df['MODO1']==4,'MODO1'] = 2
df.loc[df['MODO1']==5,'MODO1'] = 3
df.loc[df['MODO1']==6,'MODO1'] = 4
df.loc[df['MODO1']==7,'MODO1'] = 5
df.loc[df['MODO1']==8,'MODO1'] = 6
df.loc[df['MODO1']==9,'MODO1'] = 7
df.loc[df['MODO1']==10,'MODO1'] = 8
df.loc[df['MODO1']==11,'MODO1'] = 9
df.loc[df['MODO1']==12,'MODO1'] = 10
df.loc[df['MODO1']==13,'MODO1'] = 11
df.loc[df['MODO1']==14,'MODO1'] = 12
df.loc[df['MODO1']==15,'MODO1'] = 12
df.loc[df['MODO1']==0,'MODO1'] = None

# Verificando intervalo de valores - condições:
# "MODO1 >= 1" E "MODO1 <= 12"
verifica_range(df, 'MODO1', 1, 12)
log_output.info('\'\n\'=====\\n')
```

```

    return df

def passo_modo2(passo, df):
    """
    Substituir valores da coluna "MODO2"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **5** por **3**
    * Substituir todos valores **6** por **4**
    * Substituir todos valores **7** por **5**
    * Substituir todos valores **8** por **6**
    * Substituir todos valores **9** por **7**
    * Substituir todos valores **10** por **8**
    * Substituir todos valores **11** por **9**
    * Substituir todos valores **12** por **10**
    * Substituir todos valores **13** por **11**
    * Substituir todos valores **14** por **12**
    * Substituir todos valores **15** por **12**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Ônibus diesel
    2/Trólebus
    3/Ônibus Fretado
    4/Escolar
    5/Dirigindo Automóvel
    6/Passageiro de Automóvel
    7/Táxi
    8/Lotação/Perua
    9/Metrô
    10/Trem
    11/Moto
    12/Bicicleta
    13/A Pé
    14/Caminhão
    15/Outros

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Ônibus
    2/Ônibus Escolar / Empresa
    3/Dirigindo Automóvel
    4/Passageiro de Automóvel
    5/Táxi
    6/Lotação / Perua / Van / Microônibus
    7/Metrô
    8/Trem

```

9/Moto
 10/Bicicleta
 11/A Pé
 12/Outros

*[Teste: Checar se existe algum número < 1 ou > 12.
 Se encontrar, retornar erro indicando em qual linha.]*

```

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - MODO3")

df.loc[df['MODO2']==2,'MODO2'] = 1
df.loc[df['MODO2']==3,'MODO2'] = 2
df.loc[df['MODO2']==4,'MODO2'] = 2
df.loc[df['MODO2']==5,'MODO2'] = 3
df.loc[df['MODO2']==6,'MODO2'] = 4
df.loc[df['MODO2']==7,'MODO2'] = 5
df.loc[df['MODO2']==8,'MODO2'] = 6
df.loc[df['MODO2']==9,'MODO2'] = 7
df.loc[df['MODO2']==10,'MODO2'] = 8
df.loc[df['MODO2']==11,'MODO2'] = 9
df.loc[df['MODO2']==12,'MODO2'] = 10
df.loc[df['MODO2']==13,'MODO2'] = 11
df.loc[df['MODO2']==14,'MODO2'] = 12
df.loc[df['MODO2']==15,'MODO2'] = 12
df.loc[df['MODO2']==0,'MODO2'] = None

# Verificando intervalo de valores - condições:
# "MODO2 >= 1" E "MODO2 <= 12"
verifica_range(df, 'MODO2', 1, 12)
log_output.info('\n\n=====\\n')

return df

```

def passo_modo3(passo, df):
 """
 Substituir valores da coluna "MODO3"

 * Substituir todos valores **2** por **1**
 * Substituir todos valores **3** por **2**
 * Substituir todos valores **4** por **2**
 * Substituir todos valores **5** por **3**
 * Substituir todos valores **6** por **4**
 * Substituir todos valores **7** por **5**
 * Substituir todos valores **8** por **6**
 * Substituir todos valores **9** por **7**
 * Substituir todos valores **10** por **8**
 * Substituir todos valores **11** por **9**
 * Substituir todos valores **12** por **10**
 * Substituir todos valores **13** por **11**
 * Substituir todos valores **14** por **12**

```
* Substituir todos valores **15** por **12**
* Substituir todos valores **0** por **None**
```

```
##### Categorias anteriores
```

```
Valor/Descrição
```

```
----/----
```

```
1/Ônibus diesel
```

```
2/Trólebus
```

```
3/Ônibus Fretado
```

```
4/Escolar
```

```
5/Dirigindo Automóvel
```

```
6/Passageiro de Automóvel
```

```
7/Táxi
```

```
8/Lotação/Perua
```

```
9/Metrô
```

```
10/Trem
```

```
11/Moto
```

```
12/Bicicleta
```

```
13/A Pé
```

```
14/Caminhão
```

```
15/Outros
```

```
##### Categorias novas
```

```
Valor/Descrição
```

```
----/----
```

```
1/Ônibus
```

```
2/Ônibus Escolar / Empresa
```

```
3/Dirigindo Automóvel
```

```
4/Passageiro de Automóvel
```

```
5/Táxi
```

```
6/Lotação / Perua / Van / Microônibus
```

```
7/Metrô
```

```
8/Trem
```

```
9/Moto
```

```
10/Bicicleta
```

```
11/A Pé
```

```
12/Outros
```

```
[Teste: Checar se existe algum número < 1 ou > 12.
```

```
Se encontrar, retornar erro indicando em qual linha.]
```

```
:param passo: Número do passo atual para registro/log
```

```
:param df:
```

```
:return: sem retorno
```

```
"""
```

```
log_tela.info("### PASSO " + str(passo) + " - MOD03")
```

```
df.loc[df['MOD03']==2,'MOD03'] = 1
```

```
df.loc[df['MOD03']==3,'MOD03'] = 2
```

```
df.loc[df['MOD03']==4,'MOD03'] = 2
```

```
df.loc[df['MOD03']==5,'MOD03'] = 3
```

```
df.loc[df['MOD03']==6,'MOD03'] = 4
```

```
df.loc[df['MOD03']==7,'MOD03'] = 5
```

```
df.loc[df['MOD03']==8,'MOD03'] = 6
```

```
df.loc[df['MOD03']==9,'MOD03'] = 7
```

```

df.loc[df['MODO3']==10,'MODO3'] = 8
df.loc[df['MODO3']==11,'MODO3'] = 9
df.loc[df['MODO3']==12,'MODO3'] = 10
df.loc[df['MODO3']==13,'MODO3'] = 11
df.loc[df['MODO3']==14,'MODO3'] = 12
df.loc[df['MODO3']==15,'MODO3'] = 12
df.loc[df['MODO3']==0,'MODO3'] = None

# Verificando intervalo de valores - condições:
# "MODO3 >= 1" E "MODO3 <= 12"
verifica_range(df, 'MODO3', 1, 12)
log_output.info('\n\n=====\\n')

return df

def passo_modo4(passo, df):
    """
    Nada há que se fazer em relação à coluna "MODO4" - não há dados de 1987,
    coluna permanecerá vazia
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MODO4")

    df['MODO4'] = None

    log_output.info('\n\n=====\\n')

    return df

def passo_modo_prin(passo, df):
    """
    Substituir valores da coluna "MODO_PRIN"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **5** por **3**
    * Substituir todos valores **6** por **4**
    * Substituir todos valores **7** por **5**
    * Substituir todos valores **8** por **6**
    * Substituir todos valores **9** por **7**
    * Substituir todos valores **10** por **8**
    * Substituir todos valores **11** por **9**
    * Substituir todos valores **12** por **10**
    * Substituir todos valores **13** por **11**
    * Substituir todos valores **14** por **12**
    * Substituir todos valores **15** por **12**
    * Substituir todos valores **0** por **None**

    #### Categorias anteriores

```

Valor/Descrição

----/----

1/*Ônibus diesel*
 2/*Trólebus*
 3/*Ônibus Fretado*
 4/*Escolar*
 5/*Dirigindo Automóvel*
 6/*Passageiro de Automóvel*
 7/*Táxi*
 8/*Lotação/Perua*
 9/*Metrô*
 10/*Trem*
 11/*Moto*
 12/*Bicicleta*
 13/*A Pé*
 14/*Caminhão*
 15/*Outros*

Categorias novas

Valor/Descrição

----/----

1/*Ônibus*
 2/*Ônibus Escolar / Empresa*
 3/*Dirigindo Automóvel*
 4/*Passageiro de Automóvel*
 5/*Táxi*
 6/*Lotação / Perua / Van / Microônibus*
 7/*Metrô*
 8/*Trem*
 9/*Moto*
 10/*Bicicleta*
 11/*A Pé*
 12/*Outros*

[*Teste: Checar se existe algum número < 1 ou > 12.*
Se encontrar, retornar erro indicando em qual linha.]

```
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - MODO_PRIN")

df.loc[df['MODO_PRIN']==2, 'MODO_PRIN'] = 1
df.loc[df['MODO_PRIN']==3, 'MODO_PRIN'] = 2
df.loc[df['MODO_PRIN']==4, 'MODO_PRIN'] = 2
df.loc[df['MODO_PRIN']==5, 'MODO_PRIN'] = 3
df.loc[df['MODO_PRIN']==6, 'MODO_PRIN'] = 4
df.loc[df['MODO_PRIN']==7, 'MODO_PRIN'] = 5
df.loc[df['MODO_PRIN']==8, 'MODO_PRIN'] = 6
df.loc[df['MODO_PRIN']==9, 'MODO_PRIN'] = 7
df.loc[df['MODO_PRIN']==10, 'MODO_PRIN'] = 8
df.loc[df['MODO_PRIN']==11, 'MODO_PRIN'] = 9
df.loc[df['MODO_PRIN']==12, 'MODO_PRIN'] = 10
df.loc[df['MODO_PRIN']==13, 'MODO_PRIN'] = 11
```

```

df.loc[df['MODO_PRIN']==14, 'MODO_PRIN'] = 12
df.loc[df['MODO_PRIN']==15, 'MODO_PRIN'] = 12
df.loc[df['MODO_PRIN']==0, 'MODO_PRIN'] = None

# Verificando intervalo de valores - condições:
# "MODO_PRIN >= 1" E "MODO_PRIN <= 12"
verifica_range(df, 'MODO_PRIN', 1, 12)
log_output.info('\n\n=====\\n')

return df

def passo_tipo_vitag(passo, df):
    """
    * Substituir os valores **0** por **None**

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Coletivo
    # 2/Individual
    # 3/A pé

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]"""

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - TIPO_VIAG")

# Substituindo valor 0 por None
df.loc[df['TIPO_VIAG']==0, 'TIPO_VIAG'] = None

# Verificando intervalo de valores - condições:
# "MODO_PRIN >= 1" E "MODO_PRIN <= 3"
verifica_range(df, 'TIPO_VIAG', 1, 3)

log_output.info('\n\n=====\\n')
return df

def passo_tipo_est_auto(passo, df):
    """
    Substituir valores da coluna "TIPO_EST_AUTO"

    * Substituir todos valores **1** por **5**
    * Substituir todos valores **6** por **1**

    #####Categorias anteriores
    Valor/Descrição
    """

```

```

----/----
1/Zona Azul / Parquímetro
2/Estacionamento Particular
3/Estacionamento Próprio
4/Estacionamento Patrocinado
5/Meio-Fio
6/Não estacionou

# #####Categorias novas
# Valor/Descrição
# ----/----
# 0/Não Respondeu
# 1/Não Estacionou
# 2/Estacionamento Particular (Avulso / Mensal)
# 3/Estacionamento Próprio
# 4/Estacionamento Patrocinado
# 5/Rua (meio fio / zona azul / zona marrom / parquímetro)

[Teste: Checar se existe algum número < 0 ou > 5.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - TIPO_EST_AUTO")

# Substituindo valor 1 por 5
df.loc[df['TIPO_EST_AUTO']==1, 'TIPO_EST_AUTO'] = 5
# Substituindo valor 6 por 1
df.loc[df['TIPO_EST_AUTO']==6, 'TIPO_EST_AUTO'] = 1

# Verificando intervalo de valores - condições:
# "TIPO_EST_AUTO >= 0" E "TIPO_EST_AUTO <= 5"
verifica_range(df, 'TIPO_EST_AUTO', 0, 5)
log_output.info('\n\n=====\\n')

return df

def passo_valor_est_auto(passo, df, deflator):
"""
Nada há que se fazer em relação à coluna "VALOR_EST_AUTO".
Não há dados de 1987, coluna permanecerá vazia
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - VALOR_EST_AUTO")

df['VALOR_EST_AUTO'] = None

log_output.info('\n\n=====\\n')

```

```

    return df

def passo_tot_viang(passo, df):
    """
    ##### ATENÇÃO #####
    # ESTA FUNÇÃO SÓ DEVE SER #
    # EXECUTADA APÓS A GERAÇÃO#
    # DE TODOS OS ID'S, POIS #
    # HÁ UMA DESCONFIANÇA   #
    # QUANTO À QUALIDADE   #
    # DESTE DADO           #
    #####
    Calcula e confere o campo TOT_VIAG,
    baseado no maior valor de NO_VIAG para cada pessoa
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com o "TOT_VIAG" corrigido
"""
log_tela.info("### PASSO " + str(passo) + " - TOT_VIAG")
log_tela.warning('\n\n#####\n' +
                 'Este passo DEVE ser executado apenas após a \n' +
                 'execução dos passos que geram os novos IDs\n' +
                 '#####\n')

# 'Calculando' o máximo de viagens para cada indivíduo
# Agrupa por ID_PESS e encontra o máximo para NO_VIAG.
# O resultado é um objeto do tipo "Series" cujo index é
# ID_PESS e a variável é NO_VIAG. Em seguida transforma
# esse objeto num DataFrame e depois renomeia a coluna
# NO_VIAG para MAX_VIAG.
df_max = pd.DataFrame(
    df.groupby('ID_PESS')['NO_VIAG'].max()).rename(
        columns={'NO_VIAG': 'MAX_VIAG'})
# Criando um novo dataframe auxiliar apenas com ID_PESS, apenas para ficar mais leve

# O passo seguinte é atribuir o MAX_VIAG à coluna TOT_VIAG
# do dataframe df, linkando por ID_PESS. Isso é feito usando o
# método 'merge' da biblioteca pandas.
df['TOT_VIAG'] = pd.merge(df, df_max, how='left', left_on='ID_PESS',
                           right_index=True)[['MAX_VIAG']]

log_output.info('TOT_VIAG:\n\n' +
                '      Situação final dos dados: \n' +
                str(df['TOT_VIAG'].describe()) + '\n' +
                '      TOT_VIAG: Situação final dos dados: \n' +
                str(df['TOT_VIAG'].value_counts()))

# Agora uma função que irá verificar se para todo "ID_PESS" o "TOT_VIAG"
# é igual ao 'NO_VIAG' máximo.
def verifica_no_viang_tot_viang(row):
    if row['NO_VIAG'] != row['TOT_VIAG']:
        log_output.warning('TOT_VIAG: Erro encontrado na linha '

```

```
+ str(row) + ':\\n'
+ ' => ' + row
)
df.loc[:, ['ID_PESS', 'NO_VIAG', 'TOT_VIAG']]\
.groupby('ID_PESS')\
.agg({'NO_VIAG': 'max', 'ID_PESS': 'max', 'TOT_VIAG': 'max'})\
.apply(verifica_no_viag_tot_viag, axis=1)
log_output.info('\\n\\n=====\\n')

return df
```

3.7 Funções que geram os “NO”s e os “ID”s

Função/Variável	Status
gera_nos_e_ids	Verificar NO VIAG

```
In [ ]: log_tela.info('Definindo funções que geram os "NO"s e os "ID"s')
log_output.info('\n\n=====\\n')
```

```
def gera_nos_e_ids(passo, df):
    """
    Esta função gera todos os IDs e todos os NOs das variáveis:
    DOM (domicílio), FAM (família), PESS (pessoa) e VIAG (viagem)
    A ordem de geração é:
    NO_DOM, ID_DOM, NO_FAM, ID_FAM, NO_PESS, ID_PESS, NO_VIAG, ID_VIAG
    Esta ordem é fixa pois cada uma dessas variáveis depende da geração
    da variável anterior.

    Os NOs são gerados como se fossem subíndices.
    No caso dos domicílios, eles são contabilizados dentro de cada zona.
    Assim, cada novo domicílio que aparece na listagem recebe um número
    que representa sua posição na sequência de domicílios dentro da zona
    na qual ele está contido.
    As famílias seguem o mesmo raciocínio, com relação ao domicílio, as
    pessoas com relação às famílias e as viagens com relação às pessoas.

    Deve-se apenas tomar cuidado com as viagens, pois existem pessoas que
    não realizaram viagens. Neste caso, estes registros devem contabilizar
    o NO_VIAG como zero, e não como 1. Para contemplar estes casos, vamos
    utilizar a variável F_VIAG, que representa quando há ou não viagem
    naquele registro (F_VIAG==1 tem viagem, F_VIAG==0 não tem viagem).
    Assim, o que precisamos fazer para calcular o NO_VIAG é agrupar os
    registros por pessoa (ID_PESS) e, dentro de cada agrupamento, somar
    o valor de F_VIAG registro a registro (linha por linha)
    cumulativamente. No final do processo, atribui-se zero a NO_VIAG
    quando F_VIAG for igual a zero.
    """

def gera_id_dom(row):
    """
    Gera o ID_DOM baseado no 'ANO', na 'ZONA_DOM' e no 'NO_DOM'
    O argumento passado é a "linha".
    Uso:
        df['ID_DOM'] = df.apply(gera_ID_DOM, axis=1)
    Retorna: ID_DOM da respectiva linha
    """
    # Formatando o ano para string
    ano = str(row['ANO'])

    # Formatando a zona para ficar como string com 3 caracteres
    zona = str('%03d' % row['ZONA_DOM'])
```

```

# Formatando no_dom para ficar como string com 4 caracteres
no_dom = str('%04d' % row['NO_DOM'])

# Retornando o número inteiro correspondente à string
# concatenada de ano + zona + no_dom
return ano + zona + no_dom


def gera_id_fam(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
    Uso:
        df['ID_FAM'] = df.apply(gera_ID_FAM, axis=1)
    Retorna: ID_FAM da respectiva linha
    """
    # Formatando id_dom como string
    id_dom = str(row['ID_DOM'])

    # Formatando no_fam como string para ficar com 2 caracteres
    no_fam = str('%02d' % row['NO_FAM'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_DOM com NO_FAM
    return id_dom + no_fam


def gera_id_pess(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
    Uso:
        df['ID_PESS'] = df.apply(gera_ID_PESS, axis=1)
    Retorna: ID_PESS da respectiva linha
    """
    # Formatando id_fam como string
    id_fam = str(row['ID_FAM'])

    # Formatando no_pess como string para ficar com 2 caracteres
    no_pess = str('%02d' % row['NO_PESS'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_FAM com NO_PESS
    return id_fam + no_pess


def gera_id_viag(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
    Uso:
        df['ID_VIAG'] = df.apply(gera_ID_VIAG, axis=1)
    Retorna ID_VIAG da respectiva linha
    """
    # Formatando id_pess como string
    id_pess = str(row['ID_PESS'])

```

```

# Formatando no_viag como string para ficar com 2 caracteres
no_viag = str('%02d' % row['NO_VIAG'])

# Retornando o número inteiro correspondente à string
# concatenada de ID_PESS com NO_VIAG
return id_pess + no_viag

#gera NO_DOM
log_tela.info("Gerando NO_DOM")
df['NO_DOM'] = df.groupby('ZONA_DOM', sort=False)['F_DOM'].cumsum()
log_tela.info("NO_DOM gerado")
log_tela.info("Gerando ID_DOM")
#gera ID_DOM
df['ID_DOM'] = df.apply(gera_id_dom, axis=1)
log_tela.info("ID_DOM gerado")

#gera NO_FAM
log_tela.info("Gerando NO_FAM")
df['NO_FAM'] = df.groupby('ID_DOM', sort=False)['F_FAM'].cumsum()
log_tela.info("NO_FAM gerado")
log_tela.info("Gerando ID_FAM")
#gera ID_FAM
df['ID_FAM'] = df.apply(gera_id_fam, axis=1)
log_tela.info("ID_FAM gerado")

#gera NO_PESS
log_tela.info("Gerando NO_PESS")
df['NO_PESS'] = df.groupby('ID_FAM', sort=False)['F_PESS'].cumsum()
log_tela.info("NO_PESS gerado")
log_tela.info("Gerando ID_PESS")
#gera ID_PESS
df['ID_PESS'] = df.apply(gera_id_pess, axis=1)
log_tela.info("ID_PESS gerado")

#gera NO_VIAG
log_tela.info("Gerando NO_VIAG")
df['NO_VIAG'] = df.groupby('ID_PESS', sort=False)['F_VIAG'].cumsum()
# Verificando se FE_VIAG == 0 e zerando NO_VIAG nesse caso
df.loc[df['F_VIAG'] == 0, 'NO_VIAG'] = 0
log_tela.info("NO_VIAG gerado")
log_tela.info("Gerando ID_VIAG")
#gera ID_VIAG
df['ID_VIAG'] = df.apply(gera_id_viag, axis=1)
log_tela.info("ID_VIAG gerado")

return df

```

3.8 Função relativa à entrevista

Função/Variável	Status
passo_cd_entre	OK

Obs: o passo_cd_entre deve ser executado após o passo_tot_vitag.

```
In [ ]: log_tela.info('Definindo função relativa à entrevista')
log_output.info('\n\n=====\\n')

def passo_cd_entre(passo, df):
    """
    #####
    #      ATENÇÃO      #
    # ESTA FUNÇÃO SÓ DEVE SER #
    # EXECUTADA APÓS A GERAÇÃO#
    #      DO TOT_VIAG     #
    #####
    ----
    Substituir valores da coluna "CD_ENTRE"
    Todas viagens são consideradas "completas", segundo informações do Metrô

    * sem viagem: se TOT_VIAG == 0
    * com viagem: se TOT_VIAG != 0

    # #####Categorias novas
    # | Valor | Descrição |
    # | ----- | ----- |
    # | 0 | Completa sem viagem |
    # | 1 | Completa com viagem |

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""

log_tela.info("### PASSO " + str(passo) + " - CD_ENTRE")
log_tela.warning('\n\n#####\n' + 'Este passo DEVE ser executado apenas após a\n' +
                 'execução do passo que geram o TOT_VIAG.\n' +
                 '#####\n')

# Definindo 'CD_ENTRE' baseado no valor de 'TOT_VIAG'
df.loc[df['TOT_VIAG'] == 0, 'CD_ENTRE'] = 0
df.loc[df['TOT_VIAG'] != 0, 'CD_ENTRE'] = 1

verifica_dummy(df, 'CD_ENTRE')
log_output.info('\n\n=====\\n')

return df
```

```
In [ ]: def passo_correcoes(passo, df):
    log_tela.info("### PASSO " + str(passo) + " - Correções")
    log_tela.info("Corrigindo 'SERV_PAS' caso TOT_VIAG seja nulo.")
    df.loc[df['TOT_VIAG'] == 0, 'SERV_PAS_ORIG'] = 0
    df.loc[df['TOT_VIAG'] == 0, 'SERV_PAS_DEST'] = 0

    return df
```

4 Definindo a função principal (main)

Esta função vai ler os arquivos (csv) necessários e, então, irá efetuar cada passo, chamando as respectivas funções que foram definidas acima. Ao final, esta função irá salvar o resultado das transformações realizadas num arquivo csv.

