

Sérgio da Silva Rodrigues

Análise da precificação de imóveis na
cidade do Rio de Janeiro utilizando
Modelagem Hedônica e os efeitos da
autocorrelação espacial

Rio de Janeiro
2015

Rodrigues, Sérgio da Silva

Análise da precificação de imóveis na cidade do Rio de Janeiro utilizando modelagem hedônica e os efeitos da autocorrelação espacial / Sérgio da Silva Rodrigues. – 2015.

220 f.

Dissertação (mestrado) – Fundação Getulio Vargas, Escola de Matemática Aplicada.

Orientador: Renato Rocha Souza.

Inclui bibliografia.

1. Análise de regressão. 2. Modelos matemáticos. 3. Bens imóveis – Avaliação.

I. Souza, Renato Rocha. II. Fundação Getulio Vargas. Escola de Matemática Aplicada. III Título.

CDD – 519.536

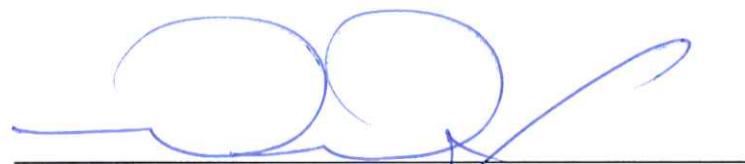
SERGIO DA SILVA RODRIGUES

**ANÁLISE DA PRECIFICAÇÃO DE IMÓCEIS NA CIDADE DO RIO DE JANEIRO
UTILIZANDO MODELAGEM HEDÔNICA E OS EFEITOS DA AUTO CORRELAÇÃO
ESPECIAL.**

Dissertação apresentada ao Curso de Mestrado em Modelagem Matemática da Informação da Escola de Matemática Aplicada da Fundação Getulio Vargas para obtenção do grau de Mestre em Modelagem Matemática da Informação.

Data da defesa: 17/07/2015

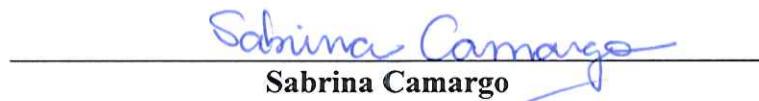
ASSINATURA DOS MEMBROS DA BANCA EXAMINADORA



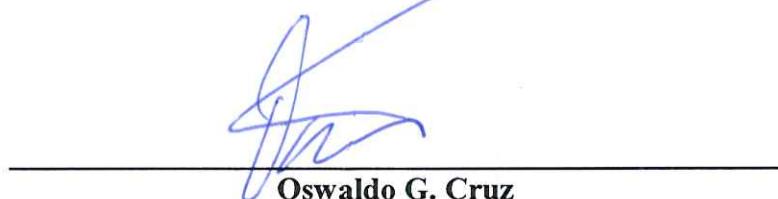
Renato Rocha Souza
Orientador (a)



Flávio Codeço Coelho



Sabrina Camargo
Sabrina Camargo



Oswaldo G. Cruz

Sérgio da Silva Rodrigues

Análise da precificação de imóveis na cidade do Rio de Janeiro utilizando Modelagem Hedônica e os efeitos da autocorrelação espacial

Dissertação apresentada a Fundação Getúlio Vargas, para a obtenção de Título de Mestre em Modelagem Matemática da Informação, na Escola de Matemática Aplicada da Fundação Getúlio Vargas.

Orientador: Renato Rocha Souza

Rio de Janeiro, RJ
2015

Agradecimentos

Meu primeiro agradecimento é a meus pais, Amaro e Alzira, pessoas simples, que investiram tudo o que tinham, tanto em amor e carinho, quanto em educação e instrução, realizando em mim os seus sonhos de conquista e reconhecimento.

Agradeço à minha esposa Luíza Helena e meu único filho Sérgio Gabriel, pela compreensão de minhas ausências ao longo dessa jornada, contínuo apoio e motivação ao alcance desse objetivo.

A Marcus Vinícius, grande patrocinador dessa conquista, que acreditou em minha capacidade, defendeu o investimento realizado pela empresa onde trabalho, perdoou as horas excedentes ao planejado dedicadas ao estudo, e que sempre incentivou-me a continuar a despeito das dificuldades meu muito obrigado por ajudar-me a realizar esse sonho.

Um agradecimento especial a Jorge Antônio Ferreira dos Santos, meu professor de matemática da quinta a sétima série, que abdicou de trabalhar em colégios particulares à época para dedicar parte de seu tempo a alunos em uma escola municipal no subúrbio da cidade, e com extremo amor à essa mais digna das profissões, maravilhou seus alunos com a magia que só um Mestre é capaz de criar.

Ao meu orientador Renato Rocha Souza, agradeço a confiança dedicada e as palavras de apoio nos momentos de dificuldade.

Agradeço a todos os professores e equipe administrativa da EMAP, em que cada um contribui com seu trabalho à expansão do meu conhecimento nessa maravilhosa área de estudo.

Finalmente, aos demais amigos que estiveram sempre ao meu lado com palavras de incentivo.

Resumo

A escolha da cidade do Rio de Janeiro como sede de grandes eventos esportivos mundiais, a Copa do Mundo de Futebol de 2014 e os Jogos Olímpicos de 2016, colocou-a no centro de investimentos em infraestrutura, mobilidade urbana e segurança pública, com consequente impacto no mercado imobiliário, tanto de novos lançamentos de empreendimentos, quanto na revenda de imóveis usados.

Acredita-se que o preço de um imóvel dependa de uma relação entre suas características estruturais como quantidade de quartos, suítes, vagas de garagem, presença de varanda, tal como sua localização, proximidade com centros de trabalho, entretenimento e áreas valorizadas ou degradadas. Uma das técnicas para avaliar a contribuição dessas características para a formação do preço do imóvel, conhecido na Econométrica como Modelagem Hedônica de Preços, é uma aplicação de regressão linear multivariada onde a variável dependente é o preço e as variáveis independentes, as respectivas características que deseja-se modelar. A utilização da regressão linear implica em observar premissas que devem ser atendidas para a confiabilidade dos resultados a serem analisados, tais como independência e homoscedasticidade dos resíduos e não colinearidade entre as variáveis independentes.

O presente trabalho objetiva aplicar a modelagem hedônica de preços para imóveis localizados na cidade do Rio de Janeiro em um modelo de regressão linear multivariada, em conjunto com outras fontes de dados para a construção de variáveis de acessibilidade e socioambiental a fim de verificar a relação de importância entre elas para a formação do preço e, em particular, exploramos brevemente a tendência de preços em função da distância a favelas. Em atenção aos pré-requisitos observados para a aplicação de regressão linear, verificamos que a premissa de independência dos preços não pode ser atestada devido a constatação da autocorrelação espacial entre os imóveis, onde não

apenas as características estruturais e de acessibilidade são levadas em consideração para a precificação do bem, mas principalmente a influência mútua que os imóveis vizinhos exercem um ao outro.

Palavras-chave: regressão linear, modelagem hedônica, imóveis

Abstract

The choice of the city of Rio de Janeiro as the host of major world sporting events, the Football World Cup 2014 and the Olympic Games in 2016, put it in the center of investments in infrastructure, urban mobility and public safety, with consequent impact on the real estate market, new enterprise releases, as in resale of used real estate units.

It is believed that the price of a real estate unit depends on a relationship between their structural characteristics like the number of rooms, suites, parking spaces, balcony presence, such as its location, proximity to work centers, entertainment and reclaimed or degraded areas. One of the techniques to assess the contribution of these features to the formation of the property price, known as the Econometric Modeling Hedonic Price, is an application of multivariate linear regression where the dependent variable is the price and the independent variables, the respective characteristics desired to model. The use of linear regression implies observe the assumptions that must be met for the reliability of the results to be analyzed, such as independence and homoscedasticity of the residuals and no collinearity among the independent variables.

This paper aims to apply the hedonic modelling prices for properties located in the city of Rio de Janeiro in a multivariate linear regression model, together with other data sources for building accessibility and social environmental variables in order to verify the relationship of importance among them for the prices and, in particular, briefly explore the role-price trend when moving away from the slums. Noting the prerequisites observed for the application of linear regression, we have found that the assumption of independence of prices cannot be proved due to the presence of spatial autocorrelation between the real estate units, where not only the structural and accessibility features are taken into account for the pricing process, but mostly the mutual influence neighbouring properties carry each other.

Keywords: hedonic model, linear regression, real estate

Listas de Figuras

2.1	Mapa da cidade do Rio de Janeiro em 1808.	6
2.2	Mapa de uso do solo cidade do Rio de Janeiro.	10
2.3	Intuição do método Mínimos Quadrados.	21
2.4	Exemplo de sobreajuste.	23
2.5	Intuição do sobreajuste (overfitting)	25
3.1	Exemplo da lista de anúncios para a cidade do Rio de Janeiro.	31
3.3	Trecho de HTML contendo preço de um imóvel.	36
3.4	Tela Inicial do Mapa Digital do Rio de Janeiro.	43
3.5	Localização da Estação de Metrô Carioca, definido como o centro da cidade.	49
3.6	Preparação das planilhas do Armazém de Dados para importação.	50
3.7	Percentual de valores ausentes.	54
3.8	Exemplo de visualização utilizado para detecção de outliers.	55
3.9	Distribuição de antes e depois da remoção dos imóveis com coordenadas repetidas.	56
3.10	Comparativo entre os 3 modelos para seleção.	63
4.2	Desvio padrão da variável <i>preco</i> ao longo da Latitude e Longitude da cidade.	70

4.3 Gráfico de Moran para Knn=100	75
4.4 Variável <i>dist_favela</i> em bairros	86
A.1 Mapa das divisões administrativas da cidade do Rio de Janeiro	98
A.2 Nomes dos bairros	99
A.3 Localização dos imóveis.	100
A.4 Localização das Delegacias de Polícia Civil.	101
A.5 Localização das estações do metrô.	102
A.6 Localização dos aglomerados subnormais, favelas.	103
A.7 Localização das lagoas.	104
A.8 Localização das praias.	105
A.9 Localização das estações de trem.	106
A.10 Geometria dos principais logradouros da Cidade do Rio de Janeiro	107
A.11 Localização das unidades de saúde privada.	108
A.12 Localização das unidades de saúde pública.	109
A.13 Localização das unidades do Corpo de Bombeiros.	110
G.1 Boxplot e histograma da variável <i>dist_centro</i>	169
G.2 Boxplot e histograma da variável <i>dist_delegacia</i>	169
G.3 Boxplot e histograma da variável <i>dist_metro</i>	170
G.4 Boxplot e histograma da variável <i>dist_favela</i>	170
G.5 Boxplot e histograma da variável <i>dist_lagoa</i>	171
G.6 Boxplot e histograma da variável <i>dist_praia</i>	171
G.7 Boxplot e histograma da variável <i>dist_logradouro</i>	172
G.8 Boxplot e histograma da variável <i>dist_saude_privada</i>	172
G.9 Boxplot e histograma da variável <i>dist_saude_publica</i>	173
G.10 Boxplot e histograma da variável <i>dist_bombeiro</i>	173
H.1 Boxplot e histograma da variável <i>se_anos_estudo</i>	174

H.2 Boxplot e histograma da variável <i>se_anos_estudo</i>	175
H.3 Boxplot e histograma da variável <i>se_perc_alfabetizacao</i>	175
H.4 Boxplot e histograma da variável <i>se_renda</i>	176
H.5 Boxplot e histograma da variável <i>se_saneamento</i>	176
H.6 Boxplot e histograma da variável <i>se_anos_estudo</i>	177
H.7 Boxplot e histograma da variável <i>se_idh</i>	177
H.8 Boxplot e histograma da variável <i>se_crm_roubo</i>	178
H.9 Boxplot e histograma da variável <i>se_crm_violento</i>	178

Listas de Tabelas

3.1	Os 20 n-gramas de maior ocorrência nas descrições dos imóveis.	38
3.2	Lista de bairros por AISP.	52
3.3	Descrição das variáveis.	58
4.1	Análise descritiva das variáveis propostas a modelagem.	66
4.1	Análise descritiva das variáveis propostas a modelagem.	67
4.2	Dez favelas com maior número de imóveis próximos.	68
4.3	Resultado do modelo com todas as variáveis, ordenado por nome. Valores em R\$ 1.000,00.	71
4.3	Resultado do modelo com todas as variáveis, ordenado por nome. Valores em R\$ 1.000,00.	72
4.4	Resultado do modelo sem variáveis correlacionadas, ordenado por nome. Valores em R\$ 1.000,00.	73
4.4	Resultado do modelo sem variáveis correlacionadas, ordenado por nome. Valores em R\$ 1.000,00.	74
4.5	Resultado do modelo <i>lag</i> espacial, ordenado por nome. Valores em R\$ 1.000,00.	76
4.6	Variáveis por quantidade de ocorrência em bairros	81
4.7	Bairros por quantidade de variáveis	82
4.8	Máximos e mínimos das variáveis com respectivos bairros.	83

4.8 Máximos e mínimos das variáveis com respectivos bairros.	84
4.8 Máximos e mínimos das variáveis com respectivos bairros.	85

List of Listings

3.1	Método em Python da 1.a etapa de captura de imóveis.	33
3.2	Exemplo de <i>Web Scrapping</i> para captura da informação preço.	37
3.3	10 ocorrências do token "sol"em seu contexto original.	39
3.4	Lista de variáveis estruturais dicotômicas.	39
3.5	Importar <i>shapefiles</i> para o banco de dados PostGIS.	44
3.6	Exemplo de cálculo de menor distância por SQL.	45

Sumário

1	Introdução	1
1.0.1	Objetivos	2
1.1	Justificativa	2
1.2	Organização do estudo	3
2	Revisão da Literatura	5
2.1	A cidade do Rio de Janeiro	5
2.2	Modelos Hedônicos	12
2.3	Modelos hedônicos realizados no Brasil	15
2.4	Regressão Linear	18
3	Metodologia	29
3.1	Obtenção dos dados	29
3.1.1	Variáveis estruturais básicas	29
3.1.2	Demais variáveis estruturais	37
3.1.3	Variáveis de acessibilidade	41
3.1.4	Variáveis socioambientais	48
3.2	Tratamento dos dados	53
3.2.1	Variáveis estruturais	53
3.2.2	Variáveis de acessibilidade	55

3.2.3	Variáveis socioambientais	57
3.3	Descrição das variáveis	57
3.3.1	Variáveis <i>quartos</i> , <i>suítes</i> e <i>garagem</i> tratadas como dicotômicas	59
3.4	Seleção do Modelo	60
3.4.1	Seleção do modelo hedônico para toda a cidade do Rio de Janeiro	61
3.4.2	Seleção do modelo hedônico para cada bairro da cidade do Rio de Janeiro	63
4	Apresentação e análise dos resultados	65
4.1	Análise descritiva das variáveis	65
4.2	Modelo hedônico para a cidade do Rio de Janeiro	70
4.3	Modelo hedônico para os bairros	79
5	Conclusão	89
5.1	Limitações e possíveis extensões	91
5.2	Últimas palavras	94
Referências Bibliográficas		95
A	Mapas da cidade do Rio de Janeiro	97
B	Listagem do módulo captura.py	111
C	Listagem do módulo zap_util.py	116
D	Listagem do IPython Notebook Variaveis_estruturais.ipynb	126
E	Listagem do IPython Notebook Variaveis_socioambientais.ipynb	143
F	Listagem do IPython Notebook Tratar_variaveis_estruturais.ipynb	152
G	Verificação das variáveis de acessibilidade	168

H Verificação das variáveis socioambientais	174
I Listagem do IPython Notebook Reg_Linear_-_Modelo_Cidade.ipynb	179
J Listagem do IPython Notebook Reg_Linear_-_Modelos_Bairro.ipynb	209
K Resultados do Modelo Hedônico para os 5 bairros em maior quantidade de observações	215
K.0.1 Copacabana	216
K.0.2 Barra da Tijuca	217
K.0.3 Tijuca	218
K.0.4 Recreio dos Bandeirantes	219
K.0.5 Botafogo	220

Capítulo 1

Introdução

Com a conquista da cidade do Rio de Janeiro em sediar os Jogos Olímpicos de Verão de 2016 e participar como uma das sedes em outros eventos esportivos de grande porte como a Copa das Confederações de 2013 e a Copa do Mundo de 2014, a cidade do Rio de Janeiro tornou-se o foco de grandes investimentos de infraestrutura para mobilidade urbana e segurança pública, com impacto direto no mercado imobiliário, elevando o custo dos imóveis residenciais a um dos mais caros do país, despertando o interesse em avaliar a distribuição desses preços pela cidade a fim de investigar quais elementos são responsáveis pela sua determinação.

Uma das técnicas a respeito, Modelagem Hedônica, considera um bem de consumo, em nosso caso imóveis residenciais, como algo constituído de diversas características de interesse dos consumidores onde cada uma contribui individualmente para o valor final. Para se obter a contribuição de cada característica, o preço do imóvel é atribuído como variável dependente e as características como variáveis independentes em um modelo de Regressão Linear e os coeficientes resultantes determinam não apenas a contribuição de cada características para o preço final mas também permite observar a relevância relativa das características entre si.

Embora existam alguns estudos similares no Brasil, a cidade do Rio de Janeiro é

citada em apenas um, Neto (2002), que utilizou uma fonte de dados com 120 observações de preços de imóveis, construindo 19 variáveis independentes para estimativa da variável preço, obtidos a partir de anúncios de lançamentos de empreendimentos.

1.0.1 Objetivos

Face ao exposto, o presente estudo aborda uma pesquisa exploratória a respeito da distribuição dos preços de imóveis residenciais usados na cidade do Rio de Janeiro, cujo principal objetivo é a construção de um modelo de estimativa da variável preço, a fim de observar o relacionamento entre características dos imóveis para a sua formação e determinar se há autocorrelação espacial, o que determinará se a precificação de imóveis ocorre de forma independente das observações próximas.

Como objetivos auxiliares, apresentamos uma análise descritiva das características modeladas, passo necessário para a contextualização dos respectivos coeficientes resultantes da regressão linear a ser aplicada. Um segundo objetivo auxiliar é a comparação da quantidade de características dos imóveis modeladas para cada bairro, e a quantidade de bairros em que cada característica está presente, dados derivados a partir da construção de modelos para cada bairro. Por fim, consideramos uma análise do relacionamento entre preço de imóveis e distância aos aglomerados subnormais, conhecidas como *favelas*.

1.1 Justificativa

As justificativas para a discussão desse tema recaem sobre uma necessidade de revisão do modelo proposto originalmente por Neto (2002) para a década atual em virtude do interesse dos agentes governamentais a respeito do mercado imobiliário como termômetro de fenômenos sociais como crime, trânsito, oportunidades de emprego e constituição

demográfica, avaliação de investimentos em benfeitorias públicas e programas sociais, além dos interesses tributários como Imposto sobre a Propriedade Predial e Territorial Urbana, IPTU¹, Imposto sobre Transmissão de *Causa Mortis* e Doação de Bens ou Direitos, ITCMD² e Imposto de Transmissão de Bens Imóveis, ITBI³. A base de cálculo para os três impostos citados é o valor *venal* do imóvel, uma estimativa promovida pelo agente público pois o valor efetivamente pago pelo comprador ao vendedor é desconhecido pelo tributador e protegido por lei, a desejo do proprietário. Uma correta aferição desse valor permite ao poder público evitar tanto fraudes, quando o proprietário declara um valor menor que o pago, quanto abusos, quando o poder público determina um valor acima do pago, preservando-se o princípio da eficiência fiscal da autoridade pública. Com relação ao setor privado, esse aborda Modelos Hedônicos em precificação de imóveis para o estudo de viabilidade de empreendimentos, determinação dos itens mais valorizados pelos consumidores e adequação de sua participação na intermediação de vendas de imóveis usados. O principal diferencial desse estudo aos demais realizados no Brasil é a obtenção dos dados de forma eletrônica, a partir de fontes de dados disponíveis na Internet a fim de obter um grande número de observações e variáveis, e uso integral de ferramentas *Open Source* que permita fácil acesso para replicação das técnicas utilizadas.

1.2 Organização do estudo

Estruturamos o estudo em 5 capítulos, organizados da seguinte forma:

O trabalho começa com o presente capítulo 1, [Introdução](#), p. 1, identificando o tema a ser abordado, uma breve descrição de trabalhos anteriores, os objetivos principal e auxiliares a serem alcançados, a metodologia a ser aplicada, as justificativas para

¹IPTU: Constituição Federal , Título VI, Capítulo I, Seção V, artigo 156, inciso I.

²ITCMD: Constituição Federal , Título VI, Capítulo I, Seção IV, artigo 155, inciso II.

³ITBI: Constituição Federal , Título VI, Capítulo I, Seção V, artigo 156, inciso II.

escolha do tema e a organização do texto.

O capítulo 2, [Revisão da Literatura](#), p. 5, aborda uma revisão da literatura pertinente ao tema escolhido, descrevendo o modelo matemático, a descrição dos principais conceitos utilizados e trabalhos anteriores.

O capítulo 3, [Metodologia](#), p. 29, descreve a metodologia utilizada, detalhando a obtenção, construção e tratamento dos dados, seleção do modelo de regressão e comentários sobre as tecnologias utilizadas .

O capítulo 4, [Apresentação e análise dos resultados](#), p. 65, relata os resultado obtidos com a aplicação do modelo e constrói algumas interpretações a partir da análise dos mesmos.

O capítulo 5, [Conclusão](#), p. 89, finaliza o trabalho discutindo os resultados gerados, apresenta oportunidades de extensão e propõe soluções que podem ser implementadas com tecnologia da informação.

A bibliografia á apresentada após os principais capítulos com uma lista das referências utilizadas pertinentes ao tema.

Os apêndices contém os códigos fontes utilizados para a captura dos dados, construção e tratamento das variáveis, seleção do modelo e execução da regressão para a cidade e para os bairros, complementando os detalhes abordados ao longo da discussão do estudo pelos capítulos.

Capítulo 2

Revisão da Literatura

Neste capítulo apresentamos a revisão bibliográfica das teorias utilizadas neste estudo.

2.1 A cidade do Rio de Janeiro

Pode-se afirmar que uma primeira demanda de moradias na cidade do Rio de Janeiro começou quando essa tomou de Salvador, em 1763, o título de capital da colônia em decorrência da exploração do ouro em Minas Gerais e servir como rota para o escoamento desse metal à Portugal¹. Pouco tempo depois um novo evento alterou para a sempre a rotina da cidade e demandou nova questão habitacional, com a chegada da corte real portuguesa, em 1808, fugindo das guerras napoleônicas que assolararam as monarquias europeias. Ao decidir-se a transferência da corte dois meses antes, preparativos foram iniciados para que a cidade fosse digna de receber os novos visitantes . Embora capital da colônia, as restrições impostas por Portugal ao Brasil impediam seu desenvolvimento econômico e consequentemente urbano. A maior parte da população era escrava e os poucos trabalhadores livres ocupavam vilarejos ao redor de Igrejas e Santas Casas de Misericórdia sob o contexto de defesa e ausência de meios de transporte

¹História da Cidade do Rio de Janeiro, Prefeitura do Rio: <http://www.rio.rj.gov.br/web/guest/exibeconteudo?article-id=87129>

(de Castro Pereira, 2009, p. 2). A chegada dos cerca de 10 a 15 mil portugueses obrigou os proprietários das melhores casas desocuparem-nas a fim de servir de residência aos novos imigrantes. Por sua vez, esses desalojados pressionaram a camada da população inferior, em um efeito cascata, iniciando uma ocupação sem o devido planejamento arquitetônico e urbano da cidade (Schultz, 2008, p. 7-11). Abaixo, ilustramos um mapa da cidade do Rio de Janeiro em 1808, solicitada por Dom João VI. Não abrange os bairros da Zona Sul e é limitado a São Cristóvão na Zona Norte.

Figura 2.1: Mapa da cidade do Rio de Janeiro em 1808. Fonte: J. A. dos Reis, Imprensa Régia 1812. ²



Ao longo do século XIX, o desenvolvimento habitacional expandiu-se significativamente nas formas de cortiços, vilas e ocupações das encostas dos morros e acima. Boa parte da população pobre e escravos recém libertados ocuparam o Morro do Castelo,

ex-combatentes da Guerra do Paraguai e Guerra dos Canudos levantaram cortiços nos morros da Providência e Santo Antônio. Discute-se que parte das favelas atuais são ex-quilombos, transformando-se de centro de resistência a local de moradia dos escravos libertos em 1888. Com a gentrificação do centro da cidade e outras áreas consideradas nobres como São Cristóvão, Botafogo, Flamengo e Glória, a população foi expandido-se onde os acidentes geográficos e limites de transporte permitiam, ([de Castro Pereira, 2009](#), p. 4-5). O resultado da ocupação à revelia de qualquer planejamento levou a cidade a uma péssima situação sanitária. Pestes e doenças assolavam a população, a qualidade do ar havia se degenerado, e a violência aumentava sem controle. A ascensão do café catapultou novo desenvolvimento à cidade e o governo viu-se obrigado a implantar uma nova infraestrutura para acomodar o crescimento industrial. Os carros puxados a burros são substituídos por bondes elétricos, e linhas ferroviárias permitiram o ocupação do subúrbio, embora acentuando a diferenciação de classes. Na transição do século XIX para o século XX, o governo aborda a questão habitacional e sanitária com incentivos ao empresariado nacional e subsídios para a construção das *vilas operárias*, habitações mais higiênicas do que os cortiços e vilas improvisadas para a classe trabalhadora ([de Castro Pereira, 2009](#), p. 5-7).

A primeira grande intervenção do poder público sobre o planejamento urbano da cidade ocorreu na prefeitura de Pereira Passos, no início do século XX, onde atribuindo-se a questão sanitária, eliminou os cortiços e vilarejos remanescentes nas áreas nobres e delimitou o subúrbio para a implantação das indústrias e moradia dos trabalhadores, dando-lhes acesso a meios de transporte rodoviários mais baratos, desestimulando a procura das vilas operárias. Os bairros periféricos ao Centro à época como Catumbi, Cidade Nova, Estácio, Lapa, Gamboa, Santo Cristo e Saúde, não receberam atenção durante a revolução urbana de Pereira Passos, tornando-se uma alternativa de moradia ao subúrbio. As administrações públicas seguintes contribuíram ainda mais para a estratificação e estigmatização social e espacial entre as classes na cidade ([de Castro Pereira,](#)

2009, p. 7-9).

Na era Vargas, a pressão dos trabalhadores por melhores condições encontrava ressonância nas idéias socialistas e comunistas. A fim de combater uma possível revolução socialista no Brasil, o governo assume certa orientação trabalhista, aproximando-se dessa classe concedendo-lhe alguns direitos já apropriados pela mesma classe na Europa, como salário-mínimo, previdência social, jornada de 8 horas, o que garantiu segurança financeira para as aquisições de moradia, e a construção de conjuntos habitacionais como parte de uma política paternalista com o objetivo de angariar votos. Entretanto, tal política estava limitada ao eixo urbano industrial, em detrimento das atividades rurais, o que incentivou a migração dos trabalhadores do campo para as cidades, resultando no crescimento desordenado do subúrbio e favelas (de Castro Pereira, 2009, p. 7-9).

Nos dias atuais, a cidade do Rio de Janeiro ocupa o ranking de segundo maior PIB³ do país, atrás somente de São Paulo. Além de servir de sede do Governo do Estado do Rio de Janeiro, também o é de grandes empresas públicas e privadas destacando-se Petrobras, Banco do Brasil, Banco Nacional do Desenvolvimento Social, Vale do Rio Doce e Organizações Globo, além de inúmeras empresas nos setores de *e-commerce*, petroquímica, farmacêutica, naval e telecomunicações. É o segundo maior polo científico do país, abrigando diversas universidades, centros de pesquisa e fundações de ensino. É referência como pólo cultural, tendo em seu território museus e teatros, além de servir como paisagem de produções cinematográficas. No setor turístico, é a cidade mais visitada do país, tendo como ápice o Carnaval, considerado a maior festa popular do mundo. Todos esses fatores, aliados à escolha da cidade como sede de grandes eventos esportivos, os Jogos Pan-americanos de 2007, Copa das Confederações de 2013, Copa do Mundo de Futebol de 2014 e Jogos Olímpicos de Verão de 2016, em que a partir de então a cidade recebeu investimentos em infraestrutura e mobilidade urbana, em

³IBGE (2011): ftp://ftp.ibge.gov.br/Pib_Municípios/2011/pdf/tabc01.pdf

conjunto com a implantação de Unidades de Polícia Pacificadora nas principais favelas e comunidades, contribuindo para a percepção de diminuição da violência contra seus habitantes, aliado ao incentivo ao crédito imobiliário promovido pelo Governo Federal na administração do presidente Luís Inácio Lula da Silva, resultou em um novo fenômeno imobiliário na cidade. Diversos novos empreendimentos foram construídos em sua extensão, principalmente nas regiões litorâneas da Barra da Tijuca e Recreio [Neto \(2002\)](#), e verificou-se um reajuste dos preços dos imóveis usados em outros bairros da cidade como Tijuca, Rio Comprido, Vila Isabel, e demais regiões do subúrbio como Madureira, Del Castilho e Campo Grande. A questão das favelas ainda não foi resolvida, e embora com crescimento planar limitado seja pela saturação do espaço urbano em seu entorno, seja pelas limitações geográficas das encostas dos morros, seu crescimento continua agora verticalmente, acompanhando a tendência das residências urbanas.

O mapa de uso do solo na p. [10](#) permite-nos conhecer a ocupação urbana da cidade. Segundo é exibido na legenda do mapa, as áreas residenciais ocupam 29,6% da área total da cidade, enquanto que a área de mata é apenas ligeiramente superior, 31,6%.