```
In [ ]: def main():
    start_time = time.time()

    log_tela.info("Horário de início da execução: %s" %
                  time.strftime("%H:%M", time.localtime(start_time)))

    # ----
    log_tela.info('Lendo arquivos CSV')

    log_output.info('Lendo CSV da base original da OD.')
    od = pd.read_csv('bases/OD_1987.csv', sep=';', decimal=',')

    deflator = 0.0966466592714303
    log_output.info('Deflator a ser utilizado para correção monetária: ' +
                    str(deflator))

    # Filtrando dataframe para pegar apenas uma amostra
    #od = od[:1000].copy()

    log_output.info('\n\n=====\\n')
    log_tela.info('Pré-Processamento.')
    od = pre_processamento(od)

    log_tela.info('Imprimindo a lista de variáveis do dataframe da OD.')
    log_tela.debug(od.columns.tolist())

    #Contador de 'PASSO'
    passo = 1

    # ----
    # ##Passo: "ANO"
    od = passo_ano(passo, od)
    passo += 1

    # ----
    # ##Passo: "DIA_SEM"
    od = passo_dia_sem(passo, od)
    passo += 1

    # ----
    # ##Passo: "UCODs"
    od = passo_ucods(passo, od)
    passo += 1

    # ----
    # ##Passo: "ZONA_DOM"
    od = passo_zona_dom(passo, od)
    passo += 1
```

```

# -----
# ##Passo: "SUBZONA_DOM"
od = passo_subzona_dom(passo, od)
passo += 1

# -----
# ##Passo: "MUN_DOM"
od = passo_mun_dom(passo, od)
passo += 1

# -----
# ##Passo: "F_DOM"
od = passo_f_dom(passo, od)
passo += 1

# -----
# ##Passo: "FE_DOM"
od = passo_fe_dom(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_DOM"
od = passo_tipo_dom(passo, od)
passo += 1

# -----
# ##Passo: "TOT_FAM"
od = passo_tot_fam(passo, od)
passo += 1

# -----
# ##Passo: "F_FAM"
od = passo_f_fam(passo, od)
passo += 1

# -----
# ##Passo: "FE_FAM"
od = passo_fe_fam(passo, od)
passo += 1

# -----
# ##Passo: "COND_MORA"
od = passo_cond_mora(passo, od)
passo += 1

# -----
# ##Passo: "QT_AUTO"
od = passo_qt_auto(passo, od)
passo += 1

# -----
# ##Passo: "QT_BICI"
od = passo_qt_bici(passo, od)
passo += 1

```

```

# -----
# ##Passo: "QT_MOTO"
od = passo_qt_moto(passo, od)
passo += 1

# -----
# ##PASSO: "REN_FAM"
od = passo_ren_fam(passo, od, deflator)
passo += 1

# -----
# ##Passo: "CD_RENFAM"
od = passo_cd_renfam(passo, od)
passo += 1

# -----
# ##Passo: "F_PESS"
od = passo_f_pess(passo, od)
passo += 1

# -----
# ##Passo: "FE_PESS"
od = passo_fe_pess(passo, od)
passo += 1

# -----
# ##Passo: "SIT_FAM"
od = passo_sit_fam(passo, od)
passo += 1

# -----
# ##Passo: "IDADE"
od = passo_idade(passo, od)
passo += 1

# -----
# ##Passo: "SEXO"
od = passo_sexo(passo, od)
passo += 1

# -----
# ##Passo: "GRAU_INSTR"
od = passo_grau_instr(passo, od)
passo += 1

# -----
# ##Passo: "OCUP"
od = passo_ocup(passo, od)
passo += 1

# -----
# ##Passo: "SETOR_ATIV"
od = passo_setor_ativ(passo, od)

```

```

    passo += 1

    # ##Passo: "REN_IND"
    od = passo_ren_ind(passo, od, deflator)
    passo += 1

    # ##Passo: "CD_RENIND"
    od = passo_cd_renind(passo, od)
    passo += 1

    # -----
    # ##Passo: "F_VIAG"
    od = passo_f_viag(passo, od)
    passo += 1

    # -----
    # ##Passo: "FE_VIAG"
    od = passo_fe_viag(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_ESC"
    od = passo_zona_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_ESC"
    od = passo_subzona_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_ESC"
    od = passo_mun_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "ESTUDA"
    # Este passo deve vir após os passos de localização da escola
    od = passo_estuda(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_TRAB1"
    od = passo_zona_trab1(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_TRAB1"
    od = passo_subzona_trab1(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_TRAB1"
    od = passo_mun_trab1(passo, od)

```

```

    passo += 1

    # -----
    # ##Passo: "ZONA_TRAB2"
    od = passo_zona_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_TRAB2"
    od = passo_subzona_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_TRAB2"
    od = passo_mun_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_ORIG"
    od = passo_zona_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_ORIG"
    od = passo_subzona_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_ORIG"
    od = passo_mun_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_DEST"
    od = passo_zona_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_DEST"
    od = passo_subzona_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_DEST"
    od = passo_mun_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "SERV_PAS_ORIG"
    od = passo_serv_pas_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "SERV_PAS_DEST"

```

```

od = passo_serv_pas_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_ORIG"
od = passo_motivo_orig(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_DEST"
od = passo_motivo_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOD01"
od = passo_modo1(passo, od)
passo += 1

# -----
# ##Passo: "MOD02"
od = passo_modo2(passo, od)
passo += 1

# -----
# ##Passo: "MOD03"
od = passo_modo3(passo, od)
passo += 1

# -----
# ##Passo: "MOD04"
od = passo_modo4(passo, od)
passo += 1

# -----
# ##Passo: "MODO_PRIN"
od = passo_modo_prin(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_VIAG"
od = passo_tipo_vitag(passo, od)
passo += 1

# -----
# ##"H_SAIDA"; "MIN_SAIDA"; "ANDA_ORIG"; "H_CHEG"; "MIN_CHEG";
# "ANDA_DEST"; "DIST_VIAG" e "DURACAO"
# Nada há que se fazer em relação aos dados das colunas acima mencionadas

# -----
# ##Passo: "TIPO_EST_AUTO"
od = passo_tipo_est_auto(passo, od)
passo += 1

# -----

```

```

# ##Passo: "VALOR_EST_AUTO"
od = passo_valor_est_auto(passo, od, deflator)
passo += 1

# -----
# ##Passo: Coordenadas
od = coordenadas(passo, od)
passo += 1

## O passo TOT_VIAG apenas é chamado após a geração dos IDs.

# -----
# ##Passo: Gerando NO's e ID's
od = gera_nos_e_ids(passo, od)
passo += 1

# -----
# ##Passo: "TOT_VIAG"
od = passo_tot_viag(passo, od)
passo += 1

# -----
# ##Passo: "CD_ENTRE"
od = passo_cd_entre(passo, od)
passo += 1

# -----
# ##Passo: Correções
od = passo_correcoes(passo, od)
passo += 1

log_tela.info('Salvando dataframe como arquivo CSV')
# -----
# ## Salvando o dataframe num arquivo local
od.to_csv('outputs/1987_od.csv', sep=';', decimal=',', index=False)

log_tela.info("Base gerada. Arquivo: outputs/1987_od.csv")

log_tela.info("Tempo total de execução: %s segundos" %
              (time.time() - start_time))

log_tela.info("Horário de finalização: %s" %
              (time.strftime("%H:%M", time.localtime(time.time()))))

log_tela.info("Terminou o main")

```

5 RUN the main() function....

```
In [ ]: if __name__ == "__main__":
    main()
```

rotina_final_1997

January 4, 2016

1 Setup Inicial

```
In [ ]: # coding: utf-8
        import math
        import logging
        import time
        import pandas as pd

        # Faz os gráficos um pouco mais bonitos
        pd.set_option('display.mpl_style', 'default')
```

2 Definindo Loggers

Define os loggers

Estes ‘loggers’ serão utilizados para salvar as saídas (outputs) em um arquivo de texto no diretório ‘outputs’.

ref: <http://stackoverflow.com/questions/17035077/python-logging-to-multiple-log-files-from-different-classes>

ATENÇÃO: RODAR O BLOCO ABAIXO APENAS UMA VEZ, SE ESTE BLOCO FOR EXECUTADO MAIS DE UMA VEZ OS LOGs SERÃO DUPLICADOS.

```
In [ ]: log_formatter = logging.Formatter('%(asctime)s | %(levelname)s: %(message)s')

log_output = logging.getLogger('log_output')
FH_output = logging.FileHandler(
            'outputs/1997_output.log', mode='w')
FH_output.setFormatter(log_formatter)
log_output.setLevel(logging.INFO)
log_output.addHandler(FH_output)
log_output.propagate = False

# Este logger (log_tela) loga na tela e também
# no arquivo de output, junto com o conteúdo do
# logger log_output
log_tela = logging.getLogger('log_tela')
SH_tela = logging.StreamHandler()
SH_tela.setFormatter(log_formatter)
log_tela.addHandler(SH_tela)
log_tela.addHandler(FH_output)
log_tela.setLevel(logging.INFO)
log_tela.propagate = False
```

3 Funções de 1997

3.1 Funções gerais assessórias

Função	Status
consulta_refext	OK
verifica_dummy	OK
verifica_range	OK
pre_processamento	OK
coordenadas	OK

```
In [ ]: log_tela.info('Definindo as funções gerais assessórias')
log_output.info('\n\n=====\\n')

def verifica_dummy(df, variavel):
    """
    Verifica se uma variável, dummy, contém algum valor diferente de 0 ou de 1.
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados.
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_dummy(dataframe, 'coluna a ser verificada')
    """
    contador_de_erros = 0
    log_tela.info('Verificando a variável Dummy: ' + variavel)

    df_erros = df[(df[variavel] != 0) & (df[variavel] != 1)]
    if len(df_erros[variavel].value_counts()) > 0:
        log_tela.warning(variavel + ": " +
                          str(len(df_erros[variavel].value_counts())) +
                          " erros encontrados:\\n" +
                          str(df_erros[variavel].value_counts()))
    else:
        log_tela.info(variavel + ": Nenhum erro encontrado.")

def verifica_range(df, variavel, valor_menor, valor_maior):
    """
    Verifica se uma variável, do tipo número inteiro, contém algum valor menor
    que "valor_menor" ou maior que "valor_maior".
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados
    :param valor_menor: Valor mínimo esperado na variável (int ou float)
    :param valor_maior: Valor máximo esperado na variável (int ou float)
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_range(dataframe, 'nome_variavel', valor_menor, valor_maior)
```

```

"""
log_tela.info('Verificando Range da variável: ' + variavel)
# Obs: Registros inválidos: None (equivalente a NA)
nulos = df[variavel].isnull().sum()
nulos = nulos if nulos else 0
log_output.info('\n\n' +
    ' - ' + 'Mínimo esperado: ' + str(valor_menor) + '\n' +
    ' - ' + 'Máximo esperado: ' + str(valor_maior) + '\n' +
    ' - ' + 'Total de registros: ' + str(len(df[variavel])) +
    '\n' +
    ' - ' + 'Registros nulos (NA): ' +
    str(df[variavel].isnull().sum()) + '\n'
)
df_erro = df[(df[variavel] < valor_menor) | (df[variavel] > valor_maior)]

if len(df_erro[variavel].value_counts()) > 0:

    result = df_erro[variavel].value_counts().sort_index()
    # Verificando limite inferior
    if result.first_valid_index() < valor_menor:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro mínimo encontrado: ' +
            str(result.first_valid_index()) + " - abaixo do esperado!")
    # Verificando limite superior
    if result.last_valid_index() > valor_maior:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro máximo encontrado: ' +
            str(result.last_valid_index()) + " - acima do esperado!")

    log_tela.warning(variavel + ': ' +
        str(len(df_erro[variavel].value_counts())) +
        ' valor(es) incorreto(s)' +
        'encontrado(s) nesta variável:\n' +
        str(df_erro[variavel].value_counts()))
else:
    log_tela.info(variavel + ": Nenhum erro encontrado.")

def pre_processamento(df):
"""
Realiza algumas ações prévias ao processamento dos dados,
removendo alguns registros e ajustando o dataframe.
"""

log_tela.info('Criando/Renomeando colunas no dataframe principal')

log_output.info('Renomeando coluna UCOD para UCOD_DOM')
df.rename(columns={'UCOD': 'UCOD_DOM'}, inplace=True)

log_output.info('Renomeando coluna ANDA_CHEG para ANDA_DEST')
df.rename(columns={'ANDA_CHEG': 'ANDA_DEST'}, inplace=True)

log_tela.info('Verificando se todas as variáveis esperadas' +

```

```

        'existem no dataframe.\n' +
        'Caso não exista alguma, ela é criada.')
variaveis = ['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
             'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
             'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
             'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
             'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
             'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
             'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
             'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
             'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
             'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
             'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
             'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
             'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
             'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
             'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
             'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
             'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
             'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
             'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
             'VALOR_EST_AUTO']

for variavel in variaveis:
    if variavel not in df.columns:
        # Se a variável não existe no dataframe então ela é criada
        # com valor padrão de NONE (NA)
        df[variavel] = None
    log_tela.info('Criando a variavel ' + variavel + ' no dataframe')

log_output.info('Reordenando as variáveis')
df = df[['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
          'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
          'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
          'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
          'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
          'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
          'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
          'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
          'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
          'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
          'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
          'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
          'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
          'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
          'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
          'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
          'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
          'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
          'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
          'VALOR_EST_AUTO']]

log_output.info('\n\n=====\\n')
log_output.info('\\n\\n=====\\n')

```

```

    return df

def coordenadas(passo, df):
    """
    Itera sobre os registros do dataframe coords,
    a cada iteração, utiliza o valor das colunas ZONA e SUBZONA
    para filtrar o dataframe df em cada um dos tipos de ZONA E
    SUBZONA (DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui os valores
    das coordenadas X e Y de cada tipo (CO_DOM_X, CO_ESC_Y, etc) com
    os valores das colunas CO_X e CO_Y do dataframe coords.

    :param passo: número do passo para o log
    :param df: Dafaframe a ser modificado (ex.: od1997)
    :return: retorna o dataframe modificado com todas as coordenadas aplicadas
    """
    log_tela.info("### PASSO " + str(passo) + " - COORDENADAS")

    def coord_aux(row):
        """
        Esta função irá receber uma linha (row) do dataframe coords e
        utilizar os valores desta linha para realizar as alterações
        de coordenadas no dataframe df.
        """

        # Começando pelo tipo DOM, alterando CO_DOM_X e depois CO_DOM_Y
        df.loc[(df['ZONA_DOM'] == row['ZONA']) &
               (df['SUBZONA_DOM'] == row['SUBZONA']), 'CO_DOM_X'] = row['CO_X']
        df.loc[(df['ZONA_DOM'] == row['ZONA']) &
               (df['SUBZONA_DOM'] == row['SUBZONA']), 'CO_DOM_Y'] = row['CO_Y']

        # Agora com o tipo ESC, alterando CO_ESC_X e depois CO_ESC_Y
        df.loc[(df['ZONA_ESC'] == row['ZONA']) &
               (df['SUBZONA_ESC'] == row['SUBZONA']), 'CO_ESC_X'] = row['CO_X']
        df.loc[(df['ZONA_ESC'] == row['ZONA']) &
               (df['SUBZONA_ESC'] == row['SUBZONA']), 'CO_ESC_Y'] = row['CO_Y']

        # Agora com o tipo TRAB1, alterando CO_TRAB1_X e depois CO_TRAB1_Y
        df.loc[(df['ZONA_TRAB1'] == row['ZONA']) &
               (df['SUBZONA_TRAB1'] == row['SUBZONA']), 'CO_TRAB1_X'] = row['CO_X']
        df.loc[(df['ZONA_TRAB1'] == row['ZONA']) &
               (df['SUBZONA_TRAB1'] == row['SUBZONA']), 'CO_TRAB1_Y'] = row['CO_Y']

        # Agora com o tipo TRAB2, alterando CO_TRAB2_X e depois CO_TRAB2_Y
        df.loc[(df['ZONA_TRAB2'] == row['ZONA']) &
               (df['SUBZONA_TRAB2'] == row['SUBZONA']), 'CO_TRAB2_X'] = row['CO_X']
        df.loc[(df['ZONA_TRAB2'] == row['ZONA']) &
               (df['SUBZONA_TRAB2'] == row['SUBZONA']), 'CO_TRAB2_Y'] = row['CO_Y']

        # Agora com o tipo ORIG, alterando CO_ORIG_X e depois CO_ORIG_Y
        df.loc[(df['ZONA_ORIG'] == row['ZONA']) &
               (df['SUBZONA_ORIG'] == row['SUBZONA']), 'CO_ORIG_X'] = row['CO_X']
        df.loc[(df['ZONA_ORIG'] == row['ZONA']) &
               (df['SUBZONA_ORIG'] == row['SUBZONA']), 'CO_ORIG_Y'] = row['CO_Y']

        # Agora com o tipo DEST, alterando CO_DEST_X e depois CO_DEST_Y
        df.loc[(df['ZONA_DEST'] == row['ZONA']) &
               (df['SUBZONA_DEST'] == row['SUBZONA']), 'CO_DEST_X'] = row['CO_X']
        df.loc[(df['ZONA_DEST'] == row['ZONA']) &
               (df['SUBZONA_DEST'] == row['SUBZONA']), 'CO_DEST_Y'] = row['CO_Y']

```

```
(df['SUBZONA_DEST'] == row['SUBZONA']), 'CO_DEST_Y'] = row['CO_Y']

log_output.info('Lendo arquivo auxiliar de Coordenadas das subzonas')
coords = pd.read_csv('auxiliares/coord_subzonas_1997.csv', sep=';', decimal=',')

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = coords.apply(coord_aux, axis=1)

log_output.info('\n\n=====\\n')

return df
```

3.2 Funções Gerais

Função/Variável	Status
passo_ano	OK
passo_dia_sem	OK
passo_ucods	OK

```
In [ ]: log_tela.info('Definindo as funções gerais')
log_output.info('\n\n=====\\n')

def passo_ano(passo, df):
    """
    Preenche a coluna "ANO" com valor 3 em todas células
    Categorias:
    /valor/ano_correspondente/
    /-----/-----
    /1/1977/
    /2/1987/
    /3/1997/
    /4/2007/

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
    :return: retorna o dataframe modificado
    """

    log_tela.info("### PASSO " + str(passo) + " - ANO")

    # Definindo valor '3' para todas as células da coluna ANO
    df["ANO"] = 3

    return df

def passo_dia_sem(passo, df):
    """
    Apenas verificar os valores existentes
    # #####Categorias:
    # Valor/Descrição
    # -----/-----
    # 2/Segunda-Feira
    # 3/Terça-Feira
    # 4/Quarta-Feira
    # 5/Quinta-Feira
    # 6/Sexta-Feira

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
    :return: retorna o dataframe modificado
    """


```

```

log_tela.info("### PASSO " + str(passo) + " - DIA_SEM")

# Substituindo **0** por **None**
df.loc[df['DIA_SEM']==0,'DIA_SEM'] = None

# Verificando intervalo de valores - condições:
# "DIA_SEM >= 2" E "DIA_SEM <= 6"
verifica_range(df, 'DIA_SEM', 2, 6)
log_output.info('\n\n=====\\n')

return df

def passo_ucods(passo, df):
    """
    Itera sobre os registros do dataframe ucods,
    a cada iteração, utiliza o valor da coluna ZONA_REF
    para filtrar o dataframe df em cada um dos tipos de ZONA
    (DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui o valor
    da respectiva UCOD.

    :param passo: número do passo para o log
    :param df: Dataframe a ser modificado (ex.: od1997)
    :return: retorna o dataframe modificado com todas as UCODS aplicadas
    """
    log_tela.info("### PASSO " + str(passo) + " - UCODS")

    def ucod_aux(row):
        df.loc[df['ZONA_DOM']==row['ZONA_REF'], 'UCOD_DOM'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_ESC']==row['ZONA_REF'], 'UCOD_ESC'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_TRAB1']==row['ZONA_REF'], 'UCOD_TRAB1'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_TRAB2']==row['ZONA_REF'], 'UCOD_TRAB2'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_ORIG']==row['ZONA_REF'], 'UCOD_ORIG'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_DEST']==row['ZONA_REF'], 'UCOD_DEST'] = row['UCOD_BUSCADA']

    log_output.info('Lendo arquivo auxiliar UCOD')
    ucods = pd.read_csv('auxiliares/UCOD-1997.csv', sep=';', decimal=',')

    # Esta variável out não é utilizada para nada além de evitar um
    # monte de output que não será utilizado e que é gerado pelo método apply.
    out = ucods.apply(ucod_aux, axis=1)

    # Verificando intervalo de valores - condições:
    # "UCOD_XXX >= 1" E "UCOD_XXX <= 67"
    for tipo in ['DOM', 'ESC', 'TRAB1', 'TRAB2', 'ORIG', 'DEST']:
        verifica_range(df, 'UCOD_' + tipo, 1, 67)
    log_output.info('\n\n=====\\n')

    return df

```

3.3 Funções do Domicílio

Função/Variável	Status
passo_zona_dom	OK
passo_subzona_dom	OK
passo_mun_dom	OK
passo_f_dom	OK
passo_fe_dom	OK
passo_tipo_dom	OK

```
In [ ]: log_tela.info('Definindo as funções do domicílio')
log_output.info('\n\n=====\\n')

def passo_zona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 389

    [Teste: Checar se existe algum número < 1 ou > 389.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_DOM")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_DOM']==0,'ZONA_DOM'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_DOM >= 1" E "ZONA_DOM <= 389"
    verifica_range(df, 'ZONA_DOM', 1, 389)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
```

```

:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DOM")

# Substituindo valor 0 por None
df.loc[df['SUBZONA_DOM']==0,'SUBZONA_DOM'] = None

# Verificando intervalo de valores - condições:
# "SUBZONA_DOM >= 1" E "SUBZONA_DOM <= 9"
verifica_range(df, 'SUBZONA_DOM', 1, 9)
log_output.info('\n\n=====\\n')

return df


def passo_mun_dom(passo, df):
"""
Checar se existe algum erro

# #####Categorias
# > 1 a 39

[Teste: Checar se existe algum número < 1 ou > 39.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MUN_DOM")

# Substituindo valor 0 por None
df.loc[df['MUN_DOM']==0,'MUN_DOM'] = None

# Verificando intervalo de valores - condições:
# "MUN_DOM >= 1" E "MUN_DOM <= 39"
verifica_range(df, 'MUN_DOM', 1, 39)
log_output.info('\n\n=====\\n')

return df


def passo_f_dom(passo, df):
"""
Checar se existe algum erro na coluna "F_DOM"

# #####Categorias
# Valor/Descrição
# ----/----
# 0/Demais registros
# 1/Primeiro Registro do Domicílio

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]

```

```

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - F_DOM")

verifica_dummy(df, 'F_DOM')
log_output.info('\n\n=====\\n')

return df

def passo_fe_dom(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_DOM"

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_DOM")
    log_output.info('\n\n=====\\n')

    return df

def passo_tipo_dom(passo, df):
    """
    * Substituir **0** por **None (NA)**
    * Substituir **3** por **2**
    * Substituir **2** por **0**

    # #####Categorias anteriores / novas
    # Valor / Descrição
    # ----/----
    # 1/Particular
    # 2/Coletivo
    # 3/Favela

    # #####Categorias novas
    # Valor / Descrição
    # ----/----
    # 0/Coletivo
    # 1/Particular

    [Teste: Checar se existe algum número < 0 ou > 1.
     Se encontrar, retornar erro indicando em qual linha.]"""