Geograficamente, a cidade é limitada ao norte pelo Rio Pavuna, fazendo fronteira com as cidades da Baixada Fluminense, Nilópolis, São João de Meriti e Duque de Caxias. À oeste a limitação é feita pelo contorno do Rio Guandu, uma das fontes de abastecimento de água para a região metropolitana, fronteira com a cidade de Itaguaí. O limite sudoeste é seu litoral com a Baía de Sepetiba e ao sul com o Oceano Atlântico, sendo quase a metade desse, no sentido oeste, pertencente à Restinga de Marambaia, área não habitada por ser propriedade militar da Marinha Brasileira, e no sentido Leste encontram-se um dos principais ativos turísticos da cidade, suas praias banháveis, cujos nomes, Recreio, Barra da Tijuca, Ipanema e Copacabana, tomam emprestado dos respectivos bairros. Finalmente, o limite à leste é o litoral com a Baía de Guanabara, cujo recebimento de esgoto residual e industrial da região metropolitana excede à capacidade de renovação da água pela comunicação com o Oceano Atlântico, resultando

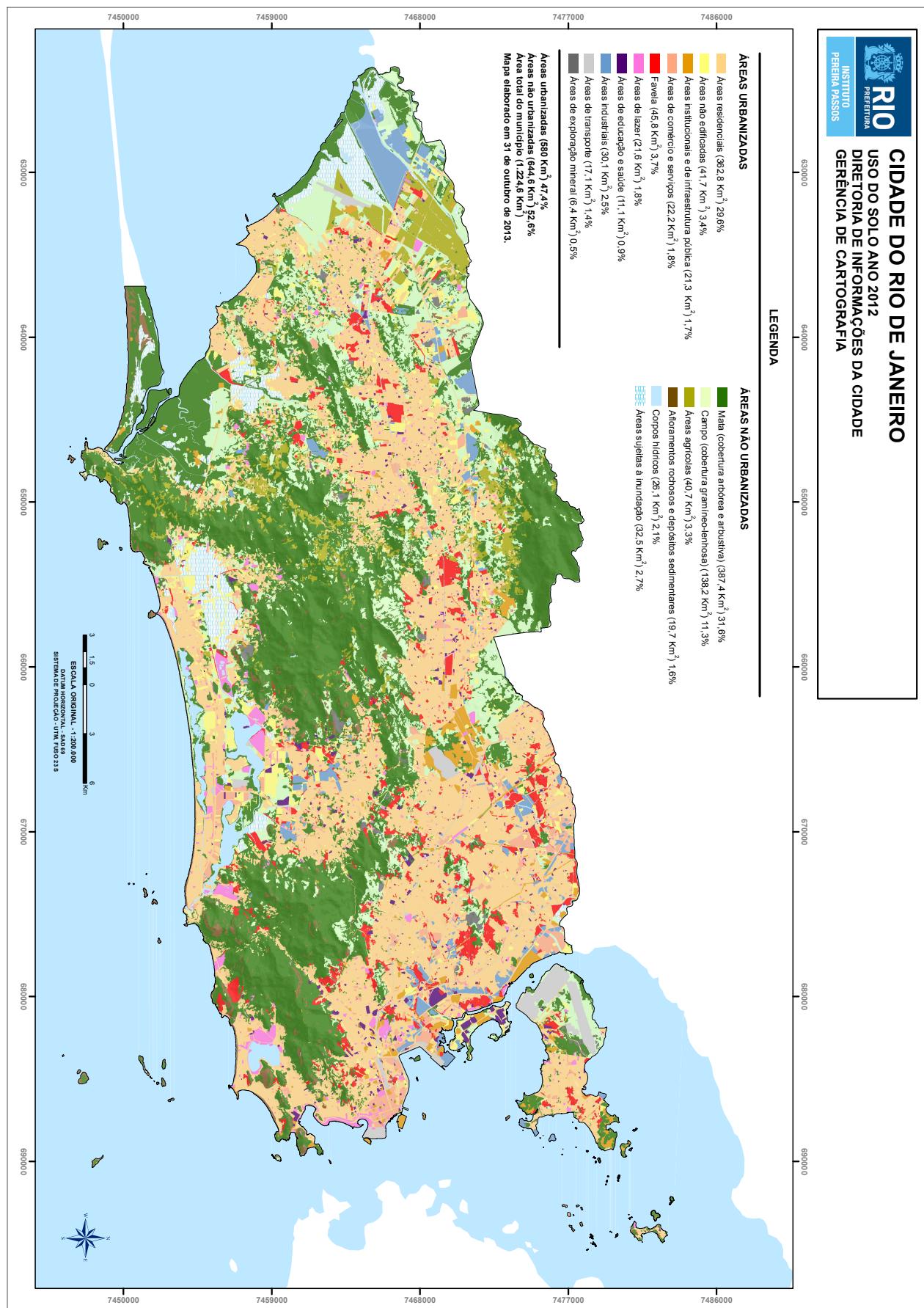


Figura 2.2: Mapa de uso do solo da cidade do Rio de Janeiro. Fonte: Instituto Pereira Passos, 2012.

em condições desfavoráveis de banho nas praias banhadas pela Baía.

Nota-se facilmente na fig. 2.2 3 grandes áreas de floresta que coincidentemente repousam sobre maciços, seções da crosta terrestre composta por cadeias de montanhas demarcadas por falhas e fendas. A mais ao norte, na vizinhança com as cidades da Baixada Fluminense, encontra-se o Maciço do Gericinó-Mendanha, também conhecido como Serra do Mendanha ou Serra de Madureira. Ao centro da cidade temos o Maciço da Pedra Branca, detentor não apenas do pico homônimo que é o ponto mais alto da cidade, 1025 metros acima do nível do mar, mas também da maior área coberta da Mata Atlântica em seu estado original⁴. Por fim, a área mais a sudeste é o Maciço da Tijuca, onde encontra-se a Floresta da Tijuca. Considerada a maior floresta urbana do mundo essa área foi degradada nos séculos XVII e XVIII pela exploração de madeira e expansão de monoculturas, principalmente o café, até que os impactos ambientais de deterioração da qualidade do ar e escassez de água obrigaram as autoridades em 1861 à execução de um plano de reflorestamento, com a desapropriação de fazendas e elevação de seu entorno como área de proteção ambiental, uma das primeiras do mundo⁵. Atualmente, os três maciços sofrem com a ocupação irregular de suas encostas, sendo o mais afetado o Maciço da Tijuca, por estar instalado no coração urbano da cidade. Ao longo de seu entorno cresceram algumas das grandes favelas da cidade como Rocinha e Vidigal ao sul, e Salgueiro, Boréu e Turano ao norte.

Embora os termos Zona Sul, Zona Norte e Zona Oeste sejam largamente utilizados pela população, eles não fazem parte de nenhuma classificação oficial da administração pública(Neto, 2002, p. 17). A divisão administrativa da cidade dá-se em Áreas de Planejamento, que por sua vez subdividem-se em Regiões Administrativas, cujo objetivo é uma administração mais eficiente dos serviços públicos de saúde, assistência social, lazer e educação, aplicados ao contexto geográfico, social e econômico local, do que

⁴Fonte: IBGE. URL: <http://biblioteca.ibge.gov.br/pt/biblioteca-catalogo?view=detalhes&id=440220>. Acessado em 30 de novembro de 2014.

⁵Fonte: ICMBio. URL: <http://www.parquedatijuca.com.br/#historia>. Acessado em 30 de novembro de 2014.

seria se fosse centralizado. Por fim, as Regiões Administrativas são divididas em bairros, sendo essa última comumente utilizada em estudos e comparações, como o Índice de Desenvolvimento Humano Municipal. Uma visualização da divisão administrativa da cidade é apresentada na fig. A.1.

Nota-se pela fig. A.1 que a extensão territorial das Regiões Administrativas e bairros aumentam conforme caminha-se para oeste, parte interior e menos desenvolvida da cidade, em função do histórico da ocupação urbana mencionada no início desse capítulo.

A seguir, abordaremos o modelo matemático aplicado à mensuração de características que formam o preço de um bem de consumo, Modelos Hedônicos.

2.2 Modelos Hedônicos

Modelo Hedônico é um método de estimativa dos valores implícitos das partes constituintes de um bem (Long et al., 2007, p. 2). (Macedo, 1999, p. 2) cita como uma das primeiras aplicações dessa técnica a análise de preços de automóveis feita por Griliches e Dhrymes na década de 1960, decompondo automóveis em tamanho, potência e acessórios, e também uma aplicação no mercado de imóveis na mesma década feita por Bailey, Muth e Nourse.

Ao escopo desse trabalho, Modelos Hedônicos têm sido indispensáveis para a avaliação do mercado imobiliário (Long et al., 2007, p. 2), cuja importância desse tema respalda-se no impacto dos estudos macroeconômicos, no interesse de agentes governamentais como *termômetro* de fenômenos sociais como crime, trânsito, oportunidades de emprego e constituição demográfica Ismail e MacGregor (2006), além de avaliação de investimentos em benfeitorias públicas e programas sociais (Long et al., 2007, p. 2). De igual forma o setor privado aborda Modelos Hedônicos em precificação de imóveis para o estudo de viabilidade de empreendimentos e determinação dos itens mais valorizadas pelos consumidores (Neto, 2002, p. 35). Embora não haja consenso na literatura sobre

quais variáveis devam ser incluídas na modelagem hedônica, usualmente essas variáveis podem ser classificadas três categorias ([Long et al., 2007](#), p.3), ([Macedo, 1999](#), p. 68):

1. Estruturais: aquelas pertencentes unicamente ao imóvel como ano de construção, número e tipo de cômodos, posição relativa à rua de acesso, benfeitorias como material utilizado no piso, presença de varanda, etc;
2. Acessibilidade: características derivadas diretamente da localização do imóvel como benfeitorias públicas, acesso à meios de transporte, unidades de saúde, escolas e outras de contexto geoespacial.
3. Socioambientais: características derivadas por proprietários e vizinhança como renda, desenvolvimento educacional, participação política e criminalidade;

A relação entre o preço do imóvel e suas características, expressas nas variáveis acima mencionadas, é comumente avaliado por Regressão Linear, onde o valor do bem é a variável dependente e suas partes constituintes são as variáveis independentes, e o erro residual entre o valor predito e o valor real é justificado em parte pelas características existentes que afetam o valor mas não expressas diretamente no modelo ([Long et al., 2007](#), p. 4):

$$\hat{Y}_i = \alpha_i + \sum_k \beta_{ki} S_{ki} + \sum_p \gamma_{pi} N_{pi} + \sum_j \lambda_{ji} L_{ji} + \epsilon_i \quad (2.1)$$

Na equação acima, \hat{Y}_i , com $i = 1, 2, \dots, n$, onde n é o número de observações do conjunto de dados, representa o preço estimado de um imóvel. O vetor S representa as variáveis estruturais, N é o vetor das variáveis socioambientais, e L o vetor das variáveis de acessibilidade, todos combinados pelos respectivos coeficientes β , γ e λ . α indica os coeficientes constantes e ϵ o erro residual decorrente de informação não explicada pelo modelo.

A eq. (2.1) pode ser representada em uma forma mais compacta agrupando-se todas as variáveis em uma única matriz Z , e os coeficientes em um único vetor coluna ρ (Long et al., 2007, p. 4):

$$\hat{Y} = \alpha + \sum Z\rho + \epsilon \quad (2.2)$$

Entretanto, debate-se a observância das premissas para aplicabilidade de Regressão Linear em Modelos Hedônicos para a estimativa de uma variável dependente da localização, como é o preço de um imóvel. Podemos esperar que dois imóveis com as mesmas características estruturais tenham valores diferentes se localizados um próximo ao centro da cidade e outro próximo à periferia, e o surgimento de um novo imóvel pode ter seu valor afetado em função dos valores de outros imóveis próximos, conforme a "primeira lei da geografia", observação dos fenômenos espaciais cunhada por Waldo Tobler:

*"Everything is related to everything else, but near things are more related than distant things."*⁶

Essa relação que afeta as unidades de informação espacialmente relacionadas denomina-se autocorrelação espacial. Um outro efeito espacial é a heterogeneidade espacial, onde o domínio de valores da informação podem ser segregados em função da localização (Ismail, 2006, p. 2-3). Até a popularização dos Sistemas de Informações Geográficas, SIG⁷, os efeitos derivados da autocorrelação espacial não recebiam a devida atenção nas modelagens hedônicas de então (Ismail, 2006, p.1). Pesquisadores no assunto têm apresentado propostas de evolução da modelagem hedônica para minimizar os efeitos da autocorrelação espacial, resumidos no trabalho de Ismail (2006) que é uma revisão da literatura a respeito. Pace et al (1998b) apud (Ismail, 2006, p. 6-9) sugere duas

⁶Tobler W., (1970) "A computer movie simulating urban growth in the Detroit region". Economic Geography, 46(2): 234-240.

⁷Do inglês Geofraphic Information Systems, GIS. Tradução nossa.

opções para lidar com a autocorrelação espacial na modelagem hedônica de imóveis. A primeira é a construção de variáveis independentes que representem distância a pontos de interesse, concomitantemente evitando-se a multicolinearidade entre elas. A segunda opção é a modelagem do erro residual ϵ que por sua vez pode ser feito por duas abordagens diferentes, uma pela especificação de uma matriz W de pesos espaciais onde a entrada m_{ij} relaciona o grau de influência da unidade j à unidade i e que é a utilizada nesse estudo, e outra abordagem por uso de técnicas de geo-estatística.

2.3 Modelos hedônicos realizados no Brasil

Dentre os quatro artigos consultados sobre modelagem hedônica nas cidades brasileiras, o mais antigo, [Macedo \(1999\)](#), faz uma comparação da modelagem tradicional, que toma apenas os aspectos intrínsecos dos imóveis, nas suas formas funcionais linear, semi logarítmica e dupla logarítmica, e a modelagem espacial, que leva em conta os efeitos da autocorrelação e heterogeneidade no mercado imobiliário da cidade de Belo Horizonte, capital do estado de Minas Gerais, utilizando *Box-Cox*, um procedimento estatístico para estabilização dos parâmetros do modelo. Foram analisados apartamentos residenciais em uma região topográfica e socioeconomicamente homogênea, dentro de uma área estipulada de $16Km^2$. Entretanto, a área escolhida não é determinada no texto, sendo que a cidade de Belo Horizonte possui um total de $331Km^2$. A regressão espacial considerou duas matrizes de pesos W : uma binária, onde a entrada $w_{ij} = 1$ se a distância entre os imóveis i e j for inferior a $1,5Km$, e zero se superior, e uma segunda matriz cuja entrada w_{ij} é inversamente proporcional à distância. Os imóveis da amostra possuem uma distância média de $2,5Km$, e máxima de $6,5Km$. Os dados foram obtidos a partir dos arquivos de impostos da Prefeitura Municipal e os preços de pesquisa realizada pela Universidade Federal de Minas Gerais em 1995, onde as características utilizadas no modelo são a área e ano de construção do imóvel, para um total de 53

observações. À exceção das distâncias entre os imóveis, nenhuma outra característica socioambiental e espacial foi utilizada. O estudo de [Macedo \(1999\)](#) rejeita a hipótese nula de que não há autocorrelação espacial entre os preços de imóveis da amostra selecionada e salienta a necessidade de incorporar os efeitos espaciais nos estudos de mercado imobiliário. Dentre as análises tradicionais, atesta que a forma funcional dupla logarítmica descreve melhor a relação entre preço e características do que as outras duas.

A cidade do Rio de Janeiro é objeto de estudo em [Neto \(2002\)](#), onde avalia 5.953 imóveis anunciados em 64 lançamentos no ano 2000, sendo 43% dos dados referentes aos bairros Recreio, Barra da Tijuca e Jacarépaguá. Entretanto, seu conjunto de dados é uma agregação em valores médios de preços dos lançamentos de um condomínio por número de quartos e se é ou não cobertura, gerando um quantidade útil de 120 observações para a modelagem de 26 características, sendo cinco estruturais, uma de acessibilidade e o restante socioambiental. A característica de acessibilidade modelada é a distância do centro geográfico do bairro do imóvel ao centro geográfico do bairro Centro. As observações foram divididas aleatoriamente em 90 para o conjunto de treinamento e 30 para o conjunto de teste. O modelo hedônico adotado foi o tradicional e utilizou o Erro Médio Quadrático⁸ aplicado ao conjunto de teste para comparar a performance das formas funcionais linear, semi e dupla logarítmica. [Neto \(2002\)](#) desenvolve seu estudo primeiramente determinando a significância estatística das variáveis agrupadas, por classificação própria, em físicas, segurança, vizinhança e dicotômicas, utilizando Teste de Wald. Verifica que apenas as variáveis de vizinhança não são estatisticamente significantes para a forma funcional linear. A seguir, constrói o modelo pela técnica geral para específico, começando por utilizar todas as variáveis disponíveis e removê-las uma a uma, à medida que falham no teste de significância estatística. Por fim, conclui que a forma funcional dupla logarítmica é a única que não apresentou problemas nos

⁸Erro Médio Quadrático: $1/n \sum_{i=1}^n (y - \hat{y}_i)^2$, onde n é a quantidade de observações testesdas, y preço informado e \hat{y} o preço estimado do modelo.

testes de heterocedasticidade e linearidade, seguida da forma semi logarítmica e linear, e que as características mais relevantes para a estimação do preço são a distância ao centro da cidade, tamanho em área, quantidade de quartos, índice de roubos, quantidade de itens de utilidade e lazer no bairro, quantidade de unidades com mesmo número de quartos no condomínio, e disponibilidade de serviços de hotelaria.

O estudo de [Rubens A. Dantas \(2007\)](#) trata especificamente da importância da autocorrelação espacial dos preços de imóveis, tomando como exemplo uma amostra de 228 apartamentos financiados pela Caixa Econômica Federal na cidade de Recife consultados entre junho de 2000 e junho de 2002. Seu conjunto de variáveis compreende 4 variáveis estruturais intrínsecas ao imóveis e 4 relacionadas aos respectivos condomínios. As variáveis de acessibilidade são as distâncias a três pólos de influência na cidade: a Praia de Boa Viagem, Parque Jaqueira e centro da cidade. A renda média do chefe de família por bairro é única variável socioambiental, totalizando 14 variáveis ao modelo. Seus resultados comparam o modelo tradicional com um modelo estendido adicionando-se uma variável indicadora da influência dos imóveis vizinhos a partir de uma matriz de pesos espaciais W onde a entrada w_{ij} representa o inverso da distância entre os imóveis i e j . Embora o modelo tradicional apresente bons resultados estatísticos, conclui-se que o modelo estendido apresenta a uma interpretação mais acurada dos coeficientes resultantes, principalmente pela redução dos respectivos desvios padrão. Ressalta-se que a utilização de variáveis espaciais no modelo tradicional não é suficiente para a anulação da autocorrelação dos preços e que a abordagem da matriz W ajuda a incorporar os aspectos localizacionais não explicitados nas variáveis.

[Favero et al. \(2008\)](#) apresenta uma abordagem ligeiramente diferente dos demais estudos com a especificação logarítmica em equações de dois estágios de Rosen para verificar quais características mais interferem nas condições de oferta e demanda e compara a importância relativa de cada uma aos perfis sociodemográficos. Seu conjunto de dados estruturais foram coletados aleatoriamente de anúncios de lançamentos do jornal *Folha de*

São Paulo e sites <http://www.imovelweb.com.br/> e <http://www.planetaimovel.com.br/>⁹ de janeiro a dezembro de 2004, totalizando 1860 apartamentos residenciais da cidade de São Paulo. A análise fatorial para determinação dos três perfis sociodemográficos - baixo, médio e alto - é feita com base em 11 variáveis relacionadas a esse tema onde destacamos Renda Familiar, Escolaridade, Taxa de Mortalidade Infantil, População e Densidade Demográfica. As variáveis estruturais somam 15 sendo 11 dicotômicas determinando a presença de utilidades de lazer e segurança no condomínio, se altura relativa ao solo é superior à metade do edifício. As outras são área, quantidade de vagas de garagem, quartos e banheiros no imóvel. Outras variáveis socioambientais e de acessibilidade são apresentadas ao contexto da análise sobre influência na oferta e demanda, totalizando ao final 26 variáveis. A conclusão apresenta os coeficientes das variáveis segmentados pelos perfis sociodemográficos, destacando que o perfil alto é o que possui o menor número de variáveis representativas, o que demonstra a importância da construção da modelagem hedônica apropriada aos citados perfis. Os testes estatísticos não indicaram problemas de heterocedasticidade, autocorrelação, e para as variáveis explicativas, não foi verificado multicolinearidade.

A seguir, fazemos uma introdução sobre o modelo matemático por trás da modelagem hedônica, regressão linear.

2.4 Regressão Linear

Fenômenos da natureza, ou aqueles provocados pela ação do homem, podem ser estudados decompondo-os em variáveis numéricas ou categóricas, a fim de observar as relações entre elas e, possivelmente, identificar padrões comportamentais ou estimar resultados com base em suposições. Essas relações podem ser determinadas elencando-se um ou mais dessas variáveis como as variáveis de interesse a serem expressas em função

⁹O endereço <http://www.planetaimovel.com.br> atualmente redireciona o usuário ao site ZAP Imóveis.

das demais variáveis restantes. Ao longo desse estudo denominaremos as variáveis de interesse por **variáveis dependentes**, cujo nome apropriadamente indica uma relação de dependência com as demais variáveis, denominadas **variáveis independentes**, (Andersen e Skovgaard, 2010, p.2). Como o escopo desse estudo limita-se a apenas uma variável independente, representamos uma observação qualquer dessa variável por y e as respectivas n variáveis independentes pelo vetor $x = (x_1, x_2, \dots, x_n)$. As matrizes $Y_{m,1}$, $X_{m,n}$ representam o conjunto de m observações das variáveis dependentes e independentes, respectivamente, sendo uma determinada observação i indicada por y_i , x_i , $i \in \{1,2,\dots,m\}$ e uma determinada variável independente j por x_j , $j \in \{1,2,\dots,n\}$. Finalmente, x_{ij} representa uma observação específica i da variável independente j .

Um dos objetivos da compreensão de um fenômeno é a capacidade de estimar um valor da variável dependente, indicado por \hat{y} , a partir de uma nova observação $x \notin X$, esperando-se seguir as relações naturalmente presentes em Y e X . Quando \hat{y} pode assumir um valor contínuo, $\hat{y} \in \mathbb{R}$, chamamos a essa estimativa de **Regressão** (Bishop, 2006, p.3), (Hastie et al., 2013, p.4). A estimativa de variáveis dependentes categóricas, aquelas que representam a pertinência a um determinado conjunto, é chamado **Regressão Logística** e não é escopo desse estudo.

(Hastie et al., 2013, p.44), (Bishop, 2006, p.138) e (Murphy, 2012, p.127) definem **Regressão Linear** a classe de modelos cuja função de regressão da variável dependente \hat{y} é uma combinação linear dos parâmetros $\beta_i \in \mathbb{R}$:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n \quad (2.3)$$

Segundo (Andersen e Skovgaard, 2010, 9-11), para os casos em que x_j é categórica, $x_j \in c_1, c_2, \dots, c_k$, não a utilizamos diretamente no modelo, mas a substituímos por variáveis **dicotômicas**, assim denominadas para representar a pertinência à categoria identificada em x_j . Tal substituição é feita criando-se k variáveis indicadoras para as

$k + 1$ categorias possíveis de x_j . Uma das categorias não é explicitamente modelada para evitar multicolinearidade entre as variáveis resultantes, por ser a combinação de todas as outras quando assumem concomitantemente o valor 0.

$$I_{r=\{1,\dots,k\}}(x_j) = \begin{cases} 1, & \text{se } x_j = c_r \\ 0, & \text{senão} \end{cases} \quad (2.4)$$

O coeficiente β_0 na equação 2.3 representa um deslocamento fixo do modelo, valor a ser assumido para o caso em que $\forall j : x_j = 0$, denominado *bias*¹⁰. Por conveniência, assumimos uma nova variável independente $x_0 = 1$, fazendo o conjunto de variáveis independentes ter dimensões $X_{m,n+1}$, com o propósito de reduzir a equação 2.3 para a forma:

$$\hat{y} = \sum_{j=0}^n \beta_j x_j \quad (2.5)$$

Entretanto os parâmetros β são desconhecidos e também precisam ser estimados. Podemos fazê-lo a partir de um subconjunto das observações Y e X , a quem denominamos **conjunto de treinamento** (Bishop, 2006, p.4), (Hastie et al., 2013, p.1). A utilização dos valores atuais Y de forma a permitir uma avaliação da eficiência da estimação de β classifica esse tipo de aprendizado como **supervisionado** (Hastie et al., 2013, p.2). Reciprocamente, aprendizados **não supervisionados** são aqueles que procuram identificar estruturas em X e não dependem de Y para avaliar o aprendizado.

Segundo (Hastie et al., 2013, p.12), um dos métodos mais populares utilizado para a estimação de β , conhecido como **Método dos Mínimos Quadrados**¹¹, consiste em minimizar a soma dos quadrados dos erros residuais $\epsilon_i = y_i - \hat{y}_i$:

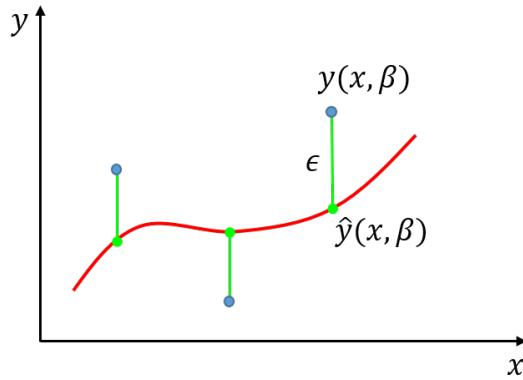
¹⁰Não há tradução clara do significado desse termo conforme (Bishop, 2006, p.138) para a Língua Portuguesa.

¹¹Do inglês *Least Squares*, tradução nossa.

$$\hat{\beta} = \operatorname{argmin}_{\beta} \sum_{i=1}^m \left(\sum_{j=0}^n \beta_j x_{ij} - y_i \right)^2 \quad (2.6)$$

Importante notar que o erro residual não é a distância euclidiana entre y e \hat{y} , mas tão somente a diferença escalar entre as duas variáveis, como pode ser visto na fig. 2.3.

Figura 2.3: Intuição do método Mínimos Quadrados. Fonte: ([Bishop, 2006](#), p.6), adaptado.



A equação 2.6 pode ser descrita em forma matricial, onde denominamos $RSS(\beta)$ a *Soma dos Resíduos Quadrados*¹²:

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^m \left(\sum_{j=0}^n \beta_j x_{ij} - y_i \right)^2 \\ &= \sum_{i=1}^m (\beta^T x_i - y_i)^2 \\ &= (Y - X\beta)^T (Y - X\beta) \end{aligned} \quad (2.7)$$

Derivando-se (2.7) com respeito a β temos:

$$0 = X^T (Y - X\beta) \quad (2.8)$$

¹²Do inglês *Residual Sum of Squares*, *RSS* ([Hastie et al., 2013](#), p.12)

Se $X^T X$ for não singular, então a solução única é dada por:

$$\hat{\beta} = (X^T X)^{-1} X^T Y \quad (2.9)$$

Se $X^T X$ for singular, então a equação 2.9 admite mais de uma solução, o que indica dependência linear entre as variáveis X . Deparamo-nos então com uma das primeiras premissas para a utilização do Método de Mínimos Quadrados para estimativa de β , que é a independência linear entre as variáveis independentes.

Finalmente, de posse de uma nova observação $z \notin X$ podemos estimar o valor da variável dependente $\hat{y}(z)$ com:

$$\hat{y}(z) = z^T \hat{\beta} \quad (2.10)$$

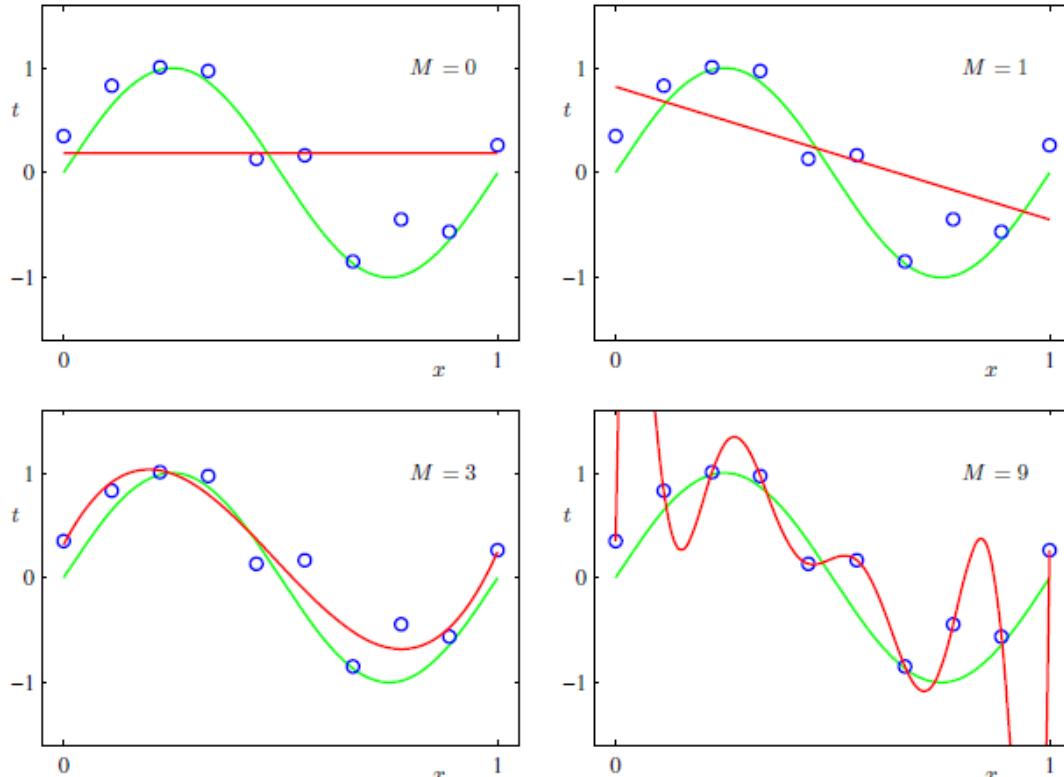
(Bishop, 2006, p.140-143) e (Andersen e Skovgaard, 2010, p.178-180) demonstram que o Método dos Mínimos Quadrados é derivado da Estimativa por Máxima Verossimilhança sob a premissa de que o erro residual $e_i = y_i - \hat{y}_i$ segue uma distribuição Normal.