    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - TIPO_DOM")

    df.loc[df['TIPO_DOM']==0,'TIPO_DOM'] = None

```

```
df.loc[df['TIPO_DOM']==3,'TIPO_DOM'] = 2
df.loc[df['TIPO_DOM']==2,'TIPO_DOM'] = 0

# Verificando intervalo de valores - condições:
    # "TIPO_DOM >= 0" E "TIPO_DOM <= 1"
verifica_dummy(df, 'TIPO_DOM')
log_output.info('\'\n\'=====\\n')

return df
```

3.4 Funções da Família

Função/Variável	Status
passo_tot_fam	OK
passo_f_fam	OK
passo_fe_fam	OK
passo_cond_mora	OK
passo_qt_auto	OK
passo_qt_bici	OK
passo_qt_moto	OK
passo_ren_fam	OK
passo_cd_renfam	OK

```
In [ ]: log_tela.info('Definindo as funções da família')
log_output.info('\n\n=====\\n')

def passo_tot_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "TOT_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - TOT_FAM")
    log_output.info('\\n\\n=====\\n')

    return df

def passo_f_fam(passo, df):
    """
    Checar se existe algum erro na coluna "F_FAM"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Família

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_FAM")

    verifica_dummy(df, 'F_FAM')
```

```

log_output.info('`n`n=====`n')

return df

def passo_fe_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_FAM")
    log_output.info('`n`n=====`n')

    return df

def passo_cond_mora(passo, df):
    """
    Substituir valores da coluna "COND_MORA"

    * Substituir todos valores **5** por **0**
    * Substituir todos valores **0** por **None**
    * Substituir todos valores **4** por **3**

    ##### Categorias anteriores

    Valor/Descrição
    ----/----
    1/Alugada
    2/Própria
    3/Cedida
    4/Outros
    5/Não respondeu

    ##### Categorias novas

    Valor/Descrição
    ----/----
    1/Alugada
    2/Própria
    3/Outros

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: dataframe modificado
    """

    log_tela.info("### PASSO " + str(passo) + " - COND_MORA")

    # Substituindo valor 5 por 0
    df.loc[df['COND_MORA']==5,'COND_MORA'] = 0

```

```

# Substituindo valor 0 por None
df.loc[df['COND_MORA']==0,'COND_MORA'] = None
# Substituindo valor 4 por 3
df.loc[df['COND_MORA']==4,'COND_MORA'] = 3

# Verificando intervalo de valores - condições:
# "COND_MORA >= 1" E "COND_MORA <= 3"
verifica_range(df, 'COND_MORA', 1, 3)
log_output.info('\n\n=====\\n')

return df


def passo_qt_auto(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "QT_AUTO"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - QT_AUTO")
    log_output.info('\n\n=====\\n')

    return df


def passo_qt_bici(passo, df):
    """
    Não existe essa informação no banco de dados de 1997, logo,
    este campo será preenchido com 'None'.
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    df['QT_BICI'] = None

    log_tela.info('### PASSO ' + str(passo) + ' - QT_BICI')
    log_output.info('\n\n=====\\n')

    return df


def passo_qt_moto(passo, df):
    """
    Não existe essa informação no banco de dados de 1997, logo,
    este campo será preenchido com 'None'.
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    df['QT_MOTO'] = None

    log_tela.info('### PASSO ' + str(passo) + ' - QT_MOTO')
    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_ren_fam(passo, df, deflator):
    """
    Corrigie a renda familiar de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_FAM")
    df['REN_FAM'] = df['REN_FAM'] * deflator
    return df

def passo_cd_renfam(passo, df):
    """
    Substituir valores da coluna "CD_RENFAM"

    * Substituir todos valores **2** por **0**
    * Substituir todos valores **3** por **2**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Renda familiar completa
    2/Não tem renda
    3/Renda familiar incompleta

    ##### Categorias novas
    Valor/Descrição
    ----/----
    0/Renda Familiar Declarada como Zero
    1/Renda Familiar Declarada e Maior que Zero
    2/Renda Atribuída

    [Teste: Checar se existe algum número < 0 ou > 2.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe corrigido
    """
    log_tela.info("### PASSO " + str(passo) + " - CD_RENFAM")

    # Substituindo valor 2 por 0
    df.loc[df['CD_RENFAM']==2, 'CD_RENFAM'] = 0
    # Substituindo valor 3 por 2
    df.loc[df['CD_RENFAM']==3, 'CD_RENFAM'] = 2

    # Dividindo a categoria '0', "Respondeu", em:

```

```
#     - 0 "Renda Familiar Declarada como Zero" e
#     - 1 "Renda Familiar Declarada e Maior que Zero"
df.loc[(df['CD_RENFAM'] == 0) &
       (df['REN_FAM'] != 0) &
       (df['REN_FAM'].notnull()), 'CD_RENFAM'] = 1

# Verificando intervalo de valores - condições:
# "CD_RENFAM >= 0" E "CD_RENFAM <= 2"
verifica_range(df, 'CD_RENFAM', 0, 2)
log_output.info('\n\n=====\\n')

return df
```

3.5 Funções da pessoa

Função/Variável	Status
passo_f_pess	OK
passo_fe_pess	OK
passo_sit_fam	OK
passo_idade	OK
passo_sexo	OK
passo_grau_instr	OK
passo_ocup	OK
passo_setor_ativ	OK
passo_ren_ind	OK
passo_cd_renind	OK
passo_estuda	Verificar

Obs.: Se ESTUDA for ser alterado pela verificação do ZONA_ESC, no main() ele deverá ser chamado após ZONA_ESC. Atualmente ele já está localizado após o ZONA_ESC na chamada do main(). Após confirmar o que será feito com ele, se for necessário alterar sua localização.

```
In [ ]: log_tela.info('Definindo as funções da pessoa')
log_output.info('\n\n=====\\n')

def passo_f_pess(passo, df):
    """
    Checar se existe algum erro na coluna "F_PESS"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Pessoa

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_PESS")

    verifica_dummy(df, 'F_PESS')
    log_output.info('\n\n=====\\n')

    return df

def passo_fe_pess(passo, df):
```

```

"""
Nada há que se fazer em relação aos dados das colunas "FE_PESS"
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
log_tela.info("### PASSO " + str(passo) + " - FE_PESS")
log_output.info('\n\n=====\\n')

return df

def passo_sit_fam(passo, df):
    """
    Não é preciso realizar nenhuma alteração

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Chefe
    2/Cônjuge
    3/Filho(a)
    4/Parente / Agregado
    5/Empregado Residente
    6/Visitante não residente na RMSP

    ##### Categorias novas:
    Valor/Descrição
    ----/----
    1/ Pessoa Responsável
    2/ Cônjuge/Companheiro(a)
    3/ Filho(a)/Enteado(a)
    4/ Outro Parente / Agregado
    5/ Empregado Residente
    6/ Outros (visitante não residente / parente do empregado)

    [Teste: Checar se existe algum número < 1 ou > 6.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SIT_FAM")

    # Verificando intervalo de valores - condições:
    # "SIT_FAM >= 1" E "SIT_FAM <= 6"
    verifica_range(df, 'SIT_FAM', 1, 6)
    log_output.info('\n\n=====\\n')

    return df

def passo_idade(passo, df):
    """

```

```

Nada há que se fazer em relação aos dados da coluna "IDADE"
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
log_tela.info("### PASSO " + str(passo) + " - IDADE")
log_output.info('\n\n=====\\n')

return df

def passo_sexo(passo, df):
    """
    # #####Categorias anteriores
    # Valor/Descrição
    # ----/----
    # 0/Não Respondeu (-> None)
    # 1/Masculino
    # 2/Feminino

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 0/Masculino
    # 1/Feminino

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: retorna o dataframe modificado

"""
log_tela.info("### PASSO " + str(passo) + " - SEXO")

Substituindo valor 0 por None
df.loc[df['SEXO']==0,'SEXO'] = None
Substituindo valor 1 por 0
df.loc[df['SEXO']==1,'SEXO'] = 0
Substituindo valor 2 por 1
df.loc[df['SEXO']==2,'SEXO'] = 1

Verificando intervalo de valores - condições:
"SEXO >= 0" E "SEXO <= 1"
verifica_dummy(df, 'SEXO')
log_output.info('\n\n=====\\n')

return df

```

def passo_grau_instr(passo, df):
    """
    Substituir valores da coluna "GRAU_INSTR"

    * Substituir todos valores **2** por **1**
```

```

* Substituir todos valores **3** por **1**
* Substituir todos valores **4** por **2**
* Substituir todos valores **5** por **2**
* Substituir todos valores **6** por **3**
* Substituir todos valores **7** por **3**
* Substituir todos valores **8** por **4**
* Substituir todos valores **0** por **None**

##### Categorias anteriores:
Valor/Descrição
----/----
1/Não-alfabetizado
2/Pré-escola
3/1º Grau incompleto
4/1º Grau completo
5/2º Grau incompleto
6/2º Grau completo
7/Superior Incompleto
8/Superior Completo

##### Categorias novas
Valor/Descrição
----/----
1/Não-Alfabetizado/Fundamental Incompleto
2/Fundamental Completo/Médio Incompleto
3/Médio Completo/Superior Incompleto
4/Superior completo

[Teste: Checar se existe algum número < 1 ou > 4.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - GRAU_INSTR")

# Substituindo valor 2 por 1
df.loc[df['GRAU_INSTR']==2, 'GRAU_INSTR'] = 1
# Substituindo valor 3 por 1
df.loc[df['GRAU_INSTR']==3, 'GRAU_INSTR'] = 1
# Substituindo valor 4 por 2
df.loc[df['GRAU_INSTR']==4, 'GRAU_INSTR'] = 2
# Substituindo valor 5 por 2
df.loc[df['GRAU_INSTR']==5, 'GRAU_INSTR'] = 2
# Substituindo valor 6 por 3
df.loc[df['GRAU_INSTR']==6, 'GRAU_INSTR'] = 3
# Substituindo valor 7 por 3
df.loc[df['GRAU_INSTR']==7, 'GRAU_INSTR'] = 3
# Substituindo valor 8 por 4
df.loc[df['GRAU_INSTR']==8, 'GRAU_INSTR'] = 4
# Substituindo valor 0 por None
df.loc[df['GRAU_INSTR']==0, 'GRAU_INSTR'] = None

# Verificando intervalo de valores - condições:

```

```

# "GRAU_INSTR >= 1" E "GRAU_INSTR <= 4"
verifica_range(df, 'GRAU_INSTR', 1, 4)
log_output.info('\n\n=====\\n')

return df

def passo_ocup(passo, df):
    """
    Substituir valores da coluna "OCUP"

    Substituir todos valores **2** por **1**
    Substituir todos valores **3** por **2**
    Substituir todos valores **5** por **3**
    Substituir todos valores **6** por **5**
    Substituir todos valores **7** por **6**
    Substituir todos valores **8** por **7**
    Substituir todos valores **0** por **None**

    #####Categorias anteriores:
    Valor/Descrição
    ----/----
    1/Ocupado
    2/Ocupado eventualmente
    3/Em licença
    4/Não ocupada
    5/Aposentado/Pensionista
    6/Nunca trabalhou
    7/Dona de casa
    8/Estudante

    #####Categorias novas
    Valor/Descrição
    ----/----
    1/Tem trabalho
    2/Em licença médica
    3/Aposentado / pensionista
    4/Desempregado
    5/Sem ocupação
    6/Dona de casa
    7/Estudante

[Teste: Checar se existe algum número < 1 ou > 7.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - OCUP")

# Substituindo valor 2 por 1
df.loc[df['OCUP']==2,'OCUP'] = 1
# Substituindo valor 3 por 2
df.loc[df['OCUP']==3,'OCUP'] = 2
# Substituindo valor 5 por 3

```

```

df.loc[df['OCUP']==5,'OCUP'] = 3
# Substituindo valor 6 por 5
df.loc[df['OCUP']==6,'OCUP'] = 5
# Substituindo valor 7 por 6
df.loc[df['OCUP']==7,'OCUP'] = 6
# Substituindo valor 8 por 7
df.loc[df['OCUP']==8,'OCUP'] = 7
# Substituindo valor 0 por None
df.loc[df['OCUP']==0,'OCUP'] = None

# Verificando intervalo de valores - condições:
# "OCUP >= 1" E "OCUP <= 7"
verifica_range(df, 'OCUP', 1, 7)
log_output.info('\n\n=====\\n')

return df

def passo_setor_ativ(passo, df):
    """
    Substituir valores da coluna "SETOR_ATIV"

    Na coluna "SETOR_ATIV", linha i,
        ler o valor da linha i da coluna "SETOR_ATIV", daí,
        buscar o mesmo valor na coluna "COD" do arquivo setor_ativ-1997.csv.
    Ao achar, retornar o valor da mesma linha, só que da coluna "COD_UNIF"

    #####Categorias anteriores
    > ver arquivo .csv

    #####Categorias novas
    Valor/Descrição
    ----/----
    1/Agrícola
    2/Construção Civil
    3/Indústria
    4/Comércio
    5/Administração Pública
    6/Serviços de Transporte
    7/Outros serviços
    8/Doutros
    9/Não se aplica

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - SETOR_ATIV")

log_output.info('Lendo arquivo de referência externa setor_ativ-1997.csv')
df_setor = pd.read_csv('auxiliares/setor_ativ-1997.csv', sep=';', decimal=',')

```

```

def setor_aux(row):
    df.loc[df['SETOR_ATIV']==row['COD'], 'SETOR_ATIV'] = row['COD_UNIF']

    # Esta variável out não é utilizada para nada além de evitar um
    # monte de output que não será utilizado e que é gerado pelo método apply.
    out = df_setor.apply(setor_aux, axis=1)

    # Substituindo valor 0 por None
    df.loc[df['SETOR_ATIV']==0, 'SETOR_ATIV'] = None

    # Verificando intervalo de valores - condições:
    # "SETOR_ATIV >= 1" E "SETOR_ATIV <= 9"
    verifica_range(df, 'SETOR_ATIV', 1, 9)
    log_output.info('\n\n=====\\n')

    return df


def passo_ren_ind(passo, df, deflator):
    """
    Corriga a renda individual de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_IND")
    df['REN_IND'] = df['REN_IND'] * deflator

    return df


def passo_cd_renid(passo, df):
    """
    Substituir valores da coluna "CD_RENIND"

    * Substituir todos valores **3** por None
    * Substituir todos valores **2** por **0**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Tem renda
    2/Não tem renda
    3/Não respondeu

    ##### Categorias novas
    Valor/Descrição
    ----/-----
    0    /Não tem renda
    1    /Tem renda
    """

    [Teste: Checar se existe algum número < 0 ou > 1.

```

```

Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - CD_RENIND")

df.loc[df['CD_RENIND']==3,'CD_RENIND'] = None
df.loc[df['CD_RENIND']==2,'CD_RENIND'] = 0

# Verificando intervalo de valores - condições:
# "CD_RENIND >= 0" E "CD_RENIND <= 1"
verifica_dummy(df, 'CD_RENIND')
log_output.info('\n\n=====\\n')

return df

def passo_estuda(passo, df):
"""
Se zona da escola for zero (0) então não estuda (0), senão, estuda (1)

#####
ATENÇÃO
ESTA FUNÇÃO SÓ DEVE SER #
EXECUTADA APÓS A GERAÇÃO#
DE TODOS OS ID'S, POIS #
HÁ UMA DESCONFIANÇA #
QUANTO À QUALIDADE #
DESTE DADO #
#####

* Substituir todos valores **1** por **0**
* Substituir todos valores **2** por **1**
* Substituir todos valores **3** por **1**
* Substituir todos valores **4** por **1**

#### Categorias anteriores
Valor/Descrição
----/----
1/Não
2/Creche/Pré-Escola
3/1º Grau/2º Grau/3º Graus
4/Outros

#### Categorias novas
Valor/Descrição
----/----
0/Não estuda
1/Estuda

[Teste: Checar se existe algum número diferente de 0 ou 1.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log

```

```

:param df:
:return: retorna dataframe com campo ESCOLA resolvido
"""

log_tela.info("### PASSO " + str(passo) + " - ESTUDA")

# Substitui 1 por 0
df.loc[df['ESTUDA'] == 1, 'ESTUDA'] = 0
# Substitui 2 por 1
df.loc[df['ESTUDA'] == 2, 'ESTUDA'] = 1
# Substitui 3 por 1
df.loc[df['ESTUDA'] == 3, 'ESTUDA'] = 1
# Substitui 4 por 1
df.loc[df['ESTUDA'] == 4, 'ESTUDA'] = 1

# Substituindo todos que declararam zona escola diferente de zero
# com campo ESTUDA igual a 1.
df.loc[(df['ZONA_ESC'] != 0)&
       (df['ZONA_ESC'].notnull()), 'ESTUDA'] = 1

verifica_dummy(df, 'ESTUDA')
log_output.info('\'\n\'=====\\n')

return df

```

3.6 Funções referentes da Viagem

Função/Variável	Status
passo_f_vitag	OK
passo_fe_vitag	OK
passo_zona_esc	OK
passo_subzona_esc	OK
passo_mun_esc	OK
passo_zona_trab1	OK
passo_subzona_trab1	OK
passo_mun_trab1	OK
passo_zona_trab2	OK
passo_subzona_trab2	OK
passo_mun_trab2	OK
passo_zona_orig	OK
passo_subzona_orig	OK
passo_mun_orig	OK
passo_zona_dest	OK
passo_subzona_dest	OK
passo_mun_dest	OK
passo_serv_pas_orig	OK
passo_serv_pas_dest	OK
passo_motivo_orig	OK
passo_motivo_dest	OK
passo_modo1	OK
passo_modo2	OK
passo_modo3	OK
passo_modo4	OK
passo_modo_prin	OK
passo_tipo_est_auto	OK
passo_valor_est_auto	OK
passo_tot_vitag	OK

Obs: O passo_tot_vitag só deve ser executado após a produção dos ID's e NO's.

```
In [ ]: log_tela.info('Definindo funções referentes às viagens')
log_output.info('\n=====\\n=====\\n')
```

```
def passo_f_vitag(passo, df):
```

```

"""
Checar se existe algum erro na coluna "F_VIAG"

# #####Categorias
# Valor/Descrição
# ----/----
# 0/Não fez viagem
# 1/Fez viagem

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com F_VIAG modificado
"""

log_tela.info("### PASSO " + str(passo) + " - F_VIAG")

# Setando F_VIAG=1 para todos os casos
df['F_VIAG'] = 1

# Setando F_VIAG=0 se DURACAO==0
df.loc[df['DURACAO']==0, 'F_VIAG'] = 0

verifica_dummy(df, 'F_VIAG')
log_output.info('\n\n=====\\n')

return df


def passo_fe_viag(passo, df):
"""
# Nada há que se fazer em relação aos dados da coluna "FE_VIAG"
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - FE_VIAG")
log_output.info('\n\n=====\\n')

return df


def passo_zona_esc(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 389

[Teste: Checar se existe algum número < 1 ou > 389.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: Sem retorno
"""

```

```

"""
log_tela.info("### PASSO " + str(passo) + " - ZONA_ESC")

# Substituindo valor 0 por None
df.loc[df['ZONA_ESC']==0,'ZONA_ESC'] = None

# Verificando intervalo de valores - condições:
# "ZONA_ESC >= 1" E "ZONA_ESC <= 389"
verifica_range(df, 'ZONA_ESC', 1, 389)
log_output.info('\n\n=====\\n')

return df
"""

def passo_subzona_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ESC")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_ESC']==0,'SUBZONA_ESC'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_ESC >= 1" E "SUBZONA_ESC <= 9"
    verifica_range(df, 'SUBZONA_ESC', 1, 9)
    log_output.info('\n\n=====\\n')

    return df

def passo_mun_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ESC")

```

```

# Substituindo valor 0 por None
df.loc[df['MUN_ESC']==0,'MUN_ESC'] = None

# Verificando intervalo de valores - condições:
# "MUN_ESC >= 1" E "MUN_ESC <= 39"
verifica_range(df, 'MUN_ESC', 1, 39)
log_output.info('\n\n=====\\n')

return df


def passo_zona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 389

    [Teste: Checar se existe algum número < 1 ou > 389.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB1']==0,'ZONA_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_TRAB1 >= 1" E "ZONA_TRAB1 <= 389"
    verifica_range(df, 'ZONA_TRAB1', 1, 389)
    log_output.info('\\n\\n=====\\n')

    return df


def passo_subzona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB1")

    # Substituindo valor 0 por None

```

```

df.loc[df['SUBZONA_TRAB1']==0, 'SUBZONA_TRAB1'] = None

# Verificando intervalo de valores - condições:
# "SUBZONA_TRAB1 >= 1" E "SUBZONA_TRAB1 <= 9"
verifica_range(df, 'SUBZONA_TRAB1', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_mun_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB1']==0, 'MUN_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB1 >= 1" E "MUN_TRAB1 <= 39"
    verifica_range(df, 'MUN_TRAB1', 1, 39)
    log_output.info('\n\n=====\\n')

    return df

def passo_zona_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 389

    [Teste: Checar se existe algum número < 1 ou > 389.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB2']==0, 'ZONA_TRAB2'] = None

```

```

# Verificando intervalo de valores - condições:
# "ZONA_TRAB2 >= 1" E "ZONA_TRAB2 <= 389"
verifica_range(df, 'ZONA_TRAB2', 1, 389)
log_output.info('\'\n\'=====\'\n')

return df

def passo_subzona_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_TRAB2']==0, 'SUBZONA_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_TRAB2 >= 1" E "SUBZONA_TRAB2 <= 9"
    verifica_range(df, 'SUBZONA_TRAB2', 1, 9)
    log_output.info('\'\n\'=====\'\n')

    return df

def passo_mun_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB2']==0, 'MUN_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB2 >= 1" E "MUN_TRAB2 <= 39"

```

```

verifica_range(df, 'MUN_TRAB2', 1, 39)
log_output.info('\n\n=====\\n')

return df

def passo_zona_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 389

    [Teste: Checar se existe algum número < 1 ou > 389.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_ORIG']==0, 'ZONA_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_ORIG >= 1" E "ZONA_ORIG <= 389"
    verifica_range(df, 'ZONA_ORIG', 1, 389)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_ORIG']==0, 'SUBZONA_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_ORIG >= 1" E "SUBZONA_ORIG <= 9"
    verifica_range(df, 'SUBZONA_ORIG', 1, 9)
    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_mun_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['MUN_ORIG']==0,'MUN_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_ORIG >= 1" E "MUN_ORIG <= 39"
    verifica_range(df, 'MUN_ORIG', 1, 39)
    log_output.info('\n\n=====\\n')

    return df

def passo_zona_dest(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 389

    [Teste: Checar se existe algum número < 1 ou > 389.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_DEST")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_DEST']==0,'ZONA_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_DEST >= 1" E "ZONA_DEST <= 389"
    verifica_range(df, 'ZONA_DEST', 1, 389)
    log_output.info('\n\n=====\\n')

    return df

```

```

def passo_subzona_dest(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 9

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DEST")

    # Substituindo valor 0 por None
    df.loc[df['SUBZONA_DEST']==0,'SUBZONA_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "SUBZONA_DEST >= 1" E "SUBZONA_DEST <= 9"
    verifica_range(df, 'SUBZONA_DEST', 1, 9)
    log_output.info('\n\n=====\\n')

    return df


def passo_mun_dest(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_DEST")

    # Substituindo valor 0 por None
    df.loc[df['MUN_DEST']==0,'MUN_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_DEST >= 1" E "MUN_DEST <= 39"
    verifica_range(df, 'MUN_DEST', 1, 39)
    log_output.info('\n\n=====\\n')

    return df


def passo_serv_pas_orig(passo, df):

```

```

"""
Substituir **0** por None
Substituir **2** por **0**

####Categorias antigas
Valor/Descrição
----/----
0/Não Respondido
1/Sim
2/Não

####Categorias novas
Valor/Descrição
----/----
0/Não
1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_ORIG")

df.loc[df['SERV_PAS_ORIG']==0,'SERV_PAS_ORIG'] = None
df.loc[df['SERV_PAS_ORIG']==2,'SERV_PAS_ORIG'] = 0

verifica_dummy(df, 'SERV_PAS_ORIG')
log_output.info('\n\n=====\\n')

return df


def passo_serv_pas_dest(passo, df):
"""
Substituir **0** por None
Substituir **2** por **0**

####Categorias antigas
Valor/Descrição
----/----
0/Não Respondido
1/Sim
2/Não

####Categorias novas
Valor/Descrição
----/----
0/Não
1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""

```

```

log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_DEST")

df.loc[df['SERV_PAS_DEST']==0, 'SERV_PAS_DEST'] = None
df.loc[df['SERV_PAS_DEST']==2, 'SERV_PAS_DEST'] = 0

verifica_dummy(df, 'SERV_PAS_DEST')
log_output.info('\n\n=====\\n')

return df

def passo_motivo_orig(passo, df):
    """
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Trabalho/Indústria
    2/Trabalho/Comércio
    3/Trabalho/Serviços
    4/Escola/Educação
    5/Compras
    6/Médico/Dentista/Saúde
    7/Recreação/Visitas
    8/Residência
    9/Outros

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Trabalho/Indústria
    2/Trabalho/Comércio
    3/Trabalho/Serviços
    4/Educação
    5/Compras
    6/Saúde
    7/Lazer
    8/Residência
    9/Outros

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - MOTIVO_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['MOTIVO_ORIG']==0, 'MOTIVO_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "MOTIVO_ORIG >= 1" E "MOTIVO_ORIG <= 9"

```

```

verifica_range(df, 'MOTIVO_ORIG', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_motivo_dest(passo, df):
    """
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Trabalho/Indústria
    2/Trabalho/Comércio
    3/Trabalho/Serviços
    4/Escola/Educação
    5/Compras
    6/Médico/Dentista/Saúde
    7/Recreação/Visitas
    8/Residência
    9/Outros

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Trabalho/Indústria
    2/Trabalho/Comércio
    3/Trabalho/Serviços
    4/Educação
    5/Compras
    6/Saúde
    7/Lazer
    8/Residência
    9/Outros

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - MOTIVO_DEST")

    # Substituindo valor 0 por None
    df.loc[df['MOTIVO_DEST']==0, 'MOTIVO_DEST'] = None

    # Verificando intervalo de valores - condições:
    # "MOTIVO_DEST >= 1" E "MOTIVO_DEST <= 9"
    verifica_range(df, 'MOTIVO_DEST', 1, 9)
    log_output.info('\n\n=====\\n')

return df