Sob determinadas condições das escolhas das variáveis independentes em X e o número m de observações, podemos incorrer em um problema denominado *overfitting*, que implica na estimativa de $\hat{\beta}$ fazer $\hat{y}(x)$ aproximar-se demasiadamente bem de $y(x)$ para $x \in X$ mas não aproximar bem em novas observações $x \notin X$. (Bishop, 2006, p.4-9) ilustra esse problema com uma aplicação bem simples de Regressão Linear que é ajustar uma curva polinomial de ordem M , $\hat{y} = \sum_{i=0}^M \hat{\beta} x^i$, a partir de dados gerados pela função $y = \sin(x)$, com um ruído aleatório aplicado. Nesse exemplo em que temos apenas uma variável em X , é proposta a construção de novas variáveis x^2, x^3, \dots, x^M para o ajuste da curva polinomial. Importante notar que essas novas variáveis não apresentam dependência linear com x , respeitando a premissa para que $X^T X$ seja não

singular.

Vê-se que na fig. 2.4 que conforme M aumenta, o polinômio resultante aproxima-se a x até que para $M = 9$ o polinômio passa exatamente sobre cada um dos dados originais mas fica evidente que se distanciará de novas observações.

Figura 2.4: Exemplo de sobreajuste. Fonte: ([Bishop, 2006](#), p.7), adaptado.



De fato, a definição de sobreajuste apresentada por ([Mitchell, 1997](#), p.67, adaptado) diz:

Given a hypothesis space H , a hypothesis $\hat{y} \in H$ is said to overfit the training data if there exists some alternative hypothesis $\hat{y}' \in H$, such that \hat{y} has smaller error than \hat{y}' over the training examples, but \hat{y}' has a smaller error than \hat{y} over the entire distribution of instances.

Tal definição é empiricamente demonstrada na fig. 2.4 em que notamos que para

$M = 3$ a soma dos erros residuais para novas observações será menor do que para $M = 9$.

Podemos verificar o sobreajuste de uma regressão medindo gráfica e numericamente o comportamento de uma medida de performance da capacidade de generalização de \hat{y} . Usualmente utiliza-se a *Raiz do Erro Médio Quadrático*¹³ para esse objetivo, que é uma extensão da equação 2.6 definida como (Bishop, 2006, p.7, adaptado):

$$RMSE = \sqrt{\frac{\sum_{i=1}^m (\hat{y}_i - y_i)^2}{n}} \quad (2.11)$$

A divisão por n nos permite comparar diferentes tamanhos de conjuntos de treinamento e a raiz quadrada por sua vez garante que a $RMSE$ seja medida na mesma escala e unidade da variável dependente y (Bishop, 2006, p.7).

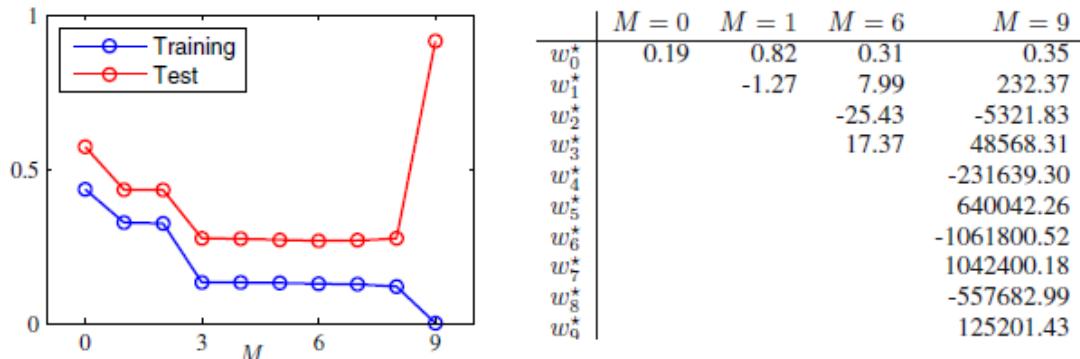
Entretanto, escolher os parâmetros cujo $RMSE$ seja mínimo utilizando o próprio conjunto de treinamento para avaliação da performance pode levar-nos ao sobreajuste, escolhendo os parâmetros do modelo que se ajustam demasiadamente bem ao conjunto de dados mas sem grande poder de generalização para novas observações. Essa situação pode ser evitada pela técnica da **Validação Cruzada** separando cada conjunto de dados de m observações em dois subconjuntos distintos, um para treinamento, e outro para validação. A estimativa dos parâmetros $\hat{\beta}$ é feita com base no subconjunto de treinamento e a performance é avaliada pela execução do modelo somente no subconjunto de validação. Importante destacar que a avaliação da performance em uma única separação arbitrária dos subconjuntos de treinamento e validação pode ainda assim incorrer em sobreajuste ao acaso. Tal risco é mitigado pela extensão da técnica de validação cruzada denominada *K-Fold*, onde separamos nosso conjunto de m observações em k subconjuntos de tamanhos iguais m/k , e cada subconjunto representa um *fold*. A performance é então obtida como a média das performances apuradas nas k iterações

¹³Do inglês *Root Mean Square Error*, $RMSE$, tradução nossa. Manteremos a sigla em inglês $RMSE$ por conveniência.

em que em cada *fold* é atribuído como conjunto de validação da inferência construída com os outros *folds* restantes como conjunto de treinamento. ([Bishop, 2006](#), p.32-33).

A fig. 2.5 demonstra a importância da Validação Cruzada na avaliação da performance da generalização de \hat{y} para novas observações. Continuando com o exemplo do ajuste de um polinômio de ordem M , calculamos a raiz do erro médio quadrático, $RMSE$, para cada M , sobre o próprio conjunto de treinamento e sobre um conjunto reservado de verificação. À medida que M aumenta a performance melhora em ambos, mas para $M = 9$ fica claro o sobreajuste quando avaliado sobre o próprio conjunto de treinamento e seu impacto na pobre performance sobre o conjunto de verificação. Um efeito prático do sobreajuste sobre os coeficientes $\hat{\beta}$ é esses assumirem valores absolutos expressivos, como pode ser visto no lado esquerdo da fig. 2.5.

Figura 2.5: Intuição do sobreajuste (overfitting) para a estimativa de uma função polinomial em x . Fonte: ([Bishop, 2006](#), p.8), adaptado.



Em função da expressiva magnitude que os coeficientes $\hat{\beta}$ podem alcançar devido ao sobreajuste, uma alternativa é aplicar sobre a estimativa de β uma penalidade proporcional ao crescimento absoluto dos próprios coeficientes, conhecido como **Regularização**¹⁴ ([Bishop, 2006](#), p.10,144-147) ou **Métodos de Encolhimento**¹⁵ ([Hastie et al., 2013](#), p.61-69). Abaixo apresentamos a estimativa de β com uma penalização denominada

¹⁴Do inglês *Regularization*, tradução nossa.

¹⁵Do inglês *Shrinkage Methods*, tradução nossa.

Ridge ([Hastie et al., 2013](#), p.63):

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=1}^m (y_i - \beta_0 - \sum_{j=1}^n \beta_j x_{ij})^2 + \lambda \sum_{j=1}^n \beta_j^2 \right) \quad (2.12)$$

O parâmetro λ controla o grau de penalidade a ser aplicado na estimativa de β . Quanto maior λ , maior será a penalidade aplicada, limitando proporcionalmente o crescimento absoluto de β . Uma penalidade excessiva pode inverter totalmente o objetivo da regularização e causar o chamado *subajuste*¹⁶, cuja consequência também é a perda de generalização de \hat{y} para novas observações, inclusive para o próprio conjunto de treinamento ([Andersen e Skovgaard, 2010](#), p.38).

O capítulo a seguir apresenta os métodos, técnicas e ferramentas utilizados para a construção e tratamento das variáveis utilizadas no estudo.

¹⁶Do inglês *Underfitting*, tradução nossa.

empty

Capítulo 3

Metodologia

Neste capítulo abordaremos a construção do modelo hedônico para responder as perguntas propostas no capítulo Introdução. Explicaremos a origem dos dados, identificação de anomalias, valores ausentes e o tratamento aplicado para sanar essas ocorrências, além das ferramentas e técnicas utilizadas para as etapas de coleta, tratamento e manipulação dos dados.

3.1 Obtenção dos dados

3.1.1 Variáveis estruturais básicas

O ZAP Imóveis¹ é um serviço de classificados de imóveis na Internet onde os proprietários, sejam pessoas físicas ou jurídicas, os anunciam detalhando informações que consideram relevantes a quem procura comprá-los. Utilizamos esse classificado como fonte de dados das informações estruturais dos imóveis e construímos um programa de computador para automatizar a busca que é feita em duas etapas. Reiteramos a política de Termo de Uso² do classificado ZAP Imóveis em que proíbe a cópia e

¹ZAP Imóveis: <http://www.zapimoveis.com.br/>.

²Política de Termos de Uso do ZAP Imóveis: <http://www.zapimoveis.com.br/informacao?opcao=termouso>. Acessado em 20 de dezembro de 2014.

utilização dos dados para fins comerciais. A manipulação dos dados é exclusivamente para a construção desse estudo, de interesse e acesso público, sem vínculo comercial com qualquer pessoa física ou jurídica, citada ou não.

Na primeira etapa, partimos de uma página inicial com a lista de anúncios do ZAP Imóveis e a registramos no banco de dados manualmente, como uma página não visitada. Quando o programa é executado, ele busca no banco de dados se existem páginas não visitadas. Como só há uma página não visitada, a primeira que cadastramos manualmente, ela é selecionada e a busca começa. Essa página inicial é a primeira de uma listagem com anúncios de imóveis da cidade do Rio de Janeiro, e pode ser acessada pela URL³ "<http://www.zapimoveis.com.br/venda/apartamentos/rj+rio-de-janeiro/>". A fig. 3.1 apresenta um recorte arbitrário dessa lista para exemplo.

³URL: Uniform Resource Locator. Mais informações em <http://www.ietf.org/rfc/rfc1738.txt>.

Figura 3.1: Exemplo da lista de anúncios para a cidade do Rio de Janeiro. Fonte: ZAP Imóveis.

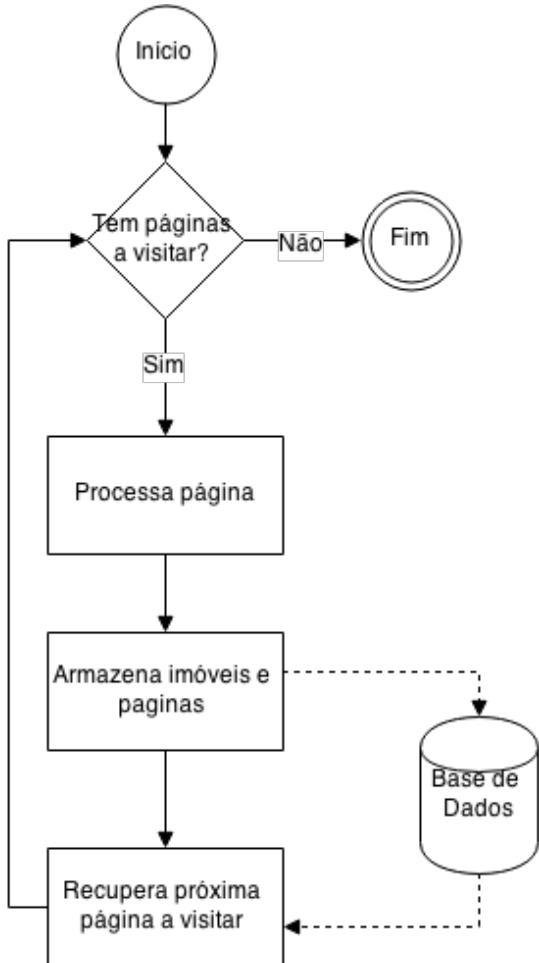
The screenshot displays four horizontal real estate listings from the ZAP Imóveis website:

- Anunciante Premium**: PECHINCHA - RIO DE JANEIRO/RJ, RUA PAULO MOREIRA DA SILVA. Price: R\$ 255.000. Features: 2 dorms, 1 garage, 55m². Advertiser: FRANCISCO XAVIER Sua Garantia Imobiliária. Date: 06/05/2014.
- JACAREPAGUA - RIO DE JANEIRO/RJ, RUA MIN GABRIEL DE PIZA**. Price: R\$ 336.000. Features: 2 dorms, 2 garages, 46m². Advertiser: GI GOLDEN IMÓVEIS. Date: 06/05/2014.
- PECHINCHA - RIO DE JANEIRO/RJ, ESTRADA TINDIBA,DO**. Price: R\$ 222.000. Features: 1 dorm, 1 garage, 42m². Advertiser: PRADA IMOBILIÁRIA. Date: 05/05/2014.
- TAQUARA - RIO DE JANEIRO/RJ, ESTRADA RODRIGUES CALDAS**. Price: R\$ 238.000. Features: 2 dorms, 1 garage, 54m². Advertiser: LUIZ FELIPE ALO MAIA. Date: 08/02/2014.

Essa listagem de anúncios serve para resumir ao usuário do site um número grande de ofertas, e caso ele se interesse por um imóvel específico, ao clicar no imóvel o usuário é transferido para o anúncio em si, onde pode observar as demais informações do imóvel. Para nosso programa, essa lista já contém algumas informações de interesse sobre os imóveis como preço, área, quantidade de quartos, quantidade de suítes, quantidade de garagens e finalmente a data de publicação, URL e o código identificador do anúncio. A

URL será usada para consultar demais informações do imóvel em outro momento. Além dos apartamentos, o programa busca links para as próximas páginas da listagem de anúncios, e os armazena no banco de dados como páginas ainda não visitadas. Ao determinar o fim da página em processamento, o algoritmo marca como visitada, busca a próxima página não visitada cadastrada no banco de dados e repete a busca, procedendo dessa forma sucessivamente até não haver mais páginas não visitadas. O registro de páginas visitadas e não visitadas permite que o algoritmo seja interrompido e retomado conforme necessidade. Quando não houver mais páginas a visitar, a busca termina e o banco de dados contém todos os imóveis anunciados com as informações disponíveis nas listagens e a URL de cada um. Abaixo apresentamos uma figura esquemática do funcionamento do programa.

Figura 3.2: Visão geral da 1.a etapa do algoritmo de captura.



Para maior detalhamento, extraímos o trecho abaixo sobre a captura das páginas e imóveis.