```

```

def passo_modo1(passo, df):
    """
    Substituir valores da coluna "MODO1"

    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **3**
    * Substituir todos valores **5** por **4**
    * Substituir todos valores **6** por **5**
    * Substituir todos valores **7** por **6**
    * Substituir todos valores **8** por **7**
    * Substituir todos valores **9** por **8**
    * Substituir todos valores **10** por **9**
    * Substituir todos valores **11** por **10**
    * Substituir todos valores **12** por **11**
    * Substituir todos valores **13** por **12**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Ônibus
    2/Ônibus Fretado
    3/Transporte Escolar
    4/Dirigindo Automóvel
    5/Passageiro de Automóvel
    6/Táxi
    7/Lotação/Perua
    8/Metrô
    9/Trem
    10/Moto
    11/Bicicleta
    12/A Pé
    13/Outros

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Ônibus
    2/Ônibus Escolar / Empresa
    3/Dirigindo Automóvel
    4/Passageiro de Automóvel
    5/Táxi
    6/Lotação / Perua / Van / Microônibus
    7/Metrô
    8/Trem
    9/Moto
    10/Bicicleta
    11/A Pé
    12/Outros

    [Teste: Checar se existe algum número < 1 ou > 12.
     Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

```

:param df:
:return: sem retorno
"""

log_tela.info("### PASSO " + str(passo) + " - MODO1")

df.loc[df['MODO1']==3,'MODO1'] = 2
df.loc[df['MODO1']==4,'MODO1'] = 3
df.loc[df['MODO1']==5,'MODO1'] = 4
df.loc[df['MODO1']==6,'MODO1'] = 5
df.loc[df['MODO1']==7,'MODO1'] = 6
df.loc[df['MODO1']==8,'MODO1'] = 7
df.loc[df['MODO1']==9,'MODO1'] = 8
df.loc[df['MODO1']==10,'MODO1'] = 9
df.loc[df['MODO1']==11,'MODO1'] = 10
df.loc[df['MODO1']==12,'MODO1'] = 11
df.loc[df['MODO1']==13,'MODO1'] = 12
df.loc[df['MODO1']==0,'MODO1'] = None

# Verificando intervalo de valores - condições:
# "MODO1 >= 1" E "MODO1 <= 12"
verifica_range(df, 'MODO1', 1, 12)
log_output.info('\n\n=====\\n')

return df

def passo_modo2(passo, df):
"""
Substituir valores da coluna "MODO2"

* Substituir todos valores **3** por **2**
* Substituir todos valores **4** por **3**
* Substituir todos valores **5** por **4**
* Substituir todos valores **6** por **5**
* Substituir todos valores **7** por **6**
* Substituir todos valores **8** por **7**
* Substituir todos valores **9** por **8**
* Substituir todos valores **10** por **9**
* Substituir todos valores **11** por **10**
* Substituir todos valores **12** por **11**
* Substituir todos valores **13** por **12**
* Substituir todos valores **0** por **None**

#### Categorias anteriores
Valor/Descrição
----/----
1/Ônibus
2/Ônibus Fretado
3/Transporte Escolar
4/Dirigindo Automóvel
5/Passageiro de Automóvel
6/Táxi
7/Lotação/Perua
8/Metrô

```

```

9/Trem
10/Moto
11/Bicicleta
12/A Pé
13/Outros

#### Categorias novas
Valor/Descrição
----/----
1/Ônibus
2/Ônibus Escolar / Empresa
3/Dirigindo Automóvel
4/Passageiro de Automóvel
5/Táxi
6/Lotação / Perua / Van / Microônibus
7/Metrô
8/Trem
9/Moto
10/Bicicleta
11/A Pé
12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - MOD03")

df.loc[df['MOD02']==3,'MOD02'] = 2
df.loc[df['MOD02']==4,'MOD02'] = 3
df.loc[df['MOD02']==5,'MOD02'] = 4
df.loc[df['MOD02']==6,'MOD02'] = 5
df.loc[df['MOD02']==7,'MOD02'] = 6
df.loc[df['MOD02']==8,'MOD02'] = 7
df.loc[df['MOD02']==9,'MOD02'] = 8
df.loc[df['MOD02']==10,'MOD02'] = 9
df.loc[df['MOD02']==11,'MOD02'] = 10
df.loc[df['MOD02']==12,'MOD02'] = 11
df.loc[df['MOD02']==13,'MOD02'] = 12
df.loc[df['MOD02']==0,'MOD02'] = None

# Verificando intervalo de valores - condições:
# "MOD02 >= 1" E "MOD02 <= 12"
verifica_range(df, 'MOD02', 1, 12)
log_output.info('\'\n\'-----\'\n\'-----\'\n')

return df

def passo_modo3(passo, df):
"""
Substituir valores da coluna "MOD03"

```

```
* Substituir todos valores **3** por **2**
* Substituir todos valores **4** por **3**
* Substituir todos valores **5** por **4**
* Substituir todos valores **6** por **5**
* Substituir todos valores **7** por **6**
* Substituir todos valores **8** por **7**
* Substituir todos valores **9** por **8**
* Substituir todos valores **10** por **9**
* Substituir todos valores **11** por **10**
* Substituir todos valores **12** por **11**
* Substituir todos valores **13** por **12**
* Substituir todos valores **0** por **None**
```

Categorias anteriores

Valor/Descrição

----/----

- 1/Ônibus
- 2/Ônibus Fretado
- 3/Transporte Escolar
- 4/Dirigindo Automóvel
- 5/Passageiro de Automóvel
- 6/Táxi
- 7/Lotação/Perua
- 8/Metrô
- 9/Trem
- 10/Moto
- 11/Bicicleta
- 12/A Pé
- 13/Outros

Categorias novas

Valor/Descrição

----/----

- 1/Ônibus
- 2/Ônibus Escolar / Empresa
- 3/Dirigindo Automóvel
- 4/Passageiro de Automóvel
- 5/Táxi
- 6/Lotação / Perua / Van / Microônibus
- 7/Metrô
- 8/Trem
- 9/Moto
- 10/Bicicleta
- 11/A Pé
- 12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.

Se encontrar, retornar erro indicando em qual linha.]

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```
log_tela.info("### PASSO " + str(passo) + " - MOD03")
```

```

df.loc[df['MODO3']==3, 'MODO3'] = 2
df.loc[df['MODO3']==4, 'MODO3'] = 3
df.loc[df['MODO3']==5, 'MODO3'] = 4
df.loc[df['MODO3']==6, 'MODO3'] = 5
df.loc[df['MODO3']==7, 'MODO3'] = 6
df.loc[df['MODO3']==8, 'MODO3'] = 7
df.loc[df['MODO3']==9, 'MODO3'] = 8
df.loc[df['MODO3']==10, 'MODO3'] = 9
df.loc[df['MODO3']==11, 'MODO3'] = 10
df.loc[df['MODO3']==12, 'MODO3'] = 11
df.loc[df['MODO3']==13, 'MODO3'] = 12
df.loc[df['MODO3']==0, 'MODO3'] = None

# Verificando intervalo de valores - condições:
# "MODO3 >= 1" E "MODO3 <= 12"
verifica_range(df, 'MODO3', 1, 12)
log_output.info('\'\n\'=====\\n')

return df

def passo_modo4(passo, df):
    """
    Substituir valores da coluna "MODO4"

    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **3**
    * Substituir todos valores **5** por **4**
    * Substituir todos valores **6** por **5**
    * Substituir todos valores **7** por **6**
    * Substituir todos valores **8** por **7**
    * Substituir todos valores **9** por **8**
    * Substituir todos valores **10** por **9**
    * Substituir todos valores **11** por **10**
    * Substituir todos valores **12** por **11**
    * Substituir todos valores **13** por **12**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Ônibus
    2/Ônibus Fretado
    3/Transporte Escolar
    4/Dirigindo Automóvel
    5/Passageiro de Automóvel
    6/Táxi
    7/Lotação/Perua
    8/Metrô
    9/Trem
    10/Moto
    11/Bicicleta
    12/A Pé

```

13/Outros

```
#### Categorias novas  
Valor/Descrição  
----/----  
1/Ônibus  
2/Ônibus Escolar / Empresa  
3/Dirigindo Automóvel  
4/Passageiro de Automóvel  
5/Táxi  
6/Lotação / Peruá / Van / Microônibus  
7/Metrô  
8/Trem  
9/Moto  
10/Bicicleta  
11/A Pé  
12/Outros
```

[Teste: Checar se existe algum número < 1 ou > 12.

Se encontrar, retornar erro indicando em qual linha.]

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

```
log_tela.info("### PASSO " + str(passo) + " - MODO4")
```

```
df.loc[df['MODO4']==3,'MODO4'] = 2  
df.loc[df['MODO4']==4,'MODO4'] = 3  
df.loc[df['MODO4']==5,'MODO4'] = 4  
df.loc[df['MODO4']==6,'MODO4'] = 5  
df.loc[df['MODO4']==7,'MODO4'] = 6  
df.loc[df['MODO4']==8,'MODO4'] = 7  
df.loc[df['MODO4']==9,'MODO4'] = 8  
df.loc[df['MODO4']==10,'MODO4'] = 9  
df.loc[df['MODO4']==11,'MODO4'] = 10  
df.loc[df['MODO4']==12,'MODO4'] = 11  
df.loc[df['MODO4']==13,'MODO4'] = 12  
df.loc[df['MODO4']==0,'MODO4'] = None
```

Verificando intervalo de valores - condições:

"MODO4 >= 1" E "MODO4 <= 12"

```
verifica_range(df, 'MODO4', 1, 12)
```

```
log_output.info('\n\n=====\\n')
```

```
return df
```

```
def passo_modo_prin(passo, df):  
    """
```

Substituir valores da coluna "MODO_PRIN"

```
* Substituir todos valores **3** por **2**  
* Substituir todos valores **4** por **3**  
* Substituir todos valores **5** por **4**
```

```

* Substituir todos valores **6** por **5**
* Substituir todos valores **7** por **6**
* Substituir todos valores **8** por **7**
* Substituir todos valores **9** por **8**
* Substituir todos valores **10** por **9**
* Substituir todos valores **11** por **10**
* Substituir todos valores **12** por **11**
* Substituir todos valores **13** por **12**
* Substituir todos valores **0** por **None**

```

Categorias anteriores

Valor/Descrição

----/----

1/*Ônibus*

2/*Ônibus Fretado*

3/*Transporte Escolar*

4/*Dirigindo Automóvel*

5/*Passageiro de Automóvel*

6/*Táxi*

7/*Lotação/Perua*

8/*Metrô*

9/*Trem*

10/*Moto*

11/*Bicicleta*

12/*A Pé*

13/*Outros*

Categorias novas

Valor/Descrição

----/----

1/*Ônibus*

2/*Ônibus Escolar / Empresa*

3/*Dirigindo Automóvel*

4/*Passageiro de Automóvel*

5/*Táxi*

6/*Lotação / Perua / Van / Microônibus*

7/*Metrô*

8/*Trem*

9/*Moto*

10/*Bicicleta*

11/*A Pé*

12/*Outros*

[Teste: Checar se existe algum número < 1 ou > 12.

Se encontrar, retornar erro indicando em qual linha.]

:param passo: Número do passo atual para registro/log

:param df:

:return: sem retorno

"""

log_tela.info("### PASSO " + str(passo) + " - MODO_PRIN")

df.loc[df['MODO_PRIN']==3,'MODO_PRIN'] = 2

df.loc[df['MODO_PRIN']==4,'MODO_PRIN'] = 3

df.loc[df['MODO_PRIN']==5,'MODO_PRIN'] = 4

```

df.loc[df['MODO_PRIN']==6, 'MODO_PRIN'] = 5
df.loc[df['MODO_PRIN']==7, 'MODO_PRIN'] = 6
df.loc[df['MODO_PRIN']==8, 'MODO_PRIN'] = 7
df.loc[df['MODO_PRIN']==9, 'MODO_PRIN'] = 8
df.loc[df['MODO_PRIN']==10, 'MODO_PRIN'] = 9
df.loc[df['MODO_PRIN']==11, 'MODO_PRIN'] = 10
df.loc[df['MODO_PRIN']==12, 'MODO_PRIN'] = 11
df.loc[df['MODO_PRIN']==13, 'MODO_PRIN'] = 12
df.loc[df['MODO_PRIN']==0, 'MODO_PRIN'] = None

# Verificando intervalo de valores - condições:
    # "MODO_PRIN >= 1" E "MODO_PRIN <= 12"
verifica_range(df, 'MODO_PRIN', 1, 12)
log_output.info('\n\n=====\\n')

return df

def passo_tipo_vitag(passo, df):
    """
* Substituir os valores **0** por **None**

# #####Categorias novas
# Valor/Descrição
# ----/----
# 1/Coletivo
# 2/Individual
# 3/A pé

[Teste: Checar se existe algum número < 1 ou > 3.
    Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno

:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - TIPO_VIAG")

# Substituindo valor 0 por None
df.loc[df['TIPO_VIAG']==0, 'TIPO_VIAG'] = None

# Verificando intervalo de valores - condições:
    # "MODO_PRIN >= 1" E "MODO_PRIN <= 3"
verifica_range(df, 'TIPO_VIAG', 1, 3)

log_output.info('\n\n=====\\n')
return df

def passo_tipo_est_auto(passo, df):
    """
Nada a fazer, essa informação não existe em 1997

```

```

# Valor/Descrição
# ----/----
# 0/Não Respondeu
# 1/Não Estacionou
# 2/Estacionamento Particular (Avulso / Mensal)
# 3/Estacionamento Próprio
# 4/Estacionamento Patrocinado
# 5/Rua (meio fio / zona azul / zona marrom / parquímetro)

[Teste: Checar se existe algum número < 0 ou > 5.
 Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: retorna dataframe modificado

"""

```

log_tela.info("### PASSO " + str(passo) + " - TIPO_EST_AUTO")

df['TIPO_EST_AUTO'] = None

log_output.info('\n\n=====\\n')

return df
```

```

def passo_valor_est_auto(passo, df, deflator):
"""
Nada há que se fazer em relação à coluna "VALOR_EST_AUTO".
Não há dados de 1997, coluna permanecerá vazia
:param passo: Número do passo atual para registro/log
:param df:
:return: sem retorno
"""
log_tela.info("### PASSO " + str(passo) + " - VALOR_EST_AUTO")

df['VALOR_EST_AUTO'] = None

log_output.info('\n\n=====\\n')

return df
```

```

def passo_tot_viang(passo, df):
"""
#####
#      ATENÇÃO      #
# ESTA FUNÇÃO SÓ DEVE SER #
# EXECUTADA APÓS A GERAÇÃO#
# DE TODOS OS ID'S, POIS #
# HÁ UMA DESCONFIANÇA   #
# QUANTO À QUALIDADE    #
#      DESTE DADO       #
#####

```


3.7 Funções que geram os “NO”s e os “ID”s

Função/Variável	Status
gera_nos_e_ids	Verificar NO VIAG

```
In [ ]: log_tela.info('Definindo funções que geram os "NO"s e os "ID"s')
log_output.info('\n\n=====\\n')
```

```
def gera_nos_e_ids(passo, df):
    """
    Esta função gera todos os IDs e todos os NOs das variáveis:
    DOM (domicílio), FAM (família), PESS (pessoa) e VIAG (viagem)
    A ordem de geração é:
    NO_DOM, ID_DOM, NO_FAM, ID_FAM, NO_PESS, ID_PESS, NO_VIAG, ID_VIAG
    Esta ordem é fixa pois cada uma dessas variáveis depende da geração
    da variável anterior.

    Os NOs são gerados como se fossem subíndices.
    No caso dos domicílios, eles são contabilizados dentro de cada zona.
    Assim, cada novo domicílio que aparece na listagem recebe um número
    que representa sua posição na sequência de domicílios dentro da zona
    na qual ele está contido.
    As famílias seguem o mesmo raciocínio, com relação ao domicílio, as
    pessoas com relação às famílias e as viagens com relação às pessoas.

    Deve-se apenas tomar cuidado com as viagens, pois existem pessoas que
    não realizaram viagens. Neste caso, estes registros devem contabilizar
    o NO_VIAG como zero, e não como 1. Para contemplar estes casos, vamos
    utilizar a variável F_VIAG, que representa quando há ou não viagem
    naquele registro (F_VIAG==1 tem viagem, F_VIAG==0 não tem viagem).
    Assim, o que precisamos fazer para calcular o NO_VIAG é agrupar os
    registros por pessoa (ID_PESS) e, dentro de cada agrupamento, somar
    o valor de F_VIAG registro a registro (linha por linha)
    cumulativamente. No final do processo, atribui-se zero a NO_VIAG
    quando F_VIAG for igual a zero.
    """

def gera_id_dom(row):
    """
    Gera o ID_DOM baseado no 'ANO', na 'ZONA_DOM' e no 'NO_DOM'
    O argumento passado é a "linha".
    Uso:
        df['ID_DOM'] = df.apply(gera_ID_DOM, axis=1)
    Retorna: ID_DOM da respectiva linha
    """
    # Formatando o ano para string
    ano = str(row['ANO'])

    # Formatando a zona para ficar como string com 3 caracteres
    zona = str('%03d' % row['ZONA_DOM'])
```

```

# Formatando no_dom para ficar como string com 4 caracteres
no_dom = str('%04d' % row['NO_DOM'])

# Retornando o número inteiro correspondente à string
# concatenada de ano + zona + no_dom
return ano + zona + no_dom


def gera_id_fam(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
    Uso:
        df['ID_FAM'] = df.apply(gera_ID_FAM, axis=1)
    Retorna: ID_FAM da respectiva linha
    """
    # Formatando id_dom como string
    id_dom = str(row['ID_DOM'])

    # Formatando no_fam como string para ficar com 2 caracteres
    no_fam = str('%02d' % row['NO_FAM'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_DOM com NO_FAM
    return id_dom + no_fam


def gera_id_pess(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
    Uso:
        df['ID_PESS'] = df.apply(gera_ID_PESS, axis=1)
    Retorna: ID_PESS da respectiva linha
    """
    # Formatando id_fam como string
    id_fam = str(row['ID_FAM'])

    # Formatando no_pess como string para ficar com 2 caracteres
    no_pess = str('%02d' % row['NO_PESS'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_FAM com NO_PESS
    return id_fam + no_pess


def gera_id_viag(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
    Uso:
        df['ID_VIAG'] = df.apply(gera_ID_VIAG, axis=1)
    Retorna ID_VIAG da respectiva linha
    """
    # Formatando id_pess como string
    id_pess = str(row['ID_PESS'])

```

```

# Formatando no_viag como string para ficar com 2 caracteres
no_viag = str('%02d' % row['NO_VIAG'])

# Retornando o número inteiro correspondente à string
# concatenada de ID_PESS com NO_VIAG
return id_pess + no_viag

#gera NO_DOM
log_tela.info("Gerando NO_DOM")
df['NO_DOM'] = df.groupby('ZONA_DOM', sort=False)['F_DOM'].cumsum()
log_tela.info("NO_DOM gerado")
log_tela.info("Gerando ID_DOM")
#gera ID_DOM
df['ID_DOM'] = df.apply(gera_id_dom, axis=1)
log_tela.info("ID_DOM gerado")

#gera NO_FAM
log_tela.info("Gerando NO_FAM")
df['NO_FAM'] = df.groupby('ID_DOM', sort=False)['F_FAM'].cumsum()
log_tela.info("NO_FAM gerado")
log_tela.info("Gerando ID_FAM")
#gera ID_FAM
df['ID_FAM'] = df.apply(gera_id_fam, axis=1)
log_tela.info("ID_FAM gerado")

#gera NO_PESS
log_tela.info("Gerando NO_PESS")
df['NO_PESS'] = df.groupby('ID_FAM', sort=False)['F_PESS'].cumsum()
log_tela.info("NO_PESS gerado")
log_tela.info("Gerando ID_PESS")
#gera ID_PESS
df['ID_PESS'] = df.apply(gera_id_pess, axis=1)
log_tela.info("ID_PESS gerado")

#gera NO_VIAG
log_tela.info("Gerando NO_VIAG")
#df['NO_VIAG'] = 1
df['NO_VIAG'] = df.groupby('ID_PESS', sort=False)['F_VIAG'].cumsum()
# Verificando se FE_VIAG == 0 e zerando NO_VIAG nesse caso
df.loc[df['FE_VIAG'] == 0, 'NO_VIAG'] = 0
log_tela.info("NO_VIAG gerado")
log_tela.info("Gerando ID_VIAG")
#gera ID_VIAG
df['ID_VIAG'] = df.apply(gera_id_viag, axis=1)
log_tela.info("ID_VIAG gerado")

return df

```

3.8 Função relativa à entrevista

Função/Variável	Status
passo_cd_entre	OK

Obs: o passo_cd_entre deve ser executado após o passo_tot_vitag.

```
In [ ]: log_tela.info('Definindo função relativa à entrevista')
log_output.info('\n\n=====\\n')

def passo_cd_entre(passo, df):
    """
    #####
    #      ATENÇÃO      #
    # ESTA FUNÇÃO SÓ DEVE SER #
    # EXECUTADA APÓS A GERAÇÃO#
    #      DO TOT_VIAG     #
    #####
    ----
    Substituir valores da coluna "CD_ENTRE"
    Todas viagens são consideradas "completas", segundo informações do Metrô

    * sem viagem: se TOT_VIAG == 0
    * com viagem: se TOT_VIAG != 0

    # #####Categorias novas
    # | Valor | Descrição |
    # | ----- | ----- |
    # | 0 | Completa sem viagem |
    # | 1 | Completa com viagem |

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""

log_tela.info("### PASSO " + str(passo) + " - CD_ENTRE")
log_tela.warning('\n\n#####\n' + 
                 'Este passo DEVE ser executado apenas após a \\n' +
                 'execução do passo que geram o TOT_VIAG.\\n' +
                 '#####\n')

# Definindo 'CD_ENTRE' baseado no valor de 'TOT_VIAG'
df.loc[df['TOT_VIAG'] == 0, 'CD_ENTRE'] = 0
df.loc[df['TOT_VIAG'] != 0, 'CD_ENTRE'] = 1

verifica_dummy(df, 'CD_ENTRE')
log_output.info('\n\n=====\\n')

return df
```

```
In [ ]: def passo_correcoes(passo, df):
    log_tela.info("### PASSO " + str(passo) + " - Correções")
    log_tela.info("Corrigindo 'SERV_PAS' caso TOT_VIAG seja nulo.")
    df.loc[df['TOT_VIAG'] == 0, 'SERV_PAS_ORIG'] = 0
    df.loc[df['TOT_VIAG'] == 0, 'SERV_PAS_DEST'] = 0

    return df
```

4 Definindo a função principal (main)

Esta função vai ler os arquivos (csv) necessários e, então, irá efetuar cada passo, chamando as respectivas funções que foram definidas acima. Ao final, esta função irá salvar o resultado das transformações realizadas num arquivo csv.