Listing 3.1: Método em Python da 1.a etapa de captura de imóveis.

```

1 def captura_apartamentos():
2     """
3     Executa a captura online dos imóveis no classificados_ZAP a partir
4     de uma
5     página inicial cadastrada no banco de dados.
6     """

```

```
6
7     log.info('Iniciando\_captura\_de\_apartamentos\_do\_zap.\_Veja\_log\_em\_'
8         'log.txt\'')
9
10    # Executa o loop enquanto houver paginas a visitar.
11    while len(paginas) > 0:
12        try:
13            url = paginas[0]['url']
14            log.info('Processando\_pagina "{}'.format(url))
15            apartamentos,paginas = __processar_pagina(url)
16            log.info('Pagina "{}"\_processada.'.format(url))
17
18            d.insere_pginas(paginas)
19            d.insere_imoveis(apartamentos)
20            d.registra_visita_pagina(url)
21        except Exception as e:
22            log.error(e)
23
24        finally:
25            paginas = d.pginas_nao_visitadas()
log.info('Nao\_ha\_paginas\_a\_visitar.\_Captura\_de\_apartamentos\_'
terminado.'
```

Ao fim da primeira etapa, temos as seguintes informações para todos os imóveis encontrados:

1. Código identificador do anúncio;
2. Nome da rua;
3. Bairro;
4. Área construída;

5. Quantidade de quartos;
6. Quantidade de suítes;
7. Quantidade de vagas de garagem;
8. Data da publicação do anúncio;
9. Valor anunciado do imóvel;
10. URL do anúncio;

A segunda etapa busca encontrar mais informações sobre os imóveis, disponíveis apenas nos anúncios específicos dos mesmos. A estrutura do programa é similar à primeira etapa. Primeiro, consultamos no banco de dados quais imóveis ainda não tiveram seus anúncios visitados. Para cada um deles, o programa acessa o anúncio utilizando a URL capturada na primeira etapa, processa a página em busca das informações e as registra no banco de dados. Essa iteração é repetida até não restarem anúncios a visitar. As informações coletadas na segunda etapa são:

1. Valor do condomínio;
2. Localização geográfica: latitude e longitude;
3. Descrição do imóvel escrita pelo anunciante: varanda, dependência de empregada, sala de jantar, etc;
4. Descrição do condomínio escrita pelo anunciante: piscina, sauna, portaria 24 horas, etc.

O processamento das páginas, tanto na primeira como na segunda etapa, para encontrar as informações sobre os imóveis e links para outras páginas com anúncios foi alcançado pelo uso da técnica conhecida como *Web Scrapping*, que consiste em

obter informações disponíveis na Internet, por meio de um algoritmo construído para percorrer a estrutura do recurso onde a informação está, identificá-la e descobrir outros recursos que possuam demais ocorrências da mesma informação. Para que se possa implementar o algoritmo com as decisões adequadas à sua estrutura é feito um estudo *a priori* no conteúdo onde as informações estão armazenadas buscando-se encontrar padrões de repetição e palavras chaves associadas. Tal atividade é bastante facilitada quando o meio em questão é uma página HTML⁴ que por definição⁵ é um documento de texto estruturado em segmentos identificados por *tags*, palavras chaves que identificam uma marcação HTML para que o navegador processe e exiba o conteúdo.

Dentro da linguagem de programação Python utilizada nesse estudo, recorremos à biblioteca Requests⁶ para o *download* das páginas HTML do ZAP Imóveis em memória, e a biblioteca BeautifulSoup⁷ para percorrer a estrutura HTML, identificar as informações desejadas e descobrir novas páginas com mais anúncios.

Por exemplo, no processamento da lista de anúncios os preços de cada imóvel é armazenados com o seguinte código HTML:

Figura 3.3: Trecho de HTML contendo preço de um imóvel. Fonte: ZAP Imóveis.

```

▼<div class="itemValor price">
  ▼<a class="valorOferta" href="http://www.zap.com.br/imoveis/oferta/Apartamento-Padrao-1-quartos-venda-RIO-DE-JANEIRO-PECHINCHA-ESTRADA-TINDIBA,DO-ID-5464480" onclick="rastreamentoNavegacaoModulo('ResultadoBusca10','Destaque');">
    <sup>R$ </sup>
    "222.000"
  </a>
  <br>
  ▶<div id="ctl00_ContentPlaceHolder1_rp0fertas_ctl09_imv1_pnlSimulador">...</div>
</div>
▶<div class="itemLogo fn">...</div>
```

O trecho em destaque na fig. 3.3 pertence ao imóvel anunciado na fig. 3.1, p. 31, no bairro Pechincha, valor de R\$ 222.000,00, mas a estrutura é a mesma para todos os imóveis listados. Durante o processamento, obtemos uma referência para cada imóvel e a partir dessa referência identificamos as informações individuais. Para o preço em

⁴Do inglês *HTML: HyperText Markup Language*.

⁵Definição de HTML: <http://www.w3.org/html>.

⁶Requests: <http://docs.python-requests.org/en/latest/>.

⁷BeautifulSoup: <http://www.crummy.com/software/BeautifulSoup>.

destaque, a referência do imóvel está armazenada no objeto `div`. O método `find` deste objeto retorna todas os `tags` do tipo especificado no primeiro parâmetro, em nosso caso, a tag `a`, que é um link para o anúncio individual do imóvel, mas segundo os desenvolvedores do ZAP Imóveis, também armazena o preço para exibição na lista. Essa `tag` tem como atributo `class` o valor `valorOferta`. Essa informação é passada no segundo parâmetro do método `find` para restringir a busca dos tags a somente aqueles com esse valor no atributo `class`. Como sabemos que só há uma instância com essas condições de busca, acessamos diretamente o conteúdo da tag com o método `contents`, que retorna um vetor de valores onde a primeira posição desse vetor faz referência ‘unidade da moeda, “R\$”, e a segunda posição, referenciada pela constante `IDX_PRICE`, o valor do anúncio. Por fim armazenamos o valor no objeto `apartamento`, sob o identificador `preco`, para referência futura. As outras informações do anúncio são obtidas de forma similar, e a implementação completa está no appendix B, p. 111.

Listing 3.2: Exemplo de *Web Scrapping* para captura da informação preço.

```

1   # Obtem preço do imovel.
2   apartamento['preco'] = div.find('a', class_='valorOferta').
                           contents[IDX_PRICE]
```

3.1.2 Demais variáveis estruturais

As descrições das características estruturais dos imóveis e serviços oferecidos pelo condomínio, coletados na segunda etapa descrita na p. 35 foram obtidas em seu formato original, um texto livre para descrição do imóvel escrito pelo anunciente. Não há consenso na literatura de Modelo Hedônico aplicado a precificação de imóveis sobre quais variáveis devam ser utilizadas no modelo de regressão (Long et al., 2007, p. 4), portanto, decidimos usar aquelas com maior quantidade de ocorrências. Para tal, primeiro transformamos as descrições em *tokens*, conjunto de caracteres tratados como um conjunto de interesse semântico, excluindo aqueles que não tem interesse semântico como pre-

posições, conjunções, conectivos, símbolos de pontuação, numerais e, principalmente, adjetivos, que expressam qualidades do imóvel mas sob o ponto de vista subjetivo do anunciante como "arejado", "claro", "lindo", "impecável", etc. Consideramos também a ocorrência de bigramas e trigramas, tuplas com dois e três *tokens* que encerram um valor semântico como por exemplo, "andar alto" e "sol manhã"⁸. Abaixo listamos a 20 maiores ocorrências dessa análise:

Tabela 3.1: Os 20 n-gramas de maior ocorrência nas descrições dos imóveis.

monogramas	bigramas	trigramas
andares	andares andares	andares andares condominio
condominio	andares condominio	empregada andares andares
sala	salao festas	armario cozinha armario
cozinha	armario cozinha	cozinha armario embutido
salao	armario embutido	sala ginastica salao
empregada	andares andar	condominio andares andar
andar	empregada andares	ginastica salao festas
armario	sala jantar	salao festas salao
varanda	cozinha armario	festas salao jogos
apartamento	condominio andares	varanda andares andares
festas	piscina playground	sala jantar varanda
piscina	sala ginastica	salao jogos sauna
churrasqueira	salao jogos	sala almoco sala
servico	ginastica salao	almoco sala jantar
playground	elevadores andares	varanda empregada andares
piso	festas salao	playground salao festas
social	varanda empregada	piscina playground quadra
sauna	sala almoco	condicionado armario cozinha
elevadores	varanda andares	quadra poliesportiva sala
armarios	jantar varanda	poliesportiva sala ginastica

Analisamos manualmente as 200 maiores ocorrências, anotando principalmente os monogramas candidatos a serem ignorados, e repetindo-se a análise. Verificamos que não encontramos características de interesse expressas nos trigramas. Complementamos a busca de características avaliando as descrições em seu contexto original, imprimindo em tela as ocorrências de cada token candidato a tornar-se uma variável, como podemos

⁸Como removemos as preposições, características como "sol da manhã" tornaram-se "sol manhã".

ver a seguir para o token "sol":

Listing 3.3: 10 ocorrências do token "sol" em seu contexto original.

```
Displaying 10 of 5414 matches:

. completa , fundos , sol da tarde , silencioso
2 quartos , 1 suite , sol da manha , varanda com
ondominio : camino del sol tipologia cobertura 2
. portaria 24 horas . sol da manha . vaga de gar
ericana . andar alto . sol da manha . bela vista
arejado , andar alto , sol da manha , vista livre
to luxo . andar alto , sol da manha e vistao mar
a na Area de serviCo , sol da manhA , vaga no con
dependencia completa , sol da manha , andar alto
eio dos bandeirantes , sol da manhA , piso cerAmi
```

Ao fim desse processo, selecionamos as características a serem representadas no modelo como variáveis **dicotômicas**, aquelas que expressam a presença de determinada característica atribuindo-se o valor 1, e ausência com o valor 0⁹.

Listing 3.4: Lista de variáveis estruturais dicotômicas.

```
1 andar_alto, andar_baixo, andar_inteiro, armario, banheira, blindex,
churrasqueira, closet, cobertura, copa, creche, dep_empregada,
duplex, elevador, elevador_privado, escritura, esquina,
est_visitantes, frente, fundos, granito, hidrometro, id,
indevassavel, jardim, lateral_, linear, mezanino, original,
piscina, planejad, play, porcelanato, portaria, recuado,
salao_festas, salao_jogos, sala_jantar, sauna, seguranca,
sol_manca, sol_tarde, terraco, triplex, unidades_andar, varanda,
vista_mar
```

⁹Variável dicotônica: http://www.lee.dante.br/pesquisa/amostragem/tipo_resposta.html.

Percorremos as descrições originais verificando a ocorrência de cada uma das características selecionadas e atribuindo o valor de respectiva variável dicotômica de acordo com a presença ou ausência da característica nas descrições. Para as características representadas pelos monogramas, essa busca é direta, bastando encontrar a palavra na descrição do imóvel. Mas para as características expressas em bigramas, precisamos considerar as diversas formas que o anunciante pode ter utilizado no momento da escrita do texto, incluindo possíveis erros comuns de digitação. Para esse objetivo, utilizamos a busca por expressões regulares, uma técnica flexível de busca de ocorrência de caracteres em texto, onde o termo de busca é descrito por uma linguagem formal. Por exemplo, para a variável `sol_minha`, definimos a expressão regular:

```
sol\s*[da|de|di|do|du]?\s*manh.
```

Essa expressão permite-nos considerar as ocorrências textuais "sol da manhã", "sol do manha", e outras com caracterestes invisíveis como espaço, mudança de linha, retorno de carro, entre os termos obrigatórios "sol" e "manh". Decidimos buscar por "manh" ao invés de "manha" com alternativa a possíveis erros de remoção de acentos.

Algumas variáveis, como `estacionamento_visitantes`, tiveram sua expressão regular preparadas para lidar com sinônimos. Tanto podemos encontrar o termo "estacionamento de visitantes", como "vaga para visitantes". Essas preocupações levaram-nos a definir a seguinte expressão regular:

```
(estacionamento|vaga(s))?\s*(d[aeiou])?\s*visitante
```

Durante a análise dos n-gramas de maior repetição e suas ocorrências nas descrições originais, deparamo-nos com a oportunidade de construir variáveis numéricas, processando o texto com expressões regulares preparadas para extrair o valor numérico das seguintes variáveis:

`andar` Numeral ordinal do andar do imóvel relativo ao solo;

`ano` Ano de construção do condomínio;

`andares` Quantidade de andares do condomínio;

`unidades_andar` Quantidade de unidades (imóveis) por andar no condomínio.

A construção dessas variáveis estruturais, em complemento às identificadas na seção 3.1.1 Variáveis estruturais básicas, p. 35, foi obtida utilizando-se a ferramenta de codificação interativa IPython Notebook¹⁰. Nessa ferramenta, podemos construir o código Python, executá-lo, e visualizar o resultado no próprio ambiente, o que acelera o processo iterativo de evolução do código até atingir o resultado esperado.

A análise das descrições textuais do imóveis para a geração de *tokens* e *n-gramas*, p. 38, e exibição das palavras vizinhas no contexto original, p. 39, foi possível com o uso da biblioteca Python NLTK¹¹.

A busca das ocorrências das características por expressão regular utilizou a biblioteca nativa do Python `re`¹², em conjunto com a biblioteca Pandas¹³.

O código completo, com a utilização das ferramentas mencionadas, pode ser consultado no appendix D - Listagem do IPython Notebook Variaveis_estruturais.ipynb, p. 126.

Desta forma, concluímos a construção das variáveis estruturais dos imóveis, obtidas a partir dos seus respectivos anúncios. Nas próximas seções, abordaremos a construção de outras variáveis derivadas a partir de algumas das variáveis estruturais.

3.1.3 Variáveis de acessibilidade

Uma das formas de tentar mitigar os efeitos da autocorrelação espacial da variável *preco* é a construção de variáveis de acessibilidade, que representam distâncias do imóvel a pontos de interesse como centro de trabalho, itens de lazer ou de influência para o

¹⁰IPython Notebook: <http://ipython.org/notebook.html>

¹¹Biblioteca NLTK: <http://nltk.org>

¹²Expressões regulares em Python: <https://docs.python.org/2/library/re.html>.

¹³Biblioteca Pandas: <http://pandas.pydata.org/>.

contexto analisado. A localização dos imóveis pode ser vista na p. 100. Nesse estudo buscamos criar variáveis de distâncias dos imóveis aos seguintes pontos de interesse:

1. Centro da cidade do Rio de Janeiro: ver descrição na p. 48;
2. Delegacias da Polícia Civil: mapa das localizações na p. 101;
3. Estações de metrô: mapa das estações na p. 102;
4. Favelas: mapa das favelas na p. 105;
5. Lagoas: mapa das localizações em p. 104;
6. Praias: mapa das praias na p. 105;
7. Principais logradouros: geometrias descritas na p. 107;
8. Unidades de saúde privada: mapa das unidades na p. 108;
9. Unidades de saúde pública: mapa dos estabelecimentos na p. 109;
10. Unidades do Corpo de Bombeiros: mapa das localizações na p. 110.

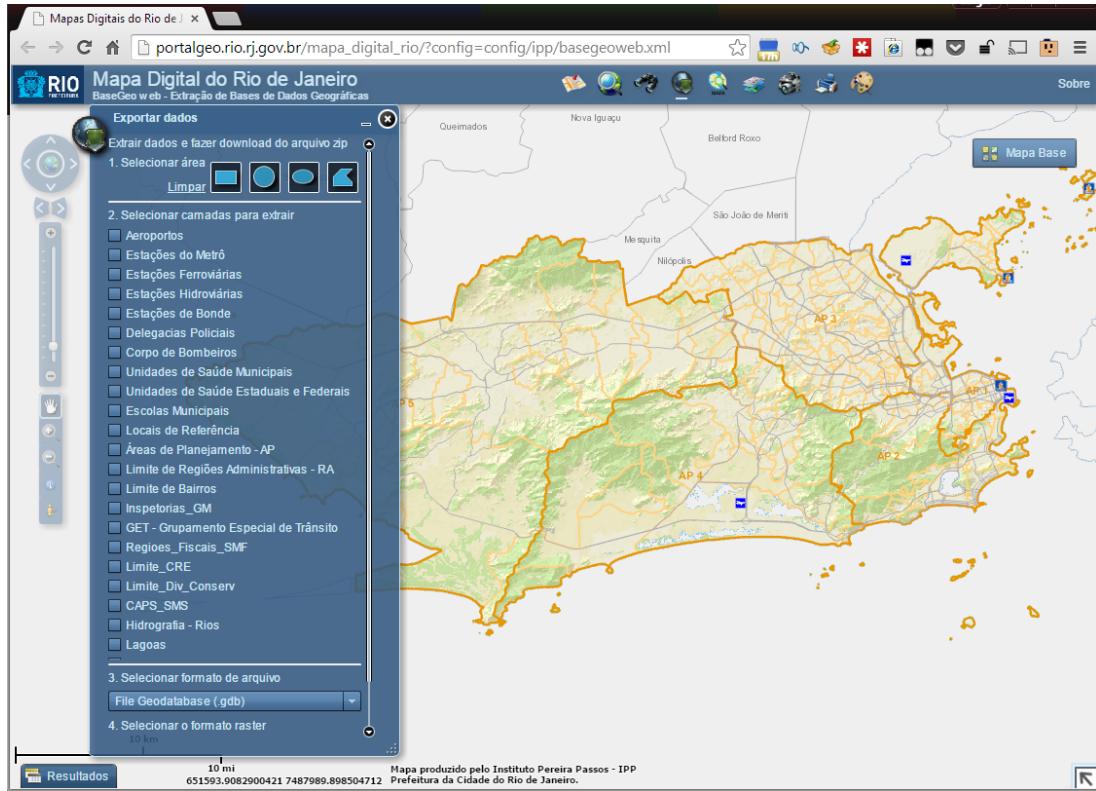
As seções a seguir detalham a coleta de cada uma dessas informações.

Informações obtidas do Mapa Digital do Rio de Janeiro

A localização de favelas, delegacias de Polícia Civil, unidades do Corpo de Bombeiros, principais logradouros, estações de trem e metrô foram obtidas da ferramenta Mapa Digital do Rio de Janeiro¹⁴, mantido pelo Instituto Municipal de Urbanismo Pereira Passos, órgão da Prefeitura Municipal do Rio de Janeiro responsável pela produção, armazenamento e divulgação de dados estatísticos sobre o município.

¹⁴Mapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml

Figura 3.4: Tela Inicial do Mapa Digital do Rio de Janeiro.



O Mapa Digital do Rio de Janeiro é uma aplicação acessível pela Internet, desenvolvida na tecnologia ArcGIS da empresa ESRI¹⁵, e permite ao usuário selecionar diversas camadas de informações georreferenciadas para toda a cidade ou uma área delimitada pelo usuário. Após a seleção das camadas de interesse é feito o *download* das informações em arquivo no formato *shapefile*, um formato proprietário da ESRI para ser utilizado em suas ferramentas, entretanto bastante popular na comunidade de Sistemas de Informações Georreferenciadas. O formato shapefile oferecido pelo Mapa Digital compõe-se de 4 arquivos distintos identificados por extensões que compartilham o mesmo nome:

¹⁵ESRI: <http://www.esri.com>

arquivo .shp contém as coordenadas dos pontos, linhas e polígonos, comumente chamados de geometrias, dos elementos georreferenciados a serem projetados em uma visualização, impressão, ou utilizado em análises geoespaciais;

arquivo .shx armazena índices para otimizar operações de acesso e leitura das geometrias no arquivo .shp;

arquivo .dbf contém os atributos associados ao elemento georreferenciado;

arquivo .sbn armazena índices espaciais para otimizar operações de análises georreferenciadas.

Importamos os arquivos shapefile para o banco de dados PostgreSQL utilizado no estudo com o seguinte comando no Windows:

Listing 3.5: Importar *shapefiles* para o banco de dados PostGIS.

```
1 for %%f in (*.shp) do
2     shp2pgsql -d -W LATIN1 -I -s 29193:4326 %%f pgr_%%~nf > pgr_%%~nf.
3         sql;
4 for %%f in (*.sql) do
5     psql -d postgresql://postgres:1234@localhost/zap -f %%f;
```

O primeiro laço seleciona os arquivos de extensão **.shp** localizados na pasta em uso e cria instruções em SQL para criação das tabelas e cadastro dos atributos nos arquivos com extensão **.sql**. As opções utilizadas nos comandos acima são:

-d determina sobreescrever a tabela caso ela já exista, o nome da tabela é o nome do arquivo **.shp**;

-W código UNICODE do texto dos atributos no arquivo **.dbf**, em nosso caso **LATIN1**;

-I cria índice espacial para otimizar análises geográficas;

- s transformar SRID, Identificador do Sistema de Referência Espacial¹⁶, em nosso caso do SRID 29193, denominado SAD69¹⁷, utilizado pelo Mapa Digital, para o SRID 4326, denominado WGS84¹⁸, escolhido como padrão para esse estudo.

O segundo laço executa os comandos nos arquivos .sql da pasta em uso no banco de dados definido pela opção -d.

Decidimos registrar apenas a menor distância para cada imóvel a cada um desses elementos com objetivo de minimizar a quantidade de informações a serem utilizadas no estudo. Tomando como exemplo Corpo de Bombeiros, após a importação *shapefile* para a tabela "pgr_corpo_de_bombeiros", o seguinte comando SQL cria uma visão materializada, uma espécie de consulta de banco de dados que fica armazenada em disco para otimizar acessos futuros, associando cada imóvel ao corpo de bombeiro mais próximo e a distância entre eles:

Listing 3.6: Exemplo de cálculo de menor distância por SQL.

```

1 CREATE MATERIALIZED VIEW vw_dist_bombeiro AS
2 SELECT DISTINCT ON (s.id)    s.id, h.gid, h.nome, ST_Distance(s.geom,
3   h.geom, true)
4 FROM vw_imovel s
5 JOIN pgr_corpo_de_bombeiros h ON ST_DWithin(s.geom, h.geom, 9999)
6 ORDER BY 1, 4

```

Estabelecimentos de Saúde

As localizações dos estabelecimentos de saúde foram coletadas do Portal de Dados Abertos da Prefeitura do Rio de Janeiro¹⁹. Esse portal disponibiliza várias informações de interesse público a respeito da cidade do Rio de Janeiro em formato tabular CSV²⁰ já com o SRID 4326.

¹⁶Do termo em inglês *Spatial Reference System Identifier*, tradução nossa. Fonte: <http://www.gaia-gis.it/gaia-sins/spatialite-cookbook/html/srid.html>.

¹⁷SAD69: <http://spatialreference.org/ref/epsg/29193/>.

¹⁸WGS84: <http://spatialreference.org/ref/epsg/4326/>.

¹⁹Portal de Dados Abertos da Prefeitura do Rio de Janeiro: <http://data.rio.rj.gov.br/>

²⁰Comma-separated values: http://en.wikipedia.org/wiki/Comma-separated_values

As informações dos estabelecimentos de saúde foram importados para o banco de dados seguindo as instruções em Python abaixo:

```

1 import zap_util as z
2 # Importa arquivo CSV para um dataframe.
3 df = z.pd.read_csv('../gis/data.rio/Estabelecimentos_de_Saude_-_
4 Dados.csv', encoding='iso-8859-1', dtype=str)
5
6 # Remove acentos dos nomes das colunas e todos os valores.
7 df.columns = z.remove_acento(list(df.columns.values))
8 df.applymap(z.remove_acento)
9
10 # Salva dataframe no banco de dados.
11 z.d.salva_dataframe(df, '_estabelecimento_saude', index=False)
12
13 # Cria campo para armazenar a coordenada geografica nativamente.
14 z.d.__executar('ALTER TABLE estabelecimento_saude ADD COLUMN geom_
15 geometry(Point,4326);')
16
17 #Cria a geometria com base na Latitude e Longitude.
18 z.d.__executar('update estabelecimento_saude set geom = ' + \
19 'stGeomFromText(''POINT('' || "Longitude" || '' || "Latitude" || '' || "Latitude" || '')'', 4326)')
20
21 # Cria chave primaria para a tabela.
22 z.d.__executar('ALTER TABLE estabelecimento_saude' +
23 ' ADD CONSTRAINT pk_estabelecimento_saude PRIMARY KEY ("CNES");')

```

O módulo "zap_util" importado na linha 1 contém diversas funções criadas para auxiliar a construção desse estudo e encontra-se listado no appendix C - Listagem do módulo [zap_util.py](#), p. 116. Separamos os estabelecimentos de saúde em dois conjuntos,

um de administração pública e um de administração privada, e calculamos a distância dos imóveis a cada um destes conjuntos a partir da geometria do tipo ponto criada com as informações de latitude e longitude. Os demais dados presentes no arquivo não foram utilizados.

Praias

A geometria das praias que circundam a cidade do Rio de Janeiro foram capturadas do serviço Open Street Map²¹, um serviço colaborativo de registro e consulta de informações georreferenciadas, similar ao Google Maps, mas cuja licença de uso *Open Database License* garante acesso gratuito para compartilhar, modificar e usar os dados disponíveis livremente.

A obtenção dos dados foi facilitada como uso do *plugin* QuickOSM para o software Quantum GIS²². O Quantum GIS é uma ferramenta *open source* de Sistemas de Informações Georreferenciadas, que permite a visualização, edição, análises e publicação de informações georreferenciadas a partir de diversas fontes como bancos de dados, arquivos texto ou serviços de dados. Suas funcionalidades são extensíveis por *plugins*, pequenos componentes de software que adicionam funcionalidades extras ao software original. O *plugin* em questão, QuickOSM, permite fazer o *download* das informações disponíveis na base de dados dos Open Street Map que intersectem uma área definida por retângulo de coordenadas personalizadas. Em nosso caso, utilizamos o Quantum GIS para determinar as coordenadas de um retângulo circunscrito à cidade do Rio de Janeiro. Essas coordenadas são passadas plugin QuickOSM, em conjunto com um filtro para obter somente dados sobre praias. O *plugin* faz o download das informações pertinentes em um arquivo temporário e apresenta o resultado visualmente como uma camada na visualização principal do Quantum GIS. Como o retângulo circunscrito à cidade intersecta outras cidades vizinhas, filtramos as praias de interesse realizando

²¹Open Street Map: <https://www.openstreetmap.org>.

²²Software Quantum GIS: <http://www.qgis.org>.

nova análise espacial onde ordenamos ao Quantum GIS a seleção de geometrias das praias contidas dentro da área definida pela geometria da cidade do Rio de Janeiro obtida do Mapa Digital do Rio de Janeiro mencionado acima. Por fim, essa seleção é salva no formato *shapefile* e importadas para o banco de dados conforme descrito na listagem 3.5. Por fim, o cálculo da menor distância é realizado de forma similar ao exemplo apresentado na listagem 3.6.

Centro da cidade do Rio de Janeiro

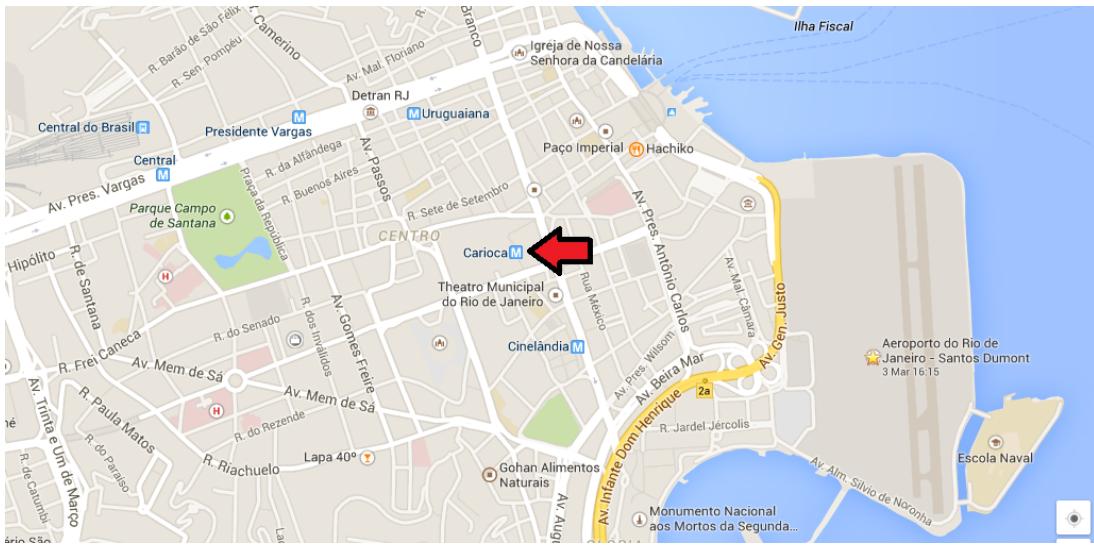
Definimos arbitrariamente o centro da cidade como a localização geográfica da estação de metrô Carioca, bairro Centro, de coordenadas latitude -22.907670, longitude -43.177917, em função de sua localização central à Avenida Rio Branco, avenida que agrupa um grande número de escritórios, centros médicos e repartições públicas ao longo de seu entorno, fig. 3.5. Construímos 3 distâncias a partir deste ponto para cada imóvel: uma distância euclidiana até a localização definida, a distância até a localização definida somente na dimensão da latitude e a última somente na dimensão da longitude. Com essas duas últimas esperamos poder verificar a contribuição da distância em relação aos eixos Norte-Sul e Leste-Oeste para os preços dos imóveis.

3.1.4 Variáveis socioambientais

A maioria das variáveis socioambientais foram obtidas a partir do Armazém de Dados da Prefeitura do Rio de Janeiro ²³, do Instituto Pereira Passos, que são uma síntese sobre os bairros da cidade do Rio de Janeiro a partir dos dados coletados nos Censos de 2000 e 2010 pelo IBGE. Neste serviço as informações são organizadas hierarquicamente em nível de Áreas de Planejamento, Regiões Administrativas e bairros. Regiões Administrativas, também conhecidas como subprefeituras, são formadas por um conjunto de bairros vizinhos para gestão dos serviços públicos específico a esses bairros. As Áreas de

²³Armazém de Dados: <http://www.armazemdedados.rio.rj.gov.br/>

Figura 3.5: Localização da Estação de Metrô Carioca, definido como o centro da cidade.