```
In [ ]: def main():
    start_time = time.time()

    log_tela.info("Horário de início da execução: %s" %
                  time.strftime("%H:%M", time.localtime(start_time)))

    # ----
    log_tela.info('Lendo arquivos CSV')

    log_output.info('Lendo CSV da base original da OD.')
    od = pd.read_csv('bases/OD_1997.csv', sep=';', decimal=',')

    deflator = 1.94136464
    log_output.info('Deflator a ser utilizado para correção monetária: ' +
                    str(deflator))

    # Filtrando dataframe para pegar apenas uma amostra
    #od = od[:1000].copy()

    log_output.info('\n\n=====\\n')
    log_tela.info('Pré-Processamento.')
    od = pre_processamento(od)

    log_tela.info('Imprimindo a lista de variáveis do dataframe da OD.')
    log_tela.debug(od.columns.tolist())

    #Contador de 'PASSO'
    passo = 1

    # ----
    # ##Passo: "ANO"
    od = passo_ano(passo, od)
    passo += 1

    # ----
    # ##Passo: "DIA_SEM"
    od = passo_dia_sem(passo, od)
    passo += 1

    # ----
    # ##Passo: "UCODs"
    od = passo_ucods(passo, od)
    passo += 1

    # ----
    # ##Passo: "ZONA_DOM"
    od = passo_zona_dom(passo, od)
    passo += 1
```

```

# -----
# ##Passo: "SUBZONA_DOM"
od = passo_subzona_dom(passo, od)
passo += 1

# -----
# ##Passo: "MUN_DOM"
od = passo_mun_dom(passo, od)
passo += 1

# -----
# ##Passo: "F_DOM"
od = passo_f_dom(passo, od)
passo += 1

# -----
# ##Passo: "FE_DOM"
od = passo_fe_dom(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_DOM"
od = passo_tipo_dom(passo, od)
passo += 1

# -----
# ##Passo: "TOT_FAM"
od = passo_tot_fam(passo, od)
passo += 1

# -----
# ##Passo: "F_FAM"
od = passo_f_fam(passo, od)
passo += 1

# -----
# ##Passo: "FE_FAM"
od = passo_fe_fam(passo, od)
passo += 1

# -----
# ##Passo: "COND_MORA"
od = passo_cond_mora(passo, od)
passo += 1

# -----
# ##Passo: "QT_AUTO"
od = passo_qt_auto(passo, od)
passo += 1

# -----
# ##Passo: "QT_BICI"
od = passo_qt_bici(passo, od)
passo += 1

```

```

# -----
# ##Passo: "QT_MOTO"
od = passo_qt_moto(passo, od)
passo += 1

# -----
# ##PASSO: "REN_FAM"
od = passo_ren_fam(passo, od, deflator)
passo += 1

# -----
# ##Passo: "CD_RENFAM"
od = passo_cd_renfam(passo, od)
passo += 1

# -----
# ##Passo: "F_PESS"
od = passo_f_pess(passo, od)
passo += 1

# -----
# ##Passo: "FE_PESS"
od = passo_fe_pess(passo, od)
passo += 1

# -----
# ##Passo: "SIT_FAM"
od = passo_sit_fam(passo, od)
passo += 1

# -----
# ##Passo: "IDADE"
od = passo_idade(passo, od)
passo += 1

# -----
# ##Passo: "SEXO"
od = passo_sexo(passo, od)
passo += 1

# -----
# ##Passo: "GRAU_INSTR"
od = passo_grau_instr(passo, od)
passo += 1

# -----
# ##Passo: "OCUP"
od = passo_ocup(passo, od)
passo += 1

# -----
# ##Passo: "SETOR_ATIV"
od = passo_setor_ativ(passo, od)

```

```

    passo += 1

    # ##Passo: "REN_IND"
    od = passo_ren_ind(passo, od, deflator)
    passo += 1

    # ##Passo: "CD_RENIND"
    od = passo_cd_renind(passo, od)
    passo += 1

    # -----
    # ##Passo: "F_VIAG"
    od = passo_f_viag(passo, od)
    passo += 1

    # -----
    # ##"FE_VIAG"
    od = passo_fe_viag(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_ESC"
    od = passo_zona_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_ESC"
    od = passo_subzona_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_ESC"
    od = passo_mun_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "ESTUDA"
    # Este passo deve vir após os passos de localização da escola
    od = passo_estuda(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_TRAB1"
    od = passo_zona_trab1(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_TRAB1"
    od = passo_subzona_trab1(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_TRAB1"
    od = passo_mun_trab1(passo, od)

```

```

    passo += 1

    # -----
    # ##Passo: "ZONA_TRAB2"
    od = passo_zona_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_TRAB2"
    od = passo_subzona_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_TRAB2"
    od = passo_mun_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_ORIG"
    od = passo_zona_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_ORIG"
    od = passo_subzona_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_ORIG"
    od = passo_mun_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_DEST"
    od = passo_zona_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_DEST"
    od = passo_subzona_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_DEST"
    od = passo_mun_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "SERV_PAS_ORIG"
    od = passo_serv_pas_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "SERV_PAS_DEST"

```

```

od = passo_serv_pas_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_ORIG"
od = passo_motivo_orig(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_DEST"
od = passo_motivo_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOD01"
od = passo_modo1(passo, od)
passo += 1

# -----
# ##Passo: "MOD02"
od = passo_modo2(passo, od)
passo += 1

# -----
# ##Passo: "MOD03"
od = passo_modo3(passo, od)
passo += 1

# -----
# ##Passo: "MOD04"
od = passo_modo4(passo, od)
passo += 1

# -----
# ##Passo: "MODO_PRIN"
od = passo_modo_prin(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_VIAG"
od = passo_tipo_vitag(passo, od)
passo += 1

# -----
# ##"H_SAIDA"; "MIN_SAIDA"; "ANDA_ORIG"; "H_CHEG"; "MIN_CHEG";
# "ANDA_DEST"; "DIST_VIAG" e "DURACAO"
# Nada há que se fazer em relação aos dados das colunas acima mencionadas

# -----
# ##Passo: "TIPO_EST_AUTO"
od = passo_tipo_est_auto(passo, od)
passo += 1

# -----

```

```

# ##Passo: "VALOR_EST_AUTO"
od = passo_valor_est_auto(passo, od, deflator)
passo += 1

# -----
# ##Passo: Coordenadas
od = coordenadas(passo, od)
passo += 1

## O passo TOT_VIAG apenas é chamado após a geração dos IDs.

# -----
# ##Passo: Gerando NO's e ID's
od = gera_nos_e_ids(passo, od)
passo += 1

# -----
# ##Passo: "TOT_VIAG"
od = passo_tot_viag(passo, od)
passo += 1

# -----
# ##Passo: "CD_ENTRE"
od = passo_cd_entre(passo, od)
passo += 1

# -----
# ##Passo: Correções
od = passo_correcoes(passo, od)
passo += 1

log_tela.info('Salvando dataframe como arquivo CSV')
# -----
# ## Salvando o dataframe num arquivo local
od.to_csv('outputs/1997_od.csv', sep=';', decimal=',', index=False)

log_tela.info("Base gerada. Arquivo: outputs/1997_od.csv")

log_tela.info("Tempo total de execução: %s segundos" %
             (time.time() - start_time))

log_tela.info("Horário de finalização: %s" %
             (time.strftime("%H:%M", time.localtime(time.time()))))

log_tela.info("Terminou o main")

```

5 RUN the main() function....

```
In [ ]: if __name__ == "__main__":
    main()
```

rotina_final_2007

January 4, 2016

```
In [ ]: # Setup Inicial  
In [ ]: # coding: utf-8  
        import math  
        import logging  
        import time  
        import pandas as pd  
        import csv  
  
        # Faz os gráficos um pouco mais bonitos  
        pd.set_option('display.mpl_style', 'default')
```

1 Definindo Loggers

Define os loggers

Estes 'loggers' serão utilizados para salvar as saídas (outputs) em um arquivo de texto no diretório 'outputs'.

ref: <http://stackoverflow.com/questions/17035077/python-logging-to-multiple-log-files-from-different-classes>

ATENÇÃO: RODAR O BLOCO ABAIXO APENAS UMA VEZ, SE ESTE BLOCO FOR EXECUTADO MAIS DE UMA VEZ OS LOGS SERÃO DUPLICADOS.

```
In [ ]: log_formatter = logging.Formatter('%(asctime)s | %(levelname)s: %(message)s')  
  
log_output = logging.getLogger('log_output')  
FH_output = logging.FileHandler(  
    'outputs/2007_output.log', mode='w')  
FH_output.setFormatter(log_formatter)  
log_output.setLevel(logging.INFO)  
log_output.addHandler(FH_output)  
log_output.propagate = False  
  
# Este logger (log_tela) loga na tela e também  
# no arquivo de output, junto com o conteúdo do  
# logger log_output  
log_tela = logging.getLogger('log_tela')  
SH_tela = logging.StreamHandler()  
SH_tela.setFormatter(log_formatter)  
log_tela.addHandler(SH_tela)  
log_tela.addHandler(FH_output)  
log_tela.setLevel(logging.INFO)  
log_tela.propagate = False
```

2 Funções de 2007

2.1 Funções gerais assessórias

Função	Status
consulta_refext	OK
verifica_dummy	OK
verifica_range	OK
pre_processamento	OK
coordenadas	OK

```
In [ ]: log_tela.info('Definindo as funções gerais assessórias')
log_output.info('\n\n=====\\n')

def verifica_dummy(df, variavel):
    """
    Verifica se uma variável, dummy, contém algum valor diferente de 0 ou de 1.
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados.
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_dummy(dataframe, 'coluna a ser verificada')
    """
    contador_de_erros = 0
    log_tela.info('Verificando a variável Dummy: ' + variavel)

    df_erros = df[(df[variavel] != 0) & (df[variavel] != 1)]
    if len(df_erros[variavel].value_counts()) > 0:
        log_tela.warning(variavel + ": " +
                          str(len(df_erros[variavel].value_counts())) +
                          " erros encontrados:\\n" +
                          str(df_erros[variavel].value_counts()))
    else:
        log_tela.info(variavel + ": Nenhum erro encontrado.")

def verifica_range(df, variavel, valor_menor, valor_maior):
    """
    Verifica se uma variável, do tipo número inteiro, contém algum valor menor
    que "valor_menor" ou maior que "valor_maior".
    :param df: dataframe com os dados a serem verificados
    :param variavel: string com o nome da variável (coluna) que tem os dados
        que devem ser verificados
    :param valor_menor: Valor mínimo esperado na variável (int ou float)
    :param valor_maior: Valor máximo esperado na variável (int ou float)
    :return: Sem retorno, apenas salva as avaliações nos logs.
    Uso:
        verifica_range(dataframe, 'nome_variavel', valor_menor, valor_maior)
```

```

"""
log_tela.info('Verificando Range da variável: ' + variavel)
# Obs: Registros inválidos: None (equivalente a NA)
nulos = df[variavel].isnull().sum()
nulos = nulos if nulos else 0
log_output.info('\n\n' +
    ' - ' + 'Mínimo esperado: ' + str(valor_menor) + '\n' +
    ' - ' + 'Máximo esperado: ' + str(valor_maior) + '\n' +
    ' - ' + 'Total de registros: ' + str(len(df[variavel])) +
    '\n' +
    ' - ' + 'Registros nulos (NA): ' +
    str(df[variavel].isnull().sum()) + '\n'
)
df_erro = df[(df[variavel] < valor_menor) | (df[variavel] > valor_maior)]

if len(df_erro[variavel].value_counts()) > 0:

    result = df_erro[variavel].value_counts().sort_index()
    # Verificando limite inferior
    if result.first_valid_index() < valor_menor:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro mínimo encontrado: ' +
            str(result.first_valid_index()) + " - abaixo do esperado!")
    # Verificando limite superior
    if result.last_valid_index() > valor_maior:
        log_tela.warning(
            variavel + ': ' + 'Valor inteiro máximo encontrado: ' +
            str(result.last_valid_index()) + " - acima do esperado!")

    log_tela.warning(variavel + ': ' +
        str(len(df_erro[variavel].value_counts())) +
        ' valor(es) incorreto(s)' +
        'encontrado(s) nesta variável:\n' +
        str(df_erro[variavel].value_counts()))
else:
    log_tela.info(variavel + ": Nenhum erro encontrado.")

def pre_processamento(df):
"""
Realiza algumas ações prévias ao processamento dos dados,
removendo alguns registros e ajustando o dataframe.
"""

log_tela.info('Criando/Renomeando colunas no dataframe principal')

log_output.info('Renomeando coluna UCOD para UCOD_DOM')
df.rename(columns={'UCOD': 'UCOD_DOM'}, inplace=True)

log_output.info('Renomeando coluna ANDA_CHEG para ANDA_DEST')
df.rename(columns={'ANDA_CHEG': 'ANDA_DEST'}, inplace=True)

log_tela.info('Verificando se todas as variáveis esperadas' +

```

```

        'existem no dataframe.\n' +
        'Caso não exista alguma, ela é criada.')
variaveis = ['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
             'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
             'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
             'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
             'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
             'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
             'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
             'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
             'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
             'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
             'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
             'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
             'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
             'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
             'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
             'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
             'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
             'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
             'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
             'VALOR_EST_AUTO']

for variavel in variaveis:
    if variavel not in df.columns:
        # Se a variável não existe no dataframe então ela é criada
        # com valor padrão de NONE (NA)
        df[variavel] = None
    log_tela.info('Criando a variavel ' + variavel + ' no dataframe')

log_output.info('Reordenando as variáveis')
df = df[['ANO', 'CD_ENTRE', 'DIA_SEM', 'UCOD_DOM', 'ZONA_DOM',
          'SUBZONA_DOM', 'MUN_DOM', 'CO_DOM_X', 'CO_DOM_Y', 'ID_DOM',
          'F_DOM', 'FE_DOM', 'NO_DOM', 'TIPO_DOM', 'TOT_FAM', 'ID_FAM',
          'F_FAM', 'FE_FAM', 'NO_FAM', 'COND_MORA', 'QT_AUTO', 'QT_BICI',
          'QT_MOTO', 'CD_RENFAM', 'REN_FAM', 'ID_PESS', 'F_PESS',
          'FE_PESS', 'NO_PESS', 'SIT_FAM', 'IDADE', 'SEXO', 'ESTUDA',
          'GRAU_INSTR', 'OCUP', 'SETOR_ATIV', 'CD_RENIND', 'REN_IND',
          'UCOD_ESC', 'ZONA_ESC', 'SUBZONA_ESC', 'MUN_ESC', 'CO_ESC_X',
          'CO_ESC_Y', 'UCOD_TRAB1', 'ZONA_TRAB1', 'SUBZONA_TRAB1',
          'MUN_TRAB1', 'CO_TRAB1_X', 'CO_TRAB1_Y', 'UCOD_TRAB2',
          'ZONA_TRAB2', 'SUBZONA_TRAB2', 'MUN_TRAB2', 'CO_TRAB2_X',
          'CO_TRAB2_Y', 'ID_VIAG', 'F_VIAG', 'FE_VIAG', 'NO_VIAG',
          'TOT_VIAG', 'UCOD_ORIG', 'ZONA_ORIG', 'SUBZONA_ORIG',
          'MUN_ORIG', 'CO_ORIG_X', 'CO_ORIG_Y', 'UCOD_DEST', 'ZONA_DEST',
          'SUBZONA_DEST', 'MUN_DEST', 'CO_DEST_X', 'CO_DEST_Y',
          'DIST_VIAG', 'SERV_PAS_ORIG', 'SERV_PAS_DEST', 'MOTIVO_ORIG',
          'MOTIVO_DEST', 'MODO1', 'MODO2', 'MODO3', 'MODO4', 'MODO_PRIN',
          'TIPO_VIAG', 'H_SAIDA', 'MIN_SAIDA', 'ANDA_ORIG', 'H_CHEG',
          'MIN_CHEG', 'ANDA_DEST', 'DURACAO', 'TIPO_EST_AUTO',
          'VALOR_EST_AUTO']]

log_output.info('\n\n=====\\n')
log_output.info('\\n\\n=====\\n')

```

```
    return df

def coordenadas(passo, df):
    """
    Nada a fazer, as coordenadas já constam na base consolidada.

    :param passo: número do passo para o log
    :param df: Dafaframe a ser modificado (ex.: od2007)
    :return: retorna o dataframe
    """
    log_tela.info("### PASSO " + str(passo) + " - COORDENADAS")
    log_output.info('\n\n=====\\n')

    return df
```

2.2 Funções Gerais

Função/Variável	Status
passo_ano	OK
passo_dia_sem	OK
passo_ucods	OK

```
In [ ]: log_tela.info('Definindo as funções gerais')
log_output.info('\n'=====\\n')

def passo_ano(passo, df):
    """
    Preenche a coluna "ANO" com valor 4 em todas células
    Categorias:
    /valor/ano_correspondente/
    /-----/-----
    /1/1977/
    /2/1987/
    /3/1997/
    /4/2007/

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - ANO")

    # Definindo valor '4' para todas as células da coluna ANO
    df["ANO"] = 4

    return df

def passo_dia_sem(passo, df):
    """
    Apenas verificar os valores existentes
    # #####Categorias:
    # Valor/Descrição
    # -----/-----
    # 2/Segunda-Feira
    # 3/Terça-Feira
    # 4/Quarta-Feira
    # 5/Quinta-Feira
    # 6/Sexta-Feira

    :param passo: Número do passo atual para registro/log
    :param df: dataframe a ser modificado
    :return: retorna o dataframe modificado
    """

```

```

log_tela.info("### PASSO " + str(passo) + " - DIA_SEM")

# Substituindo **0** por **None**
df.loc[df['DIA_SEM']==0,'DIA_SEM'] = None

# Verificando intervalo de valores - condições:
# "DIA_SEM >= 2" E "DIA_SEM <= 6"
verifica_range(df, 'DIA_SEM', 2, 6)
log_output.info('\n\n=====\\n')

return df

def passo_ucods(passo, df):
    """
    Itera sobre os registros do dataframe ucods,
    a cada iteração, utiliza o valor da coluna ZONA_REF
    para filtrar o dataframe df em cada um dos tipos de ZONA
    (DOM, ESC, TRAB1, TRAB2, ORIG, DEST) e substitui o valor
    da respectiva UCOD.

    :param passo: número do passo para o log
    :param df: Dataframe a ser modificado (ex.: od2007)
    :return: retorna o dataframe modificado com todas as UCODS aplicadas
    """
    log_tela.info("### PASSO " + str(passo) + " - UCODS")

    def ucod_aux(row):
        df.loc[df['ZONA_DOM']==row['ZONA_REF'], 'UCOD_DOM'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_ESC']==row['ZONA_REF'], 'UCOD_ESC'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_TRAB1']==row['ZONA_REF'], 'UCOD_TRAB1'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_TRAB2']==row['ZONA_REF'], 'UCOD_TRAB2'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_ORIG']==row['ZONA_REF'], 'UCOD_ORIG'] = row['UCOD_BUSCADA']
        df.loc[df['ZONA_DEST']==row['ZONA_REF'], 'UCOD_DEST'] = row['UCOD_BUSCADA']

    log_output.info('Lendo arquivo auxiliar UCOD')
    ucods = pd.read_csv('auxiliares/UCOD-2007.csv', sep=';', decimal=',')

    # Esta variável out não é utilizada para nada além de evitar um
    # monte de output que não será utilizado e que é gerado pelo método apply.
    out = ucods.apply(ucod_aux, axis=1)

    # Verificando intervalo de valores - condições:
    # "UCOD_XXX >= 1" E "UCOD_XXX <= 67"
    for tipo in ['DOM', 'ESC', 'TRAB1', 'TRAB2', 'ORIG', 'DEST']:
        verifica_range(df, 'UCOD_' + tipo, 1, 67)
    log_output.info('\n\n=====\\n')

    return df

```

2.3 Funções do Domicílio

Função/Variável	Status
passo_zona_dom	OK
passo_subzona_dom	OK
passo_mun_dom	OK
passo_f_dom	OK
passo_fe_dom	OK
passo_tipo_dom	OK

```
In [ ]: log_tela.info('Definindo as funções do domicílio')
log_output.info('\n\n=====\\n')

def passo_zona_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 460

    [Teste: Checar se existe algum número < 1 ou > 460.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_DOM")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_DOM']==0,'ZONA_DOM'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_DOM >= 1" E "ZONA_DOM <= 460"
    verifica_range(df, 'ZONA_DOM', 1, 460)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_dom(passo, df):
    """
    A variável SUBZONA_DOM ficará vazia (NA) em 2007 pois este ano já contém
    os dados de coordenadas.

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DOM")
```

```

df['SUBZONA_DOM'] = None

log_output.info('\n\n=====\\n')

return df

def passo_mun_dom(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_DOM")

    # Substituindo valor 0 por None
    df.loc[df['MUN_DOM']==0,'MUN_DOM'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_DOM >= 1" E "MUN_DOM <= 39"
    verifica_range(df, 'MUN_DOM', 1, 39)
    log_output.info('\n\n=====\\n')

    return df

def passo_f_dom(passo, df):
    """
    Checar se existe algum erro na coluna "F_DOM"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro do Domicílio

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - F_DOM")

    verifica_dummy(df, 'F_DOM')
    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_fe_dom(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_DOM"

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_DOM")
    log_output.info('\n\n=====\\n')

    return df

def passo_tipo_dom(passo, df):
    """
    * Substituir **0** por **None (NA)**
    * Substituir **3** por **2**
    * Substituir **2** por **0**

    # #####Categorias anteriores / novas
    # Valor / Descrição
    # ----/----
    # 1/Particular
    # 2/Coletivo
    # 3/Favela

    # #####Categorias novas
    # Valor / Descrição
    # ----/----
    # 0/Coletivo
    # 1/Particular

    [Teste: Checar se existe algum número < 0 ou > 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - TIPO_DOM")

    df.loc[df['TIPO_DOM']==0,'TIPO_DOM'] = None
    df.loc[df['TIPO_DOM']==3,'TIPO_DOM'] = 2
    df.loc[df['TIPO_DOM']==2,'TIPO_DOM'] = 0

    # Verificando intervalo de valores - condições:
    # "TIPO_DOM >= 0" E "TIPO_DOM <= 1"
    verifica_dummy(df, 'TIPO_DOM')
    log_output.info('\n\n=====\\n')

```

```
return df
```

2.4 Funções da Família

Função/Variável	Status
passo_tot_fam	OK
passo_f_fam	OK
passo_fe_fam	OK
passo_cond_mora	OK
passo_qt_auto	OK
passo_qt_bici	OK
passo_qt_moto	OK
passo_ren_fam	OK
passo_cd_renfam	OK

```
In [ ]: log_tela.info('Definindo as funções da família')
log_output.info('\n\n=====\\n')

def passo_tot_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "TOT_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - TOT_FAM")
    log_output.info('\\n\\n=====\\n')

    return df

def passo_f_fam(passo, df):
    """
    Checar se existe algum erro na coluna "F_FAM"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Família

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - F_FAM")

    verifica_dummy(df, 'F_FAM')
```

```

log_output.info('\n\n=====\\n')
return df

def passo_fe_fam(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "FE_FAM"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - FE_FAM")
    log_output.info('\n\n=====\\n')

    return df

def passo_cond_mora(passo, df):
    """
    Substituir valores da coluna "COND_MORA"

    * Substituir todos valores **5** por **0**
    * Substituir todos valores **0** por **None**
    * Substituir todos valores **4** por **3**

    ##### Categorias anteriores

    Valor/Descrição
    ----/----
    1/Alugada
    2/Própria
    3/Cedida
    4/Outros
    5/Não respondeu

    ##### Categorias novas

    Valor/Descrição
    ----/----
    1/Alugada
    2/Própria
    3/Outros

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: passo
    :param df: dataframe
    :return: dataframe modificado
    """

    log_tela.info("### PASSO " + str(passo) + " - COND_MORA")

    # Substituindo valor 5 por 0
    df.loc[df['COND_MORA']==5,'COND_MORA'] = 0

```

```

# Substituindo valor 0 por None
df.loc[df['COND_MORA']==0,'COND_MORA'] = None
# Substituindo valor 4 por 3
df.loc[df['COND_MORA']==4,'COND_MORA'] = 3

# Verificando intervalo de valores - condições:
# "COND_MORA >= 1" E "COND_MORA <= 3"
verifica_range(df, 'COND_MORA', 1, 3)
log_output.info(' \n\n===== \n')

return df


def passo_qt_auto(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "QT_AUTO"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - QT_AUTO")
    log_output.info(' \n\n===== \n')

    return df


def passo_qt_bici(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "QT_AUTO"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    log_tela.info('### PASSO ' + str(passo) + ' - QT_BICI')
    log_output.info(' \n\n===== \n')

    return df


def passo_qt_moto(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "QT_AUTO"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: dataframe modificado
    """
    log_tela.info('### PASSO ' + str(passo) + ' - QT_MOTO')
    log_output.info(' \n\n===== \n')

    return df


def passo_ren_fam(passo, df, deflator):
    """

```

Corrigir a renda familiar de acordo com o deflator passado na função.

:param passo: Número do passo atual para registro/log

:param df: dataframe

:param deflator: Deflator utilizado para correção da renda.

:return: retorna o dataframe com as devidas modificações

"""

```
log_tela.info("### PASSO " + str(passo) + " - REN_FAM")

df['REN_FAM'] = df['REN_FAM'] * deflator

return df
```

def passo_cd_renfam(passo, df):

"""

Substituir valores da coluna "CD_RENFAM"

** Substituir todos valores **2** por **0***

** Substituir todos valores **3** por **2***

** Substituir todos valores **4** por **2***

Categorias anteriores

Valor/Descrição

----/----

1/Renda familiar declarada e maior que zero

2/Renda familiar declarada como zero

3/Renda atribuída pelo Critério Brasil

4/Renda atribuída pela Média da Zona

Categorias novas

Valor/Descrição

----/----

0/Renda Familiar Declarada como Zero

1/Renda Familiar Declarada e Maior que Zero

2/Renda Atribuída

[Teste: Checar se existe algum número < 0 ou > 2.

Se encontrar, retornar erro indicando em qual linha.]

:param passo: Número do passo atual para registro/log

:param df:

:return: retorna o dataframe corrigido

"""

```
log_tela.info("### PASSO " + str(passo) + " - CD_RENFAM")

# Substituindo valor 2 por 0
df.loc[df['CD_RENFAM']==2, 'CD_RENFAM'] = 0
# Substituindo valor 3 por 2
df.loc[df['CD_RENFAM']==3, 'CD_RENFAM'] = 2
# Substituindo valor 4 por 2
df.loc[df['CD_RENFAM']==4, 'CD_RENFAM'] = 2

# Dividindo a categoria '0', "Respondeu", em:
#   - 0 "Renda Familiar Declarada como Zero" e
#   - 1 "Renda Familiar Declarada e Maior que Zero"
```

```
df.loc[(df['CD_RENFAM'] == 0) &
       (df['REN_FAM'] != 0) &
       (df['REN_FAM'].notnull()), 'CD_RENFAM'] = 1

# Verificando intervalo de valores - condições:
    # "CD_RENFAM >= 0" E "CD_RENFAM <= 2"
verifica_range(df, 'CD_RENFAM', 0, 2)
log_output.info('\\n\\n=====\\n')

return df
```

2.5 Funções da pessoa

Função/Variável	Status
passo_f_pess	OK
passo_fe_pess	OK
passo_sit_fam	OK
passo_idade	OK
passo_sexo	OK
passo_grau_instr	OK
passo_ocup	OK
passo_setor_ativ	OK
passo_ren_ind	OK
passo_cd_renind	OK
passo_estuda	Verificar

Obs.: Se ESTUDA for ser alterado pela verificação do ZONA_ESC, no main() ele deverá ser chamado após ZONA_ESC. Atualmente ele já está localizado após o ZONA_ESC na chamada do main(). Após confirmar o que será feito com ele, se for necessário alterar sua localização.

```
In [ ]: log_tela.info('Definindo as funções da pessoa')
log_output.info('\n\n=====\\n')

def passo_f_pess(passo, df):
    """
    Checar se existe algum erro na coluna "F_PESS"

    # #####Categorias
    # Valor/Descrição
    # ----/----
    # 0/Demais registros
    # 1/Primeiro Registro da Pessoa

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - F_PESS")

    verifica_dummy(df, 'F_PESS')
    log_output.info('\n\n=====\\n')

    return df

def passo_fe_pess(passo, df):
```

```

"""
Nada há que se fazer em relação aos dados das colunas "FE_PESS"
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
log_tela.info("### PASSO " + str(passo) + " - FE_PESS")
log_output.info('\n\n=====\\n')

return df

def passo_sit_fam(passo, df):
    """
    * Substituir todos valores **5** por **4**
    * Substituir todos valores **6** por **5**
    * Substituir todos valores **7** por **6**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Pessoa Responsável
    2/Cônjuge/Companheiro(a)
    3/Filho(a)/Enteado(a)
    4/Outro Parente
    5/Agregado
    6/Empregado Residente
    7/Parente do empregado

    ##### Categorias novas:
    Valor/Descrição
    ----/----
    1/ Pessoa Responsável
    2/ Cônjuge/Companheiro(a)
    3/ Filho(a)/Enteado(a)
    4/ Outro Parente / Agregado
    5/ Empregado Residente
    6/ Outros (visitante não residente / parente do empregado)

    [Teste: Checar se existe algum número < 1 ou > 6.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SIT_FAM")

    # Substituindo valor 5 por 4
    df.loc[df['SIT_FAM']==5,'SIT_FAM'] = 4
    # Substituindo valor 6 por 5
    df.loc[df['SIT_FAM']==6,'SIT_FAM'] = 5
    # Substituindo valor 7 por 6
    df.loc[df['SIT_FAM']==7,'SIT_FAM'] = 6

```

```

# Verificando intervalo de valores - condições:
# "SIT_FAM >= 1" E "SIT_FAM <= 6"
verifica_range(df, 'SIT_FAM', 1, 6)
log_output.info('\n\n=====\\n')

return df

def passo_idade(passo, df):
    """
    Nada há que se fazer em relação aos dados da coluna "IDADE"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return:
    """
    log_tela.info("### PASSO " + str(passo) + " - IDADE")
    log_output.info('\n\n=====\\n')

    return df

def passo_sexo(passo, df):
    """
    #####Categorias anteriores
    # Valor/Descrição
    # ----/----
    # 0/Não Respondeu (-> None)
    # 1/Masculino
    # 2/Feminino

    #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 0/Masculino
    # 1/Feminino

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SEXO")

    # Substituindo valor 0 por None
    df.loc[df['SEXO']==0,'SEXO'] = None
    # Substituindo valor 1 por 0
    df.loc[df['SEXO']==1,'SEXO'] = 0
    # Substituindo valor 2 por 1
    df.loc[df['SEXO']==2,'SEXO'] = 1

    # Verificando intervalo de valores - condições:
    # "SEXO >= 0" E "SEXO <= 1"
    verifica_dummy(df, 'SEXO')

```

```

log_output.info('\n\n=====\\n')

return df

def passo_grau_instr(passo, df):
    """
    Substituir valores da coluna "GRAU_INSTR"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **2**
    * Substituir todos valores **4** por **3**
    * Substituir todos valores **5** por **4**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores:
    Valor/Descrição
    ----/----
    1/Não-alfabetizado/Primario Incompleto
    2/Primario Completo/Ginasio Incompleto
    3/Ginásio Completo/Colegial Incompleto
    4/Colegial Completo/Superior Incompleto
    5/Superior Completo

    ##### Categorias novas
    Valor/Descrição
    ----/----
    1/Não-Alfabetizado/Fundamental Incompleto
    2/Fundamental Completo/Médio Incompleto
    3/Médio Completo/Superior Incompleto
    4/Superior completo

    [Teste: Checar se existe algum número < 1 ou > 4.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - GRAU_INSTR")

    # Substituindo valor 2 por 1
    df.loc[df['GRAU_INSTR']==2, 'GRAU_INSTR'] = 1
    # Substituindo valor 3 por 2
    df.loc[df['GRAU_INSTR']==3, 'GRAU_INSTR'] = 2
    # Substituindo valor 4 por 3
    df.loc[df['GRAU_INSTR']==4, 'GRAU_INSTR'] = 3
    # Substituindo valor 5 por 4
    df.loc[df['GRAU_INSTR']==5, 'GRAU_INSTR'] = 4
    # Substituindo valor 0 por None
    df.loc[df['GRAU_INSTR']==0, 'GRAU_INSTR'] = None

    # Verificando intervalo de valores - condições:
    # "GRAU_INSTR >= 1" E "GRAU_INSTR <= 4"
    verifica_range(df, 'GRAU_INSTR', 1, 4)

```

```

log_output.info('\'\n\n=====\\n')

return df

def passo_ocup(passo, df):
    """
    Substituir valores da coluna "OCUP"

    Substituir todos valores **2** por **1**
    Substituir todos valores **3** por **2**
    Substituir todos valores **4** por **3**
    Substituir todos valores **5** por **4**
    Substituir todos valores **6** por **5**
    Substituir todos valores **7** por **6**
    Substituir todos valores **8** por **7**
    Substituir todos valores **0** por **None**

    #####Categorias anteriores:
    Valor/Descrição
    ----/----
    1/Tem trabalho
    2/Faz bico
    3/Em licença médica
    4/Aposentado / pensionista
    5/Sem trabalho
    6/Nunca trabalhou
    7/Dona de casa
    8/Estudante

    #####Categorias novas
    Valor/Descrição
    ----/----
    1/Tem trabalho
    2/Em licença médica
    3/Aposentado / pensionista
    4/Desempregado
    5/Sem ocupação
    6/Dona de casa
    7/Estudante

[Teste: Checar se existe algum número < 1 ou > 7.
Se encontrar, retornar erro indicando em qual linha.]]

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - OCUP")

# Substituindo valor 2 por 1
df.loc[df['OCUP']==2,'OCUP'] = 1
# Substituindo valor 3 por 2
df.loc[df['OCUP']==3,'OCUP'] = 2
# Substituindo valor 4 por 3
df.loc[df['OCUP']==4,'OCUP'] = 3

```

```

# Substituindo valor 5 por 4
df.loc[df['OCUP']==5,'OCUP'] = 4
# Substituindo valor 6 por 5
df.loc[df['OCUP']==6,'OCUP'] = 5
# Substituindo valor 7 por 6
df.loc[df['OCUP']==7,'OCUP'] = 6
# Substituindo valor 8 por 7
df.loc[df['OCUP']==8,'OCUP'] = 7
# Substituindo valor 0 por None
df.loc[df['OCUP']==0,'OCUP'] = None

# Verificando intervalo de valores - condições:
# "OCUP >= 1" E "OCUP <= 7"
verifica_range(df, 'OCUP', 1, 7)
log_output.info('\'\n\'=====\\n')

return df

def passo_setor_ativ(passo, df):
    """
    Substituir valores da coluna "SETOR_ATIV"

    Na coluna "SETOR_ATIV", linha i,
        ler o valor da linha i da coluna "SETOR_ATIV", daí,
        buscar o mesmo valor na coluna "COD" do arquivo setor_ativ-2007.csv.
    Ao achar, retornar o valor da mesma linha, só que da coluna "COD_UNIF"

    #####Categorias anteriores
    > ver arquivo .csv

    #####Categorias novas
    Valor/Descrição
    ----/----
    1/Agrícola
    2/Construção Civil
    3/Indústria
    4/Comércio
    5/Administração Pública
    6/Serviços de Transporte
    7/Doutros serviços
    8/Doutros
    9/Não se aplica

    [Teste: Checar se existe algum número < 1 ou > 9.
     Se encontrar, retornar erro indicando em qual linha.]"
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna dataframe modificado
    """
    log_tela.info("### PASSO " + str(passo) + " - SETOR_ATIV")

    log_output.info('Lendo arquivo de referência externa setor_ativ-2007.csv')
    df_setor = pd.read_csv('auxiliares/setor_ativ-2007.csv', sep=';', decimal=',')

```

```

def setor_aux(row):
    df.loc[df['SETOR_ATIV']==row['COD'], 'SETOR_ATIV'] = row['COD_UNIF']

# Esta variável out não é utilizada para nada além de evitar um
# monte de output que não será utilizado e que é gerado pelo método apply.
out = df_setor.apply(setor_aux, axis=1)

# Substituindo valor 0 por None
df.loc[df['SETOR_ATIV']==0, 'SETOR_ATIV'] = None

# Verificando intervalo de valores - condições:
    # "SETOR_ATIV >= 1" E "SETOR_ATIV <= 9"
verifica_range(df, 'SETOR_ATIV', 1, 9)
log_output.info('\n\n=====\\n')

return df


def passo_ren_ind(passo, df, deflator):
    """
    Corriga a renda individual de acordo com o deflator passado na função.
    :param passo: Número do passo atual para registro/log
    :param df: dataframe
    :param deflator: Deflator utilizado para correção da renda.
    :return: retorna o dataframe com as devidas modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - REN_IND")

    df['REN_IND'] = df['REN_IND'] * deflator

    return df


def passo_cd_renind(passo, df):
    """
    Substituir valores da coluna "CD_RENIND"

    * Substituir todos valores **3** por None
    * Substituir todos valores **2** por **0**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Tem renda
    2/Não tem renda
    3/Não respondeu

    ##### Categorias novas
    Valor/Descrição
    ----/-----
    0    /Não tem renda
    1    /Tem renda
    """

```

```

[Teste: Checar se existe algum número < 0 ou > 1.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com as devidas modificações
"""
log_tela.info("### PASSO " + str(passo) + " - CD_RENIND")

df.loc[df['CD_RENIND']==3,'CD_RENIND'] = None
df.loc[df['CD_RENIND']==2,'CD_RENIND'] = 0

# Verificando intervalo de valores - condições:
# "CD_RENIND >= 0" E "CD_RENIND <= 1"
verifica_dummy(df, 'CD_RENIND')
log_output.info('\n\n=====\\n')

return df

def passo_estuda(passo, df):
"""
Se zona da escola for zero (0) então não estuda (0), senão, estuda (1)

#####
#      ATENÇÃO      #
# ESTA FUNÇÃO SÓ DEVE SER #
# EXECUTADA APÓS A GERAÇÃO#
# DE TODOS OS ID'S, POIS #
# HÁ UMA DESCONFIANÇA   #
# QUANTO À QUALIDADE    #
#      DESTE DADO       #
#####

* Substituir todos valores **1** por **0**
* Substituir todos valores **2** por **1**
* Substituir todos valores **3** por **1**
* Substituir todos valores **4** por **1**
* Substituir todos valores **5** por **1**
* Substituir todos valores **6** por **1**

#### Categorias anteriores
Valor/Descrição
----/----
1/Não
2/Creche/Pré-Escola
3/1º Grau/Fundamental
4/2º Grau/Médio
5/Superior/Universitário
6/Outros

#### Categorias novas
Valor/Descrição
----/----
0/Não estuda

```

1 / Estuda

```
[Teste: Checar se existe algum número diferente de 0 ou 1.  
Se encontrar, retornar erro indicando em qual linha.]  
:param passo: Número do passo atual para registro/log  
:param df:  
:return: retorna dataframe com campo ESCOLA resolvido  
"""  
log_tela.info("### PASSO " + str(passo) + " - ESTUDA")  
  
# Substitui 1 por 0  
df.loc[df['ESTUDA'] == 1, 'ESTUDA'] = 0  
# Substitui 2 por 1  
df.loc[df['ESTUDA'] == 2, 'ESTUDA'] = 1  
# Substitui 3 por 1  
df.loc[df['ESTUDA'] == 3, 'ESTUDA'] = 1  
# Substitui 4 por 1  
df.loc[df['ESTUDA'] == 4, 'ESTUDA'] = 1  
# Substitui 5 por 1  
df.loc[df['ESTUDA'] == 5, 'ESTUDA'] = 1  
# Substitui 6 por 1  
df.loc[df['ESTUDA'] == 6, 'ESTUDA'] = 1  
  
# Substituindo todos que declararam zona escola diferente de zero  
# com campo ESTUDA igual a 1.  
df.loc[(df['ZONA_ESC'] != 0)&  
       (df['ZONA_ESC'].notnull()), 'ESTUDA'] = 1  
  
verifica_dummy(df, 'ESTUDA')  
log_output.info('\n\n=====\\n')  
  
return df
```

2.6 Funções referentes da Viagem

Função/Variável	Status
passo_f_vitag	OK
passo_fe_vitag	OK
passo_zona_esc	OK
passo_subzona_esc	OK
passo_mun_esc	OK
passo_zona_trab1	OK
passo_subzona_trab1	OK
passo_mun_trab1	OK
passo_zona_trab2	OK
passo_subzona_trab2	OK
passo_mun_trab2	OK
passo_zona_orig	OK
passo_subzona_orig	OK
passo_mun_orig	OK
passo_zona_dest	OK
passo_subzona_dest	OK
passo_mun_dest	OK
passo_serv_pas_orig	OK
passo_serv_pas_dest	OK
passo_motivo_orig	OK
passo_motivo_dest	OK
passo_modo1	OK
passo_modo2	OK
passo_modo3	OK
passo_modo4	OK
passo_modo_prin	OK
passo_tipo_est_auto	OK
passo_valor_est_auto	OK
passo_tot_vitag	OK

Obs: O passo_tot_vitag só deve ser executado após a produção dos ID's e NO's.

```
In [ ]: log_tela.info('Definindo funções referentes às viagens')
log_output.info('\n'=====\\n')
```

```
def passo_f_vitag(passo, df):
```

```

"""
Checar se existe algum erro na coluna "F_VIAG"

# #####Categorias
# Valor/Descrição
# ----/----
# 0/Não fez viagem
# 1/Fez viagem

[Teste: Checar se existe algum número diferente de 0 ou 1.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com F_VIAG modificado
"""

log_tela.info("### PASSO " + str(passo) + " - F_VIAG")

# Setando F_VIAG=1 para todos os casos
df['F_VIAG'] = 1

# Definindo F_VIAG=0 se DURACAO==0
df.loc[df['DURACAO']==0, 'F_VIAG'] = 0

verifica_dummy(df, 'F_VIAG')
log_output.info('\n\n=====\\n')

return df


def passo_fe_viag(passo, df):
"""
# Nada há que se fazer em relação aos dados da coluna "FE_VIAG"
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""

log_tela.info("### PASSO " + str(passo) + " - FE_VIAG")
log_output.info('\n\n=====\\n')

return df


def passo_zona_esc(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 460

[Teste: Checar se existe algum número < 1 ou > 460.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""

```

```

"""
log_tela.info("### PASSO " + str(passo) + " - ZONA_ESC")

# Substituindo valor 0 por None
df.loc[df['ZONA_ESC']==0,'ZONA_ESC'] = None

# Verificando intervalo de valores - condições:
# "ZONA_ESC >= 1" E "ZONA_ESC <= 460"
verifica_range(df, 'ZONA_ESC', 1, 460)
log_output.info('\n\n=====\\n')

return df
"""

def passo_subzona_esc(passo, df):
    """
    A variável SUBZONA_ESC ficará vazia (NA) em 2007 pois este ano já contém os dados de coordenadas.

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ESC")

    df['SUBZONA_ESC'] = None

    log_output.info('\n\n=====\\n')

    return df

def passo_mun_esc(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ESC")

    # Substituindo valor 0 por None
    df.loc[df['MUN_ESC']==0,'MUN_ESC'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_ESC >= 1" E "MUN_ESC <= 39"
    verifica_range(df, 'MUN_ESC', 1, 39)
    log_output.info('\n\n=====\\n')

```

```

    return df

def passo_zona_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 460

    [Teste: Checar se existe algum número < 1 ou > 460.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB1")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_TRAB1']==0,'ZONA_TRAB1'] = None

    # Verificando intervalo de valores - condições:
    # "ZONA_TRAB1 >= 1" E "ZONA_TRAB1 <= 460"
    verifica_range(df, 'ZONA_TRAB1', 1, 460)
    log_output.info('\n\n=====\\n')

    return df

def passo_subzona_trab1(passo, df):
    """
    A variável SUBZONA_TRAB1 ficará vazia (NA) em 2007 pois este ano já contém
    os dados de coordenadas.

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB1")

    df['SUBZONA_TRAB1'] = None

    log_output.info('\n\n=====\\n')

    return df

def passo_mun_trab1(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

```

```

[Teste: Checar se existe algum número < 1 ou > 39.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""

log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB1")

# Substituindo valor 0 por None
df.loc[df['MUN_TRAB1']==0,'MUN_TRAB1'] = None

# Verificando intervalo de valores - condições:
# "MUN_TRAB1 >= 1" E "MUN_TRAB1 <= 39"
verifica_range(df, 'MUN_TRAB1', 1, 39)
log_output.info('\n\n=====\\n')

return df


def passo_zona_trab2(passo, df):
"""
Checar se existe algum erro

# #####Categorias:
# > 1 a 460

[Teste: Checar se existe algum número < 1 ou > 460.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""

log_tela.info("### PASSO " + str(passo) + " - ZONA_TRAB2")

# Substituindo valor 0 por None
df.loc[df['ZONA_TRAB2']==0,'ZONA_TRAB2'] = None

# Verificando intervalo de valores - condições:
# "ZONA_TRAB2 >= 1" E "ZONA_TRAB2 <= 460"
verifica_range(df, 'ZONA_TRAB2', 1, 460)
log_output.info('\n\n=====\\n')

return df


def passo_subzona_trab2(passo, df):
"""
A variável SUBZONA_TRAB2 ficará vazia (NA) em 2007 pois este ano já contém
os dados de coordenadas.

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações

```

```

"""
log_tela.info("### PASSO " + str(passo) + " - SUBZONA_TRAB2")
df['SUBZONA_TRAB2'] = None
log_output.info('\n\n=====\\n')
return df

def passo_mun_trab2(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_TRAB2")

    # Substituindo valor 0 por None
    df.loc[df['MUN_TRAB2']==0, 'MUN_TRAB2'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_TRAB2 >= 1" E "MUN_TRAB2 <= 39"
    verifica_range(df, 'MUN_TRAB2', 1, 39)
    log_output.info('\n\n=====\\n')

    return df

def passo_zona_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias:
    # > 1 a 460

    [Teste: Checar se existe algum número < 1 ou > 460.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - ZONA_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['ZONA_ORIG']==0, 'ZONA_ORIG'] = None

    # Verificando intervalo de valores - condições:

```

```

# "ZONA_ORIG >= 1" E "ZONA_ORIG <= 460"
verifica_range(df, 'ZONA_ORIG', 1, 460)
log_output.info('\'\n\'=====\'\n')

return df

def passo_subzona_orig(passo, df):
    """
    A variável SUBZONA_ORIG ficará vazia (NA) em 2007 pois este ano já contém os dados de coordenadas.

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - SUBZONA_ORIG")

    df['SUBZONA_ORIG'] = None

    log_output.info('\'\n\'=====\'\n')

    return df

def passo_mun_orig(passo, df):
    """
    Checar se existe algum erro

    # #####Categorias
    # > 1 a 39

    [Teste: Checar se existe algum número < 1 ou > 39.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: retorna o dataframe sem modificações
    """
    log_tela.info("### PASSO " + str(passo) + " - MUN_ORIG")

    # Substituindo valor 0 por None
    df.loc[df['MUN_ORIG']==0, 'MUN_ORIG'] = None

    # Verificando intervalo de valores - condições:
    # "MUN_ORIG >= 1" E "MUN_ORIG <= 39"
    verifica_range(df, 'MUN_ORIG', 1, 39)
    log_output.info('\'\n\'=====\'\n')

    return df

def passo_zona_dest(passo, df):
    """
    Checar se existe algum erro

```

```

# #####Categorias:
# > 1 a 460

[Teste: Checar se existe algum número < 1 ou > 460.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""
log_tela.info("### PASSO " + str(passo) + " - ZONA_DEST")

# Substituindo valor 0 por None
df.loc[df['ZONA_DEST']==0,'ZONA_DEST'] = None

# Verificando intervalo de valores - condições:
# "ZONA_DEST >= 1" E "ZONA_DEST <= 460"
verifica_range(df, 'ZONA_DEST', 1, 460)
log_output.info('\n\n=====\\n')

return df

def passo_subzona_dest(passo, df):
"""
A variável SUBZONA_DEST ficará vazia (NA) em 2007 pois este ano já contém
os dados de coordenadas.

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""
log_tela.info("### PASSO " + str(passo) + " - SUBZONA_DESTM")

df['SUBZONA_DEST'] = None

log_output.info('\n\n=====\\n')

return df

def passo_mun_dest(passo, df):
"""
Checar se existe algum erro

# #####Categorias
# > 1 a 39

[Teste: Checar se existe algum número < 1 ou > 39.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""

```

```

log_tela.info("### PASSO " + str(passo) + " - MUN_DEST")

# Substituindo valor 0 por None
df.loc[df['MUN_DEST']==0,'MUN_DEST'] = None

# Verificando intervalo de valores - condições:
# "MUN_DEST >= 1" E "MUN_DEST <= 39"
verifica_range(df, 'MUN_DEST', 1, 39)
log_output.info('\n\n=====\\n')

return df

def passo_serv_pas_orig(passo, df):
    """
    Substituir **0** por None
    Substituir **2** por **0**

    #####Categorias antigas
    Valor/Descrição
    ----/----
    0/Não Respondido
    1/Sim
    2/Não

    #####Categorias novas
    Valor/Descrição
    ----/----
    0/Não
    1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""
log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_ORIG")

df.loc[df['SERV_PAS_ORIG']==0,'SERV_PAS_ORIG'] = None
df.loc[df['SERV_PAS_ORIG']==2,'SERV_PAS_ORIG'] = 0

verifica_dummy(df, 'SERV_PAS_ORIG')
log_output.info('\n\n=====\\n')

return df

def passo_serv_pas_dest(passo, df):
    """
    Substituir **0** por None
    Substituir **2** por **0**

    #####Categorias antigas
    Valor/Descrição
    ----/----

```

```

0/Não Respondido
1/Sim
2/Não

####Categorias novas
Valor/Descrição
----/----
0/Não
1/Sim

:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe sem modificações
"""
log_tela.info("### PASSO " + str(passo) + " - SERV_PAS_DEST")

df.loc[df['SERV_PAS_DEST']==0,'SERV_PAS_DEST'] = None
df.loc[df['SERV_PAS_DEST']==2,'SERV_PAS_DEST'] = 0

verifica_dummy(df, 'SERV_PAS_DEST')
log_output.info('\n\n=====\\n')

return df

def passo_motivo_orig(passo, df):
"""

* Substituir todos valores **10** por **9**
* Substituir todos valores **0** por **None**

#### Categorias anteriores
Valor/Descrição
----/----
1/Trabalho/Indústria
2/Trabalho/Comércio
3/Trabalho/Serviços
4/Educação
5/Compras
6/Saúde
7/Lazer
8/Residência
9/Procurar Emprego
10/Assuntos Pessoais

#### Categorias novas
Valor/Descrição
----/----
1/Trabalho/Indústria
2/Trabalho/Comércio
3/Trabalho/Serviços
4/Educação
5/Compras
6/Saúde

```

```

7/Lazer
8/Residência
9/Doutros

[Teste: Checar se existe algum número < 1 ou > 9.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - MOTIVO_ORIG")

# Substitui 10 por 9
df.loc[df['MOTIVO_ORIG'] == 10, 'MOTIVO_ORIG'] = 9
# Substitui 0 por None
df.loc[df['MOTIVO_ORIG'] == 0, 'MOTIVO_ORIG'] = None

# Verificando intervalo de valores - condições:
# "MOTIVO_ORIG >= 1" E "MOTIVO_ORIG <= 9"
verifica_range(df, 'MOTIVO_ORIG', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_motivo_dest(passo, df):
"""
* Substituir todos valores **10** por **9**
* Substituir todos valores **0** por **None**

#### Categorias anteriores
Valor/Descrição
----/----
1/Trabalho/Indústria
2/Trabalho/Comércio
3/Trabalho/Serviços
4/Educação
5/Compras
6/Saúde
7/Lazer
8/Residência
9/Procurar Emprego
10/Assuntos Pessoais

#### Categorias novas
Valor/Descrição
----/----
1/Trabalho/Indústria
2/Trabalho/Comércio
3/Trabalho/Serviços
4/Educação
5/Compras
6/Saúde

```

```

7/Lazer
8/Residência
9/Doutros

[Teste: Checar se existe algum número < 1 ou > 9.
 Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: dataframe modificado
"""
log_tela.info("### PASSO " + str(passo) + " - MOTIVO_DEST")

# Substitui 10 por 9
df.loc[df['MOTIVO_DEST'] == 10, 'MOTIVO_DEST'] = 9
# Substitui 10 por 9
df.loc[df['MOTIVO_DEST'] == 0, 'MOTIVO_DEST'] = None

# Verificando intervalo de valores - condições:
# "MOTIVO_DEST >= 1" E "MOTIVO_DEST <= 9"
verifica_range(df, 'MOTIVO_DEST', 1, 9)
log_output.info('\n\n=====\\n')

return df

def passo_modo1(passo, df):
"""
Substituir valores da coluna "MOD01"

* Substituir todos valores **2** por **1**
* Substituir todos valores **3** por **1**
* Substituir todos valores **4** por **2**
* Substituir todos valores **5** por **2**
* Substituir todos valores **6** por **3**
* Substituir todos valores **7** por **4**
* Substituir todos valores **8** por **5**
* Substituir todos valores **9** por **6**
* Substituir todos valores **10** por **6**
* Substituir todos valores **11** por **6**
* Substituir todos valores **12** por **7**
* Substituir todos valores **13** por **8**
* Substituir todos valores **14** por **9**
* Substituir todos valores **15** por **10**
* Substituir todos valores **16** por **11**
* Substituir todos valores **17** por **12**
* Substituir todos valores **0** por **None**

#### Categorias anteriores
Valor/Descrição
----/----
1/Onibus Municipio S.Paulo
2/Onibus Outros Municipios
3/Onibus Metropolitano
4/Onibus Fretado

```

```

5/Escolar
6/Dirigindo Automóvel
7/Passageiro de Automóvel
8/Taxi
9/Microônibus/Van Município de S.Paulo
10/Microônibus/Van Outros Municípios
11/Microônibus/Van Metropolitano
12/Metrô
13/Trem
14/Moto
15/Bicicleta
16/A Pé
17/Outros

```

```

#### Categorias novas
Valor/Descrição
----/----
1/Ônibus
2/Ônibus Escolar / Empresa
3/Dirigindo Automóvel
4/Passageiro de Automóvel
5/Táxi
6/Lotação / Peruá / Van / Microônibus
7/Metrô
8/Trem
9/Moto
10/Bicicleta
11/A Pé
12/Outros

```

```

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com as devidas modificações
"""
log_tela.info("### PASSO " + str(passo) + " - MODO1")

df.loc[df['MODO1']==2,'MODO1'] = 1
df.loc[df['MODO1']==3,'MODO1'] = 1
df.loc[df['MODO1']==4,'MODO1'] = 2
df.loc[df['MODO1']==5,'MODO1'] = 2
df.loc[df['MODO1']==6,'MODO1'] = 3
df.loc[df['MODO1']==7,'MODO1'] = 4
df.loc[df['MODO1']==8,'MODO1'] = 5
df.loc[df['MODO1']==9,'MODO1'] = 6
df.loc[df['MODO1']==10,'MODO1'] = 6
df.loc[df['MODO1']==11,'MODO1'] = 6
df.loc[df['MODO1']==12,'MODO1'] = 7
df.loc[df['MODO1']==13,'MODO1'] = 8
df.loc[df['MODO1']==14,'MODO1'] = 9
df.loc[df['MODO1']==15,'MODO1'] = 10
df.loc[df['MODO1']==16,'MODO1'] = 11
df.loc[df['MODO1']==17,'MODO1'] = 12

```

```

df.loc[df['MODO1']==0,'MODO1'] = None

# Verificando intervalo de valores - condições:
# "MODO1 >= 1" E "MODO1 <= 12"
verifica_range(df, 'MODO1', 1, 12)
log_output.info('\n\n=====\\n')

return df


def passo_modo2(passo, df):
    """
    Substituir valores da coluna "MODO2"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **1**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **5** por **2**
    * Substituir todos valores **6** por **3**
    * Substituir todos valores **7** por **4**
    * Substituir todos valores **8** por **5**
    * Substituir todos valores **9** por **6**
    * Substituir todos valores **10** por **6**
    * Substituir todos valores **11** por **6**
    * Substituir todos valores **12** por **7**
    * Substituir todos valores **13** por **8**
    * Substituir todos valores **14** por **9**
    * Substituir todos valores **15** por **10**
    * Substituir todos valores **16** por **11**
    * Substituir todos valores **17** por **12**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Onibus Municipio S.Paulo
    2/Onibus Outros Municipios
    3/Onibus Metropolitano
    4/Onibus Fretado
    5/Escolar
    6/Dirigindo Automóvel
    7/Passageiro de Automóvel
    8/Taxi
    9/Microônibus/Van Município de S.Paulo
    10/Microônibus/Van Outros Municípios
    11/Microônibus/Van Metropolitano
    12/Metrô
    13/Trem
    14/Moto
    15/Bicicleta
    16/A Pé
    17/Outros

    ##### Categorias novas

```

Valor/Descrição

----/----

1/*Ônibus*
 2/*Ônibus Escolar / Empresa*
 3/*Dirigindo Automóvel*
 4/*Passageiro de Automóvel*
 5/*Táxi*
 6/*Lotação / Peruá / Van / Microônibus*
 7/*Metrô*
 8/*Trem*
 9/*Moto*
 10/*Bicicleta*
 11/*A Pé*
 12/*Outros*

*[Teste: Checar se existe algum número < 1 ou > 12.
 Se encontrar, retornar erro indicando em qual linha.]*

```
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com as devidas modificações
"""

log_tela.info("### PASSO " + str(passo) + " - MODO1")

df.loc[df['MODO2']==2,'MODO2'] = 1
df.loc[df['MODO2']==3,'MODO2'] = 1
df.loc[df['MODO2']==4,'MODO2'] = 2
df.loc[df['MODO2']==5,'MODO2'] = 2
df.loc[df['MODO2']==6,'MODO2'] = 3
df.loc[df['MODO2']==7,'MODO2'] = 4
df.loc[df['MODO2']==8,'MODO2'] = 5
df.loc[df['MODO2']==9,'MODO2'] = 6
df.loc[df['MODO2']==10,'MODO2'] = 6
df.loc[df['MODO2']==11,'MODO2'] = 6
df.loc[df['MODO2']==12,'MODO2'] = 7
df.loc[df['MODO2']==13,'MODO2'] = 8
df.loc[df['MODO2']==14,'MODO2'] = 9
df.loc[df['MODO2']==15,'MODO2'] = 10
df.loc[df['MODO2']==16,'MODO2'] = 11
df.loc[df['MODO2']==17,'MODO2'] = 12
df.loc[df['MODO2']==0,'MODO2'] = None

# Verificando intervalo de valores - condições:
# "MODO2 >= 1" E "MODO2 <= 12"
verifica_range(df, 'MODO2', 1, 12)
log_output.info('\n\n=====\\n')

return df

def passo_modo3(passo, df):
"""
Substituir valores da coluna "MODO3"

* Substituir todos valores **2** por **1**

```

* Substituir todos valores **3** por **1**
* Substituir todos valores **4** por **2**
* Substituir todos valores **5** por **2**
* Substituir todos valores **6** por **3**
* Substituir todos valores **7** por **4**
* Substituir todos valores **8** por **5**
* Substituir todos valores **9** por **6**
* Substituir todos valores **10** por **6**
* Substituir todos valores **11** por **6**
* Substituir todos valores **12** por **7**
* Substituir todos valores **13** por **8**
* Substituir todos valores **14** por **9**
* Substituir todos valores **15** por **10**
* Substituir todos valores **16** por **11**
* Substituir todos valores **17** por **12**
* Substituir todos valores **0** por **None**

Categorias anteriores

Valor/Descrição

----/----

- 1/Ônibus Município S.Paulo
- 2/Ônibus Outros Municípios
- 3/Ônibus Metropolitano
- 4/Ônibus Fretado
- 5/Escolar
- 6/Dirigindo Automóvel
- 7/Passageiro de Automóvel
- 8/Taxi
- 9/Microônibus/Van Município de S.Paulo
- 10/Microônibus/Van Outros Municípios
- 11/Microônibus/Van Metropolitano
- 12/Metrô
- 13/Trem
- 14/Moto
- 15/Bicicleta
- 16/A Pé
- 17/Outros

Categorias novas

Valor/Descrição

----/----

- 1/Ônibus
- 2/Ônibus Escolar / Empresa
- 3/Dirigindo Automóvel
- 4/Passageiro de Automóvel
- 5/Táxi
- 6/Lotação / Peruá / Van / Microônibus
- 7/Metrô
- 8/Trem
- 9/Moto
- 10/Bicicleta
- 11/A Pé
- 12/Outros

```

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com as devidas modificações
"""

log_tela.info("### PASSO " + str(passo) + " - MODO1")

df.loc[df['MODO3']==2,'MODO3'] = 1
df.loc[df['MODO3']==3,'MODO3'] = 1
df.loc[df['MODO3']==4,'MODO3'] = 2
df.loc[df['MODO3']==5,'MODO3'] = 2
df.loc[df['MODO3']==6,'MODO3'] = 3
df.loc[df['MODO3']==7,'MODO3'] = 4
df.loc[df['MODO3']==8,'MODO3'] = 5
df.loc[df['MODO3']==9,'MODO3'] = 6
df.loc[df['MODO3']==10,'MODO3'] = 6
df.loc[df['MODO3']==11,'MODO3'] = 6
df.loc[df['MODO3']==12,'MODO3'] = 7
df.loc[df['MODO3']==13,'MODO3'] = 8
df.loc[df['MODO3']==14,'MODO3'] = 9
df.loc[df['MODO3']==15,'MODO3'] = 10
df.loc[df['MODO3']==16,'MODO3'] = 11
df.loc[df['MODO3']==17,'MODO3'] = 12
df.loc[df['MODO3']==0,'MODO3'] = None

# Verificando intervalo de valores - condições:
# "MODO3 >= 1" E "MODO3 <= 12"
verifica_range(df, 'MODO3', 1, 12)
log_output.info('\'\n\'=====\\n')

return df

def passo_modo4(passo, df):
    """
    Substituir valores da coluna "MODO4"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **1**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **5** por **2**
    * Substituir todos valores **6** por **3**
    * Substituir todos valores **7** por **4**
    * Substituir todos valores **8** por **5**
    * Substituir todos valores **9** por **6**
    * Substituir todos valores **10** por **6**
    * Substituir todos valores **11** por **6**
    * Substituir todos valores **12** por **7**
    * Substituir todos valores **13** por **8**
    * Substituir todos valores **14** por **9**
    * Substituir todos valores **15** por **10**
    * Substituir todos valores **16** por **11**
    * Substituir todos valores **17** por **12**

```

* Substituir todos valores **0** por **None**

```
#### Categorias anteriores
Valor/Descrição
----/----
1/Ônibus Município S.Paulo
2/Ônibus Outros Municípios
3/Ônibus Metropolitano
4/Ônibus Fretado
5/Escolar
6/Dirigindo Automóvel
7/Passageiro de Automóvel
8/Taxi
9/Microônibus/Van Município de S.Paulo
10/Microônibus/Van Outros Municípios
11/Microônibus/Van Metropolitano
12/Metrô
13/Trem
14/Moto
15/Bicicleta
16/A Pé
17/Outros
```

```
#### Categorias novas
Valor/Descrição
----/----
1/Ônibus
2/Ônibus Escolar / Empresa
3/Dirigindo Automóvel
4/Passageiro de Automóvel
5/Táxi
6/Lotação / Peruá / Van / Microônibus
7/Metrô
8/Trem
9/Moto
10/Bicicleta
11/A Pé
12/Outros
```

[Teste: Checar se existe algum número < 1 ou > 12.

Se encontrar, retornar erro indicando em qual linha.]

```
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com as devidas modificações
"""
log_tela.info("### PASSO " + str(passo) + " - MODO1")
```

```
df.loc[df['MODO4']==2,'MODO4'] = 1
df.loc[df['MODO4']==3,'MODO4'] = 1
df.loc[df['MODO4']==4,'MODO4'] = 2
df.loc[df['MODO4']==5,'MODO4'] = 2
df.loc[df['MODO4']==6,'MODO4'] = 3
df.loc[df['MODO4']==7,'MODO4'] = 4
df.loc[df['MODO4']==8,'MODO4'] = 5
```

```

df.loc[df['MODO4']==9,'MODO4'] = 6
df.loc[df['MODO4']==10,'MODO4'] = 6
df.loc[df['MODO4']==11,'MODO4'] = 6
df.loc[df['MODO4']==12,'MODO4'] = 7
df.loc[df['MODO4']==13,'MODO4'] = 8
df.loc[df['MODO4']==14,'MODO4'] = 9
df.loc[df['MODO4']==15,'MODO4'] = 10
df.loc[df['MODO4']==16,'MODO4'] = 11
df.loc[df['MODO4']==17,'MODO4'] = 12
df.loc[df['MODO4']==0,'MODO4'] = None

# Verificando intervalo de valores - condições:
    # "MODO4 >= 1" E "MODO4 <= 12"
verifica_range(df, 'MODO4', 1, 12)
log_output.info(' \n\n===== \n')

return df

def passo_modo_prin(passo, df):
    """
    Substituir valores da coluna "MODO_PRIN"

    * Substituir todos valores **2** por **1**
    * Substituir todos valores **3** por **1**
    * Substituir todos valores **4** por **2**
    * Substituir todos valores **5** por **2**
    * Substituir todos valores **6** por **3**
    * Substituir todos valores **7** por **4**
    * Substituir todos valores **8** por **5**
    * Substituir todos valores **9** por **6**
    * Substituir todos valores **10** por **6**
    * Substituir todos valores **11** por **6**
    * Substituir todos valores **12** por **7**
    * Substituir todos valores **13** por **8**
    * Substituir todos valores **14** por **9**
    * Substituir todos valores **15** por **10**
    * Substituir todos valores **16** por **11**
    * Substituir todos valores **17** por **12**
    * Substituir todos valores **0** por **None**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    1/Onibus Municipio S.Paulo
    2/Onibus Outros Municipios
    3/Onibus Metropolitano
    4/Onibus Fretado
    5/Escolar
    6/Dirigindo Automóvel
    7/Passageiro de Automóvel
    8/Taxi
    9/Microônibus/Van Município de S.Paulo
    10/Microônibus/Van Outros Municípios

```

```

11/Microônibus/Van Metropolitano
12/Metrô
13/Trem
14/Moto
15/Bicicleta
16/A Pé
17/Outros

##### Categorias novas
Valor/Descrição
----/----
1/Ônibus
2/Ônibus Escolar / Empresa
3/Dirigindo Automóvel
4/Passageiro de Automóvel
5/Táxi
6/Lotação / Peruá / Van / Microônibus
7/Metrô
8/Trem
9/Moto
10/Bicicleta
11/A Pé
12/Outros

[Teste: Checar se existe algum número < 1 ou > 12.
Se encontrar, retornar erro indicando em qual linha.]
```

:param passo: Número do passo atual para registro/log

:param df:

:return: retorna o dataframe com as devidas modificações

"""

```

log_tela.info("### PASSO " + str(passo) + " - MODO_PRIN")

df.loc[df['MODO_PRIN']==2, 'MODO_PRIN'] = 1
df.loc[df['MODO_PRIN']==3, 'MODO_PRIN'] = 1
df.loc[df['MODO_PRIN']==4, 'MODO_PRIN'] = 2
df.loc[df['MODO_PRIN']==5, 'MODO_PRIN'] = 2
df.loc[df['MODO_PRIN']==6, 'MODO_PRIN'] = 3
df.loc[df['MODO_PRIN']==7, 'MODO_PRIN'] = 4
df.loc[df['MODO_PRIN']==8, 'MODO_PRIN'] = 5
df.loc[df['MODO_PRIN']==9, 'MODO_PRIN'] = 6
df.loc[df['MODO_PRIN']==10, 'MODO_PRIN'] = 6
df.loc[df['MODO_PRIN']==11, 'MODO_PRIN'] = 6
df.loc[df['MODO_PRIN']==12, 'MODO_PRIN'] = 7
df.loc[df['MODO_PRIN']==13, 'MODO_PRIN'] = 8
df.loc[df['MODO_PRIN']==14, 'MODO_PRIN'] = 9
df.loc[df['MODO_PRIN']==15, 'MODO_PRIN'] = 10
df.loc[df['MODO_PRIN']==16, 'MODO_PRIN'] = 11
df.loc[df['MODO_PRIN']==17, 'MODO_PRIN'] = 12
df.loc[df['MODO_PRIN']==0, 'MODO_PRIN'] = None

# Verificando intervalo de valores - condições:
# "MODO_PRIN >= 1" E "MODO_PRIN <= 12"
verifica_range(df, 'MODO_PRIN', 1, 12)
log_output.info('\'\n\'=====\\n')
```

```

    return df

def passo_tipo_vitag(passo, df):
    """
    * Substituir os valores **0** por **None**

    # #####Categorias novas
    # Valor/Descrição
    # ----/----
    # 1/Coletivo
    # 2/Individual
    # 3/A pé

    [Teste: Checar se existe algum número < 1 ou > 3.
     Se encontrar, retornar erro indicando em qual linha.]
    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno

    :param passo: Número do passo atual para registro/log
    :param df:
    :return: sem retorno
    """
    log_tela.info("### PASSO " + str(passo) + " - TIPO_VIAG")

    # Substituindo valor 0 por None
    df.loc[df['TIPO_VIAG']==0,'TIPO_VIAG'] = None

    # Verificando intervalo de valores - condições:
    # "MODO_PRIN >= 1" E "MODO_PRIN <= 3"
    verifica_range(df, 'TIPO_VIAG', 1, 3)

    log_output.info('\n\n=====\\n')
    return df


def passo_tipo_est_auto(passo, df):
    """
    * Substituir todos valores **0** por None (NA)
    * Substituir todos valores **8** por None (NA)
    * Substituir todos valores **1** por **0**
    * Substituir todos valores **3** por **1**
    * Substituir todos valores **4** por **1**
    * Substituir todos valores **6** por **1**
    * Substituir todos valores **7** por **1**
    * Substituir todos valores **5** por **2**

    ##### Categorias anteriores
    Valor/Descrição
    ----/----
    0/Não Respondeu
    1/Não Estacionou
    2/Zona Azul/Zona Marrom

```

```

3/Estacionamento Patrocinado
4/Estacionamento Próprio
5/Meio-Fio
6/Avulso
7/Mensal
8/Não Respondeu

##### Categorias novas
Valor/Descrição
----/----
0/Não Estacionou
1/Estacionamento Particular (Avulso / Mensal / Próprio / Patrocinado)
2/Estacionamento Público (meio fio / zona azul / zona marrom / parquímetro)

[Teste: Checar se existe algum número < 0 ou > 5.
Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna dataframe modificado
"""

log_tela.info("### PASSO " + str(passo) + " - TIPO_EST_AUTO")

# Substituindo valor 0 por None
df.loc[df['TIPO_EST_AUTO']==0,'TIPO_EST_AUTO'] = None
# Substituindo valor 8 por None
df.loc[df['TIPO_EST_AUTO']==8,'TIPO_EST_AUTO'] = None
# Substituindo valor 1 por 0
df.loc[df['TIPO_EST_AUTO']==1,'TIPO_EST_AUTO'] = 0
# Substituindo valor 3 por 1
df.loc[df['TIPO_EST_AUTO']==3,'TIPO_EST_AUTO'] = 1
# Substituindo valor 4 por 1
df.loc[df['TIPO_EST_AUTO']==4,'TIPO_EST_AUTO'] = 1
# Substituindo valor 6 por 1
df.loc[df['TIPO_EST_AUTO']==6,'TIPO_EST_AUTO'] = 1
# Substituindo valor 7 por 1
df.loc[df['TIPO_EST_AUTO']==7,'TIPO_EST_AUTO'] = 1
# Substituindo valor 5 por 2
df.loc[df['TIPO_EST_AUTO']==5,'TIPO_EST_AUTO'] = 2

verifica_range(df, 'TIPO_EST_AUTO', 0, 2)
log_output.info('\n\n=====\\n')

return df

def passo_valor_est_auto(passo, df, deflator):
"""
Corrigir o valor da variável com o deflator passado.

:param passo: Número do passo atual para registro/log
:param df: dataframe a ser modificado
:return: dataframe com o VALOR_EST_AUTO corrigido pelo deflator
"""

```

```

log_tela.info("### PASSO " + str(passo) + " - VALOR_EST_AUTO")

df['VALOR_EST_AUTO'] = df['VALOR_EST_AUTO'] * deflator

log_output.info('\n\n=====\\n')

return df

def passo_tot_viag(passo, df):
    """
    #####
    # ATENÇÃO #
    # ESTA FUNÇÃO SÓ DEVE SER #
    # EXECUTADA APÓS A GERAÇÃO#
    # DE TODOS OS ID'S, POIS #
    # HÁ UMA DESCONFIANÇA #
    # QUANTO À QUALIDADE #
    # DESTE DADO #
    #####
    Calcula e confere o campo TOT_VIAG,
    baseado no maior valor de NO_VIAG para cada pessoa
:param passo: Número do passo atual para registro/log
:param df:
:return: retorna o dataframe com o "TOT_VIAG" corrigido
"""

log_tela.info("### PASSO " + str(passo) + " - TOT_VIAG")
log_tela.warning('\n\n#####\\n' +
                 'Este passo DEVE ser executado apenas após a \\n' +
                 'execução dos passos que geram os novos IDs\\n' +
                 '#####\\n' )

# 'Calculando' o máximo de viagens para cada indivíduo
# Agrupa por ID_PESS e encontra o máximo para NO_VIAG.
# O resultado é um objeto do tipo "Series" cujo index é
# ID_PESS e a variável é NO_VIAG. Em seguida transforma
# esse objeto num DataFrame e depois renomeia a coluna
# NO_VIAG para MAX_VIAG.
df_max = pd.DataFrame(
    df.groupby('ID_PESS')['NO_VIAG'].max()).rename(
        columns={'NO_VIAG': 'MAX_VIAG'})

# Criando um novo dataframe auxiliar apenas com ID_PESS, apenas para ficar mais leve

# O passo seguinte é atribuir o MAX_VIAG à coluna TOT_VIAG
# do dataframe df, linkando por ID_PESS. Isso é feito usando o
# método 'merge' da biblioteca pandas.
df['TOT_VIAG'] = pd.merge(df, df_max, how='left', left_on='ID_PESS',
                           right_index=True)['MAX_VIAG']

log_output.info('TOT_VIAG:\\n\\n' +
                '    Situação final dos dados: \\n' +
                str(df['TOT_VIAG'].describe()) + '\\n' +
                '    TOT_VIAG: Situação final dos dados: \\n' +
                str(df['TOT_VIAG'].value_counts()))

```

```

# Agora uma função que irá verificar se para todo "ID_PESS" o "TOT_VIAG"
# é igual ao 'NO_VIAG' máximo.
def verifica_no_viag_tot_viag(row):
    if row['NO_VIAG'] != row['TOT_VIAG']:
        log_output.warning('TOT_VIAG: Erro encontrado na linha '
                           + str(row) + ':\n'
                           + ' => ' + row
                           )
df.loc[:, ['ID_PESS', 'NO_VIAG', 'TOT_VIAG']]\
    .groupby('ID_PESS')\
    .agg({'NO_VIAG': 'max', 'ID_PESS': 'max', 'TOT_VIAG': 'max'})\
    .apply(verifica_no_viag_tot_viag, axis=1)
log_output.info('\n\n=====\\n')

return df

```

2.7 Funções que geram os “NO”s e os “ID”s

Função/Variável	Status
gera_nos_e_ids	Verificar NO VIAG

```
In [ ]: log_tela.info('Definindo funções que geram os "NO"s e os "ID"s')
log_output.info('\n\n=====\\n')
```

```
def gera_nos_e_ids(passo, df):
    """
    Esta função gera todos os IDs e todos os NOs das variáveis:
    DOM (domicílio), FAM (família), PESS (pessoa) e VIAG (viagem)
    A ordem de geração é:
    NO_DOM, ID_DOM, NO_FAM, ID_FAM, NO_PESS, ID_PESS, NO_VIAG, ID_VIAG
    Esta ordem é fixa pois cada uma dessas variáveis depende da geração
    da variável anterior.

    Os NOs são gerados como se fossem subíndices.
    No caso dos domicílios, eles são contabilizados dentro de cada zona.
    Assim, cada novo domicílio que aparece na listagem recebe um número
    que representa sua posição na sequência de domicílios dentro da zona
    na qual ele está contido.
    As famílias seguem o mesmo raciocínio, com relação ao domicílio, as
    pessoas com relação às famílias e as viagens com relação às pessoas.

    Deve-se apenas tomar cuidado com as viagens, pois existem pessoas que
    não realizaram viagens. Neste caso, estes registros devem contabilizar
    o NO_VIAG como zero, e não como 1. Para contemplar estes casos, vamos
    utilizar a variável F_VIAG, que representa quando há ou não viagem
    naquele registro (F_VIAG==1 tem viagem, F_VIAG==0 não tem viagem).
    Assim, o que precisamos fazer para calcular o NO_VIAG é agrupar os
    registros por pessoa (ID_PESS) e, dentro de cada agrupamento, somar
    o valor de F_VIAG registro a registro (linha por linha)
    cumulativamente. No final do processo, atribui-se zero a NO_VIAG
    quando F_VIAG for igual a zero.
    """

def gera_id_dom(row):
    """
    Gera o ID_DOM baseado no 'ANO', na 'ZONA_DOM' e no 'NO_DOM'
    O argumento passado é a "linha".
    Uso:
        df['ID_DOM'] = df.apply(gera_ID_DOM, axis=1)
    Retorna: ID_DOM da respectiva linha
    """
    # Formatando o ano para string
    ano = str(row['ANO'])

    # Formatando a zona para ficar como string com 3 caracteres
    zona = str('%03d' % row['ZONA_DOM'])
```

```

# Formatando no_dom para ficar como string com 4 caracteres
no_dom = str('%04d' % row['NO_DOM'])

# Retornando o número inteiro correspondente à string
# concatenada de ano + zona + no_dom
return ano + zona + no_dom


def gera_id_fam(row):
    """
    Gera o ID_FAM baseado no 'ID_DOM' e no 'NO_FAM'
    O argumento passado é a "linha".
    Uso:
        df['ID_FAM'] = df.apply(gera_ID_FAM, axis=1)
    Retorna: ID_FAM da respectiva linha
    """
    # Formatando id_dom como string
    id_dom = str(row['ID_DOM'])

    # Formatando no_fam como string para ficar com 2 caracteres
    no_fam = str('%02d' % row['NO_FAM'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_DOM com NO_FAM
    return id_dom + no_fam


def gera_id_pess(row):
    """
    Gera o ID_PESS baseado no 'ID_FAM' e no 'NO_PESS'
    O argumento passado é a "linha".
    Uso:
        df['ID_PESS'] = df.apply(gera_ID_PESS, axis=1)
    Retorna: ID_PESS da respectiva linha
    """
    # Formatando id_fam como string
    id_fam = str(row['ID_FAM'])

    # Formatando no_pess como string para ficar com 2 caracteres
    no_pess = str('%02d' % row['NO_PESS'])

    # Retornando o número inteiro correspondente à string
    # concatenada de ID_FAM com NO_PESS
    return id_fam + no_pess


def gera_id_viag(row):
    """
    Gera o ID_VIAG baseado no 'ID_PESS' e no 'NO_VIAG'
    O argumento passado é a "linha".
    Uso:
        df['ID_VIAG'] = df.apply(gera_ID_VIAG, axis=1)
    Retorna ID_VIAG da respectiva linha
    """
    # Formatando id_pess como string
    id_pess = str(row['ID_PESS'])

```

```

# Formatando no_viag como string para ficar com 2 caracteres
no_viag = str('%02d' % row['NO_VIAG'])

# Retornando o número inteiro correspondente à string
# concatenada de ID_PESS com NO_VIAG
return id_pess + no_viag

#gera NO_DOM
log_tela.info("Gerando NO_DOM")
df['NO_DOM'] = df.groupby('ZONA_DOM', sort=False)['F_DOM'].cumsum()
log_tela.info("NO_DOM gerado")
log_tela.info("Gerando ID_DOM")
#gera ID_DOM
df['ID_DOM'] = df.apply(gera_id_dom, axis=1)
log_tela.info("ID_DOM gerado")

#gera NO_FAM
log_tela.info("Gerando NO_FAM")
df['NO_FAM'] = df.groupby('ID_DOM', sort=False)['F_FAM'].cumsum()
log_tela.info("NO_FAM gerado")
log_tela.info("Gerando ID_FAM")
#gera ID_FAM
df['ID_FAM'] = df.apply(gera_id_fam, axis=1)
log_tela.info("ID_FAM gerado")

#gera NO_PESS
log_tela.info("Gerando NO_PESS")
df['NO_PESS'] = df.groupby('ID_FAM', sort=False)['F_PESS'].cumsum()
log_tela.info("NO_PESS gerado")
log_tela.info("Gerando ID_PESS")
#gera ID_PESS
df['ID_PESS'] = df.apply(gera_id_pess, axis=1)
log_tela.info("ID_PESS gerado")

#gera NO_VIAG
log_tela.info("Gerando NO_VIAG")
#df['NO_VIAG'] = 1
df['NO_VIAG'] = df.groupby('ID_PESS', sort=False)['F_VIAG'].cumsum()
# Verificando se FE_VIAG == 0 e zerando NO_VIAG nesse caso
df.loc[df['FE_VIAG'] == 0, 'NO_VIAG'] = 0
log_tela.info("NO_VIAG gerado")
log_tela.info("Gerando ID_VIAG")
#gera ID_VIAG
df['ID_VIAG'] = df.apply(gera_id_viag, axis=1)
log_tela.info("ID_VIAG gerado")

return df

```

2.8 Função relativa à entrevista

Função/Variável	Status
passo_cd_entre	OK

Obs: o passo_cd_entre deve ser executado após o passo_tot_viag.

```
In [ ]: log_tela.info('Definindo função relativa à entrevista')
log_output.info('\n'=====\\n')

def passo_cd_entre(passo, df):
    """
    #####
    #      ATENÇÃO      #
    # ESTA FUNÇÃO SÓ DEVE SER #
    # EXECUTADA APÓS A GERAÇÃO#
    #      DO TOT_VIAG     #
    #####
    ----
    Substituir valores da coluna "CD_ENTRE"
    Todas viagens são consideradas "completas", segundo informações do Metrô

    * sem viagem: se TOT_VIAG == 0
    * com viagem: se TOT_VIAG != 0

    #####Categorias anteriores
    | Valor | Descrição |
    | ----- | ----- |
    | 5 | Completa sem viagem |
    | 6 | Completa com viagem |

    #####Categorias novas
    | Valor | Descrição |
    | ----- | ----- |
    | 0 | Completa sem viagem |
    | 1 | Completa com viagem |

    [Teste: Checar se existe algum número diferente de 0 ou 1.
     Se encontrar, retornar erro indicando em qual linha.]
:param passo: Número do passo atual para registro/log
:param df:
:return:
"""
log_tela.info("### PASSO " + str(passo) + " - CD_ENTRE")
log_tela.warning('\n'#####\\n' +
                 'Este passo DEVE ser executado apenas após a \\n' +
                 'execução do passo que geram o TOT_VIAG.\\n' +
                 '#####\\n' )

# Definindo 'CD_ENTRE' baseado no valor de 'TOT_VIAG'
df.loc[df['TOT_VIAG'] == 0, 'CD_ENTRE'] = 0
```

```
df.loc[df['TOT_VIAG'] != 0, 'CD_ENTRE'] = 1

verifica_dummy(df, 'CD_ENTRE')
log_output.info('\n\n=====\\n')

return df

In [ ]: def passo_correcoes(passo, df):
    log_tela.info("### PASSO " + str(passo) + " - Correções")
    log_tela.info("Nenhuma correção a ser efetuada.")

    return df
```

3 Definindo a função principal (main)

Esta função vai ler os arquivos (csv) necessários e, então, irá efetuar cada passo, chamando as respectivas funções que foram definidas acima. Ao final, esta função irá salvar o resultado das transformações realizadas num arquivo csv.

```
In [ ]: def main():
    start_time = time.time()

    log_tela.info("Horário de início da execução: %s" %
                  time.strftime("%H:%M", time.localtime(start_time)))

    # ----
    log_tela.info('Lendo arquivos CSV')

    log_output.info('Lendo CSV da base original da OD.')
    od = pd.read_csv('bases/OD_2007.csv', sep=';', decimal=',')

    deflator = 1
    log_output.info('Deflator a ser utilizado para correção monetária: ' +
                    str(deflator))

    # Filtrando dataframe para pegar apenas uma amostra
    #od = od[:1000].copy()

    log_output.info('\n\n=====\\n')
    log_tela.info('Pré-Processamento.')
    od = pre_processamento(od)

    log_tela.info('Imprimindo a lista de variáveis do dataframe da OD.')
    log_tela.debug(od.columns.tolist())

    #Contador de 'PASSO'
    passo = 1

    # ----
    # ##Passo: "ANO"
    od = passo_ano(passo, od)
    passo += 1

    # ----
    # ##Passo: "DIA_SEM"
    od = passo_dia_sem(passo, od)
    passo += 1

    # ----
    # ##Passo: "UCODs"
    od = passo_ucods(passo, od)
    passo += 1

    # ----
    # ##Passo: "ZONA_DOM"
    od = passo_zona_dom(passo, od)
    passo += 1
```

```

# -----
# ##Passo: "SUBZONA_DOM"
od = passo_subzona_dom(passo, od)
passo += 1

# -----
# ##Passo: "MUN_DOM"
od = passo_mun_dom(passo, od)
passo += 1

# -----
# ##Passo: "F_DOM"
od = passo_f_dom(passo, od)
passo += 1

# -----
# ##Passo: "FE_DOM"
od = passo_fe_dom(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_DOM"
od = passo_tipo_dom(passo, od)
passo += 1

# -----
# ##Passo: "TOT_FAM"
od = passo_tot_fam(passo, od)
passo += 1

# -----
# ##Passo: "F_FAM"
od = passo_f_fam(passo, od)
passo += 1

# -----
# ##Passo: "FE_FAM"
od = passo_fe_fam(passo, od)
passo += 1

# -----
# ##Passo: "COND_MORA"
od = passo_cond_mora(passo, od)
passo += 1

# -----
# ##Passo: "QT_AUTO"
od = passo_qt_auto(passo, od)
passo += 1

# -----
# ##Passo: "QT_BICI"
od = passo_qt_bici(passo, od)
passo += 1

```

```

# -----
# ##Passo: "QT_MOTO"
od = passo_qt_moto(passo, od)
passo += 1

# -----
# ##PASSO: "REN_FAM"
od = passo_ren_fam(passo, od, deflator)
passo += 1

# -----
# ##Passo: "CD_RENFAM"
od = passo_cd_renfam(passo, od)
passo += 1

# -----
# ##Passo: "F_PESS"
od = passo_f_pess(passo, od)
passo += 1

# -----
# ##Passo: "FE_PESS"
od = passo_fe_pess(passo, od)
passo += 1

# -----
# ##Passo: "SIT_FAM"
od = passo_sit_fam(passo, od)
passo += 1

# -----
# ##Passo: "IDADE"
od = passo_idade(passo, od)
passo += 1

# -----
# ##Passo: "SEXO"
od = passo_sexo(passo, od)
passo += 1

# -----
# ##Passo: "GRAU_INSTR"
od = passo_grau_instr(passo, od)
passo += 1

# -----
# ##Passo: "OCUP"
od = passo_ocup(passo, od)
passo += 1

# -----
# ##Passo: "SETOR_ATIV"
od = passo_setor_ativ(passo, od)

```

```

    passo += 1

    # ##Passo: "REN_IND"
    od = passo_ren_ind(passo, od, deflator)
    passo += 1

    # ##Passo: "CD_RENIND"
    od = passo_cd_renind(passo, od)
    passo += 1

    # -----
    # ##Passo: "F_VIAG"
    od = passo_f_viag(passo, od)
    passo += 1

    # -----
    # ##"FE_VIAG"
    od = passo_fe_viag(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_ESC"
    od = passo_zona_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_ESC"
    od = passo_subzona_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_ESC"
    od = passo_mun_esc(passo, od)
    passo += 1

    # -----
    # ##Passo: "ESTUDA"
    # Este passo deve vir após os passos de localização da escola
    od = passo_estuda(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_TRAB1"
    od = passo_zona_trab1(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_TRAB1"
    od = passo_subzona_trab1(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_TRAB1"
    od = passo_mun_trab1(passo, od)

```

```

    passo += 1

    # -----
    # ##Passo: "ZONA_TRAB2"
    od = passo_zona_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_TRAB2"
    od = passo_subzona_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_TRAB2"
    od = passo_mun_trab2(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_ORIG"
    od = passo_zona_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_ORIG"
    od = passo_subzona_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_ORIG"
    od = passo_mun_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "ZONA_DEST"
    od = passo_zona_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "SUBZONA_DEST"
    od = passo_subzona_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "MUN_DEST"
    od = passo_mun_dest(passo, od)
    passo += 1

    # -----
    # ##Passo: "SERV_PAS_ORIG"
    od = passo_serv_pas_orig(passo, od)
    passo += 1

    # -----
    # ##Passo: "SERV_PAS_DEST"

```

```

od = passo_serv_pas_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_ORIG"
od = passo_motivo_orig(passo, od)
passo += 1

# -----
# ##Passo: "MOTIVO_DEST"
od = passo_motivo_dest(passo, od)
passo += 1

# -----
# ##Passo: "MOD01"
od = passo_modo1(passo, od)
passo += 1

# -----
# ##Passo: "MOD02"
od = passo_modo2(passo, od)
passo += 1

# -----
# ##Passo: "MOD03"
od = passo_modo3(passo, od)
passo += 1

# -----
# ##Passo: "MOD04"
od = passo_modo4(passo, od)
passo += 1

# -----
# ##Passo: "MODO_PRIN"
od = passo_modo_prin(passo, od)
passo += 1

# -----
# ##Passo: "TIPO_VIAG"
od = passo_tipo_vitag(passo, od)
passo += 1

# -----
# ##"H_SAIDA"; "MIN_SAIDA"; "ANDA_ORIG"; "H_CHEG"; "MIN_CHEG";
# "ANDA_DEST"; "DIST_VIAG" e "DURACAO"
# Nada há que se fazer em relação aos dados das colunas acima mencionadas

# -----
# ##Passo: "TIPO_EST_AUTO"
od = passo_tipo_est_auto(passo, od)
passo += 1

# -----

```

```

# ##Passo: "VALOR_EST_AUTO"
od = passo_valor_est_auto(passo, od, deflator)
passo += 1

# -----
# ##Passo: Coordenadas
od = coordenadas(passo, od)
passo += 1

# -----
# ##Passo: Gerando NO's e ID's
od = gera_nos_e_ids(passo, od)
passo += 1

# -----
# ##Passo: "TOT_VIAG"
od = passo_tot_viag(passo, od)
passo += 1

# -----
# ##Passo: "CD_ENTRE"
od = passo_cd_entre(passo, od)
passo += 1

# -----
# ##Passo: Correções
od = passo_correcoes(passo, od)
passo += 1

log_tela.info('Salvando dataframe como arquivo CSV')
# -----
# ## Salvando o dataframe num arquivo local
od.to_csv('outputs/2007_od.csv', sep=';', decimal=',', index=False)

log_tela.info("Base gerada. Arquivo: outputs/2007_od.csv")

log_tela.info("Tempo total de execução: %s segundos" %
             (time.time() - start_time))

log_tela.info("Horário de finalização: %s" %
             (time.strftime("%H:%M", time.localtime(time.time()))))

log_tela.info("Terminou o main")

```

4 RUN the main() function...

```
In [ ]: if __name__ == "__main__":
    main()
```

ANEXO E – Síntese de resultados para 4 conglomerados

RESUMO DE RESULTADOS PARA AGRUPAMENTO POR ATRIBUTOS DE VIAGENS DA FAMÍLIA - MÉTODO WARD											
Cluster nº (ward)	% de pessoas de 1977	% de pessoas de 1987	% de pessoas de 1997	% de pessoas de 2007	Nº de registos	Nº de viagens	Nº médio de viagens por pessoa	Nº médio de viagens por família	Média da distância total da família (m)	Distância média por viagem da pessoa (m)	Média da duração total da viagem da pessoa (min)
1	100	0	0	0	229046	187441	2,82	7,70	13049,87	4627,61	87,68
2	0	100	0	0	223926	186334	2,82	7,60	12198,88	4533,88	85,52
3	0	0	100	0	196989	169665	2,82	7,61	12044,87	4621,11	82,20
4	0	0	0	100	196989	169665	2,64	6,25	13282,73	5010,88	94,56
Diferenças % entre Min e Max											
Diferenças % entre Min e Max											
7%											
19%											
CARACTERÍSTICAS DE VIAGENS											
Cluster nº (ward)	Nº de pessoas	% de pessoas feminino	% de pessoas do sexo masculino	Média de idade (anos)	% de pessoa responsável	% de conjuges	% filhos / enteados	% de outros parentes / agregados	% de empregados	% de outros	% de pessoas não alfabetizadas ou com fundamental incompleto
1	108028	51,71	48,29	27,17	24,13	18,9	44,36	10,11	2,25	0,24	74,81
2	110813	52,01	47,99	27,84	25,45	19,67	44,17	9,60	0,92	0,19	72,92
3	95789	52,09	47,91	30,1	27,18	19,47	41,86	10,66	0,95	0,08	60,03
4	91195	51,73	46,27	29,78	33,76	20,26	34,49	10,00	1,17	0,07	37,91
Diferenças % entre Min e Max											
4%											
27%											
29%											
9%											
22%											
10%											
95%											
71%											
CARACTERÍSTICAS DE PESSOAS											
Cluster nº (ward)	Nº de famílias	% de famílias com presença de crianças entre 0 e 4 anos	% de famílias com presença de crianças entre 5 e 9 anos	% de famílias com presença de crianças entre 10 e 14 anos	% de famílias com presença de idosos (60+)	% de famílias com presença de trabalhadores (as) na família	% de famílias que têm 1 automóvel	% de famílias que têm 2 ou mais automóveis	% de famílias que têm 3 ou mais automóveis	% de famílias que têm 4 ou mais automóveis	% de pessoas com fundamental incompleto ou médio incompleto
1	108028	4,13	29,99	27,65	28,11	89,96	1,46	3,46	13,53	0,65	11,16
2	228217	3,93	32,44	32,42	26,79	87,22	1,45	3,23	12,41	0,67	9,78
3	230555	5,68	26,75	26,09	26,64	85,58	1,54	3,44	12,45	0,72	11,47
4	308555	2,98	12,40	14,41	15,97	86,57	1,42	38,25	17,55	0,78	10,58
Diferenças % entre Min e Max											
28%											
60%											
56%											
44%											
11%											
39%											
8%											
11%											
37%											
39%											
71%											
CARACTERÍSTICAS DE FAMÍLIAS											
Cluster nº (ward)	Nº de famílias	Tamanho médio da família	% de famílias com presença de crianças entre 0 e 4 anos	% de famílias com presença de crianças entre 5 e 9 anos	% de famílias com presença de crianças entre 10 e 14 anos	% de famílias com presença de idosos (60+)	% de famílias com presença de trabalhadores (as) na família	% de famílias que têm 1 automóvel	% de famílias que têm 2 ou mais automóveis	% de famílias que têm 3 ou mais automóveis	% de famílias que têm 4 ou mais automóveis
1	229046	4,13	29,99	27,65	28,11	89,96	1,46	3,46	13,53	0,65	11,16
2	123485	102679	0	100	0	0	0	0	0	0	0
3	100443	80439	0	100	0	0	0	0	0	0	0
4	335358	333199	0	0	51,94	48,06	1,42	3,52	14,95	0,67	12,57
Diferenças % entre Min e Max											
10%											
15%											
CARACTERÍSTICAS DE VIAGENS											
Cluster nº (centr)	Nº de registos	Nº de viagens	% de pessoas de 1977	% de pessoas de 1987	% de pessoas de 1997	% de pessoas de 2007	Nº médio de viagens por pessoa	Nº médio de viagens por família	Média da distância total da família (m)	Média da distância media por viagem da pessoa (m)	Média da duração total da viagem da pessoa (min)
1	229046	187441	100	0	0	0	2,82	7,70	13049,87	4627,61	87,68
2	123485	102679	0	100	0	0	2,68	7,10	12107,87	4516,87	85,24
3	100443	80439	0	100	0	0	2,54	6,87	12175,88	4593,88	85,86
4	335358	333199	0	0	51,94	48,06	2,52	6,51	12244,24	4525,87	93,60
Diferenças % entre Min e Max											
10%											
15%											
CARACTERÍSTICAS DE PESSOAS											
Cluster nº (centr)	Nº de pessoas	% de pessoas do sexo masculino	Média de idade (anos)	% de pessoa responsável	% de conjuges	% filhos / enteados	% de outros parentes / agregados	% de empregados	% de outros	% de pessoas não alfabetizadas ou com fundamental incompleto	% de pessoas com fundamental incompleto ou médio incompleto
1	108028	51,71	47,99	27,17	24,13	18,90	44,36	10,11	2,25	0,24	74,81
2	59100	53,00	47,60	29,60	26,52	19,73	41,75	10,52	1,26	0,22	67,73
3	51713	50,89	49,11	25,82	24,23	19,60	46,92	8,54	0,54	0,16	12,57
4	190185	52,88	47,12	33,46	30,34	19,98	38,21	10,34	1,05	0,07	49,40
Diferenças % entre Min e Max											
4%											
23%											
19%											
7%											
CARACTERÍSTICAS DE FAMÍLIAS											
Cluster nº (centr)	Nº de famílias	Tamanho médio da família	% de famílias com presença de crianças entre 0 e 4 anos	% de famílias com presença de crianças entre 5 e 9 anos	% de famílias com presença de crianças entre 10 e 14 anos	% de famílias com presença de idosos (60+)	% de famílias com presença de trabalhadores (as) na família	% de famílias que têm 1 automóvel	% de famílias que têm 2 ou mais automóveis	% de famílias que têm 3 ou mais automóveis	% de famílias que têm 4 ou mais automóveis
1	229046	4,13	29,99	27,65	28,11	89,96	1,46	3,46	13,53	0,65	11,16
2	123485	3,77	29,63	26,35	30,56	85,27	1,45	3,27	12,46	0,63	11,00
3	12532	4,13	35,55	37,16	18,62	88,58	1,45	31,26	8,15	0,50	30,75
4	57709	3,30	19,33	20,94	31,66	82,78	1,48	36,67	16,55	0,75	22,86
Diferenças % entre Min e Max											
7%											
19%											
23%											
CARACTERÍSTICAS DE VIAGENS											
Cluster nº (centr)	Nº de registos	Nº de viagens	% de pessoas que servem no destino	Média das Pessoas que servem no destino	Média das Distâncias de Viagens	Média das Durações de Viagens	Média das Distâncias de Viagens da Família	Média das Durações de Viagens da Família	Média das Distâncias de Viagens da Família (m)	Média das Durações de Viagens da Família (min)	Média das Durações de Viagens da Família (segundos)
1	229046	187441	31,11	31,11	87,68	87,68	13049,87	13049,87	13049,87	87,68	87,68
2	123485	102679	32,64	32,64	85,52	85,52	12256,88	12256,88	12256,88	85,52	85,52
3	100443	80439	33,00	33,00	85,86	85,86	12349,87	12349,87	12349,87	85,86	85,86
4	335358	333199	35,88	35,88	93,16	93,16	124,28	124,28	124,28	93,16	93,16
Diferenças % entre Min e Max											
6%											
13%											
CARACTERÍSTICAS DE PESSOAS											
Cluster nº (centr)	Nº de pessoas	% de pessoas feminino	% de pessoas do sexo masculino	Média de idade (anos)	% de pessoa responsável	% de conjuges	% filhos / enteados	% de outros parentes / agregados	% de empregados	% de outros	% de pessoas não alfabetizadas ou com fundamental incompleto
1	108028	51,71	48,29	27,17	24,13	18,90	44,36	10,11	2,25	0,24	74,81
2	59100	53,00	47,60	29,60	26,52	19,73	41,75	10,52	1,26	0,22	67,73
3	51713	50,89	49,11	25,82	24,23	19,60	46,92	8,54	0,54	0,16	12,57
4	190185	52,88	47,12	33,46	30,34	19,98	38,21	10,34	1,05	0,07	49,40
Diferenças % entre Min e Max											
4%											
23%											
19%											