Planejamento são agregações de Regiões Administrativas, prevalecidas pela distribuição geográfica na cidade. A fig. A.1, p. 98, apresenta a divisão administrativa da cidade ao nível de bairro, Regiões Administrativas e Áreas de Planejamento.

Desta forma, o menor nível de agregação dos dados socioambientais que encontramos para a cidade foram por bairros. Abaixo listamos as informações selecionadas, ano em que a informação foi gerada e número da tabela no Armazém de Dados para referência:

Índice de Desenvolvimento Humano Índice de Desenvolvimento Humano Municipal, ano 2000, tabela nº 1172;

Média de anos de estudo Média de anos de estudo das pessoas responsáveis pelos domicílios particulares permanentes, ano 2000, tabela nº 488;

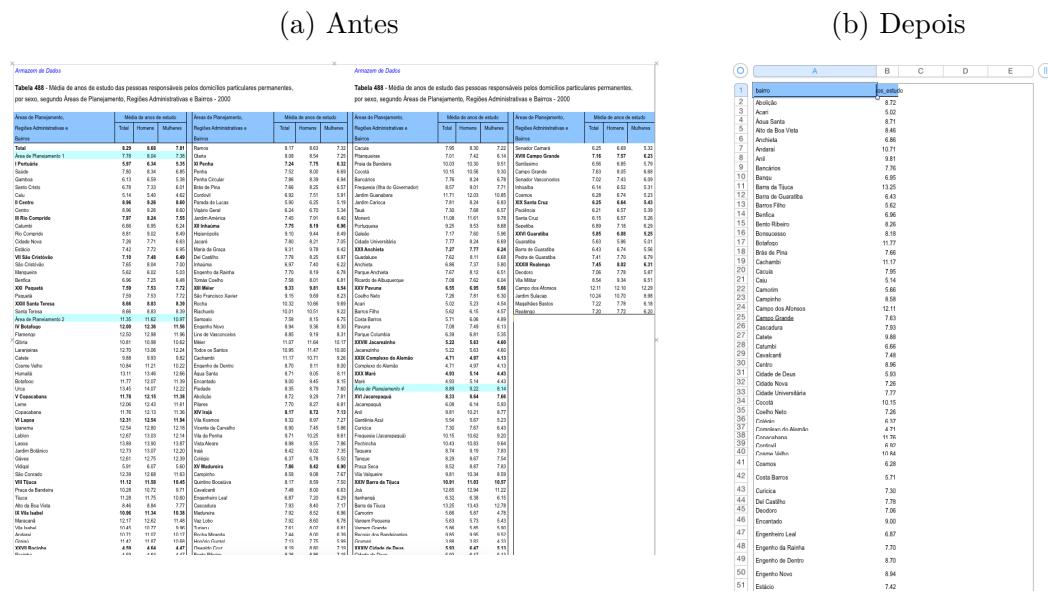
Percentual de alfabetização Percentual de alfabetização de pessoas de 10 anos ou mais de idade, ano 2010, tabela nº 3132 ;

Rendimento nominal médio Rendimento nominal médio de pessoas de 10 anos ou mais de idade, ano 2010, tabela nº 3151;

Saneamento Percentual de domicílios com banheiros exclusivos ligados à rede geral de esgoto, ano 2010, tabela nº 3169:

Essas informações estão disponíveis como planilhas eletrônicas no formato Microsoft Excel. Para a importação dos dados, primeiro editamos as planilhas individualmente, removendo informações não relevantes ao interesse do estudo, e reorganizando a listagem dos dados originalmente distribuídos em várias colunas, para uma única.

Figura 3.6: Preparação das planilhas do Armazém de Dados para importação.



A seguir, a importação foi feita utilizando a biblioteca Python XLRD²⁴ para a leitura das planilhas no formato Microsoft Excel e armazenamento em memória para a etapa posterior de tratamento, descrito na página 57.

Já os dados sobre criminalidade foram recuperados dos relatórios eletrônicos Resumo das Principais Incidências Criminais, divulgadas mensalmente pelo Instituto de Segurança Pública do Estado do Rio de Janeiro, ISP²⁵. Esses relatórios apresentam um total de 39

²⁴Biblioteca XLRD: <https://pypi.python.org/pypi/xlrd>

²⁵Instituto de Segurança Pública: <http://www.isp.rj.gov.br/dadosoficiais.asp>

categorias de incidências criminais comunicadas na forma de Registros de Ocorrência ou Registros de Aditamento lavrados nas delegacias de Polícia Civil em todo o estado do Rio de Janeiro.

A partir dessa fonte, coletamos os dados de criminalidade divulgados entre outubro de 2013 a outubro de 2014, de forma a cobrir todos os meses de anúncios publicados em nosso conjunto de imóveis. Escolhemos utilizar as taxas por 100 mil habitantes a fim de permitir a comparação entre bairros conforme respectiva população. Dentre as 39 categorias disponíveis, resolvemos construir duas variáveis:

Crimes violentos somatório das taxas de homicídio doloso, lesão corporal seguida de morte, latrocínio, tentativa de homicídio, lesão corporal dolosa e estupro;

Crimes de roubo somatório das taxas de roubo a estabelecimento comercial, a residência, a veículo, de carga, de transeunte, em coletivo, de banco, de caixa eletrônico, de aparelho celular, com condução da vítima para saque e furto de veículos

Tais relatórios segregam os dados em Áreas Integradas de Segurança Pública, AISPs, onde cada delegacia é responsável por uma dessas áreas, cujos limites podem conter mais de um bairro, e ocorre de um mesmo bairro ser dividido 2 AISPs, como é o caso dos bairros Centro, Colégio e Tijuca. Para o caso da cidade do Rio de Janeiro, os bairros dessa são agrupados em 17 AISPs, conforme tabela [3.2](#).

Tabela 3.2: Lista de bairros por AISP.

AISP	Bairros
2	Botafogo, Catete, Cosme Velho, Flamengo, Glória, Humaitá, Laranjeiras, Urca
3	Abolição, Cachambi, Del Castilho, Encantado, Engenho Novo, Engenho da Rainha, Engenho de Dentro, Inhaúma, Jacarezinho, Jacaré, Lins de Vasconcelos, Maria da Graça, Méier, Piedade, Pilares, Riachuelo, Rocha, Sampaio, São Francisco Xavier, Todos os Santos, Tomás Coelho, Água Santa
4	Caju, Catumbi, Centro, Cidade Nova, Estácio, Mangueira, Maracanã, Praça da Bandeira, Rio Comprido, São Cristóvão, Tijuca, Vasco da Gama
5	Centro, Gamboa, Lapa, Paqueta, Santa Teresa, Santo Cristo, Saúde
6	Alto da Boa Vista, Andaraí, Grajaú, Tijuca, Vila Isabel
9	Bento Ribeiro, Campinho, Cascadura, Cavalcanti, Coelho Neto, Colégio, Engenheiro Leal, Honório Gurgel, Madureira, Marechal Hermes, Osvaldo Cruz, Praça Seca, Quintino Bocaiúva, Rocha Miranda, Turiaçu, Vaz Lobo, Vila Valqueire
14	Bangu, Campo dos Afonsos, Deodoro, Gericinó, Jardim Sulacap, Magalhães Bastos, Padre Miguel, Realengo, Senador Camará, Vila Militar
16	Brás de Pina, Complexo do Alemao, Cordovil, Jardim América, Olaria, Parada de Lucas, Penha, Penha Circular, Vigário Geral
17	Bancários, Cacuia, Cidade Universitária, Cocotá, Freguesia <u>Ilha</u> , Galeão, Jardim Carioca, Jardim Guanabara, Moneró, Pitangueiras, Portuguesa, Praia da Bandeira, Ribeira, Tauá, Zumbi
18	Anil, Cidade de Deus, Curicica, Freguesia <u>Jacarepaguá</u> , Gardênia Azul, Jacarepaguá, Pechincha, Tanque, Taquara
19	Copacabana, Leme
22	Benfica, Bonsucesso, Higienópolis, Manguinhos, Maré, Ramos
23	Gávea, Ipanema, Jardim Botânico, Lagoa, Leblon, Rocinha, São Conrado, Vidigal
27	Guaratiba, Paciência, Pedra de Guaratiba, Santa Cruz, Sepetiba
31	Barra da Tijuca, Barra de Guaratiba, Camorim, Grumari, Itanhangá, Joá, Recreio dos Bandeirantes, Vargem Grande, Vargem Pequena
40	Campo Grande, Cosmos, Inhoaíba, Santíssimo, Senador Vasconcelos
41	Acari, Anchieta, Barros Filho, Colégio, Costa Barros, Guadalupe, Irajá, Parque Anchieta, Parque Colúmbia, Pavuna, Ricardo de Albuquerque, Vicente de Carvalho, Vila Kosmos, Vila da Penha, Vista Alegre

Desta forma, os imóveis de um bairro possuem o mesmo valor para cada uma das duas variáveis de criminalidade. Os bairros Centro, Colégio e Tijuca são atendidos em mais de um AISPs. Nesses casos definimos o valor das variáveis de criminalidade como a média dos AISPs correspondentes.

A implementação da importação e o tratamento para os dados ausentes das variáveis socioambientais está detalhado no apêndice [Listagem do IPython Notebook Variaveis_socioambientais.ipynb](#), p. 143.

3.2 Tratamento dos dados

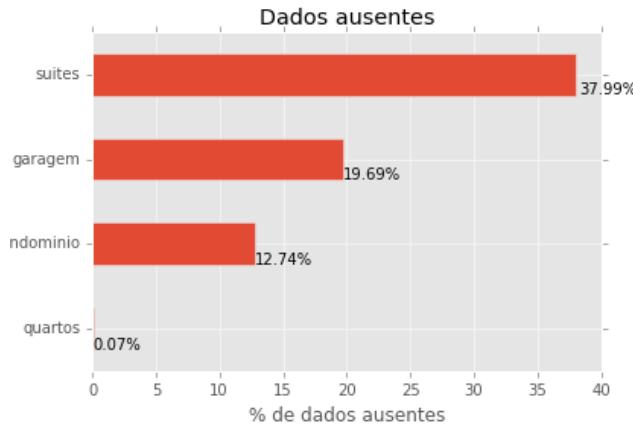
O tratamento dos dados é o passo necessário para a garantir da qualidade dos mesmos de forma que o modelo a ser proposto possa ser construído dentro das condições ideais, evitando perturbações não conhecidas por erros, inconsistência ou presença de valores muito além do limite esperado.

3.2.1 Variáveis estruturais

Obtivemos um total de 91.091 anúncios de vendas de imóveis do tipo apartamento padrão, com a condição de usado, para a cidade no ZAP Imóveis cujos anúncios foram publicados entre outubro de 2013 a outubro de 2014. Desses, 59.760 possuem coordenadas geográficas que utilizamos para determinar quais pertencem de fato à cidade do Rio de Janeiro, cujo resultado após essa análise são 59.412 imóveis. Em seguida verificamos cada uma das variáveis estruturais básicas, mencionadas na página [34](#). A fig. [3.7](#) apresenta o percentual de observações ausentes a cada uma dessas variáveis.

Como tratamento, excluímos as 43 observações cuja variável *quartos* estava ausente, resultando em 59.369 observações. Já as outras variáveis *suítes*, *garagem* e *condomínio*, assumimos que o percentual representa adequadamente a ausência da característica em questão nos anúncios. Sendo assim, definimos os valores ausentes como 0.

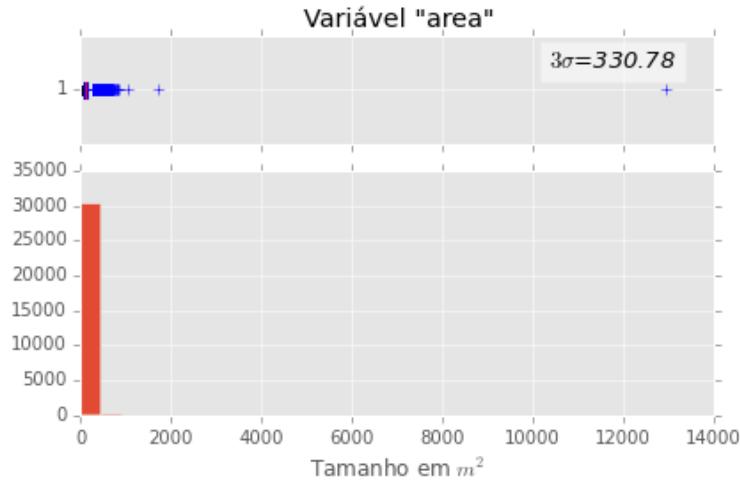
Figura 3.7: Percentual de valores ausentes.



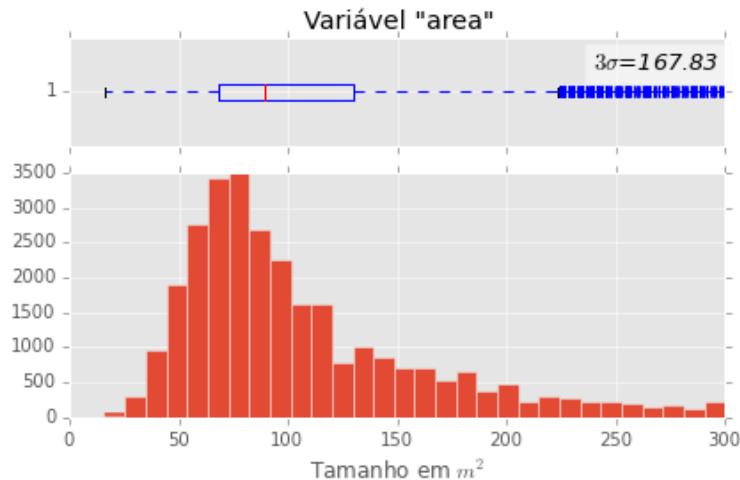
Em seguida passamos a avaliar a presença de *outliers* na distribuição das variáveis estruturais. Para tal, construímos uma visualização específica com um boxplot horizontal e um histograma, de forma que é possível avaliar a distribuição da variável em ambos os gráficos ao longo de seus valores. O gráfico boxplot conta ainda com um cálculo do valor de 3 desvios padrão como referência. A fig. 3.8 apresenta como exemplo da visualização a variável ***area*** antes e depois da remoção dos outliers. Ao analisar todas as variáveis, identificamos um total de 9.369 observações com outliers que após serem removidos nosso conjunto de dados resultou em 50.000 observações. Demais visualizações para as outras variáveis, bem como o detalhamento passo a passo dos critérios utilizados para a identificação dos outliers estão apresentados no appendix F, [Listagem do IPython Notebook Tratar_variaveis_estruturais.ipynb](#), p. 152.

Figura 3.8: Exemplo de visualização utilizado para detecção de outliers.

(a) Com outliers.



(b) Sem outliers.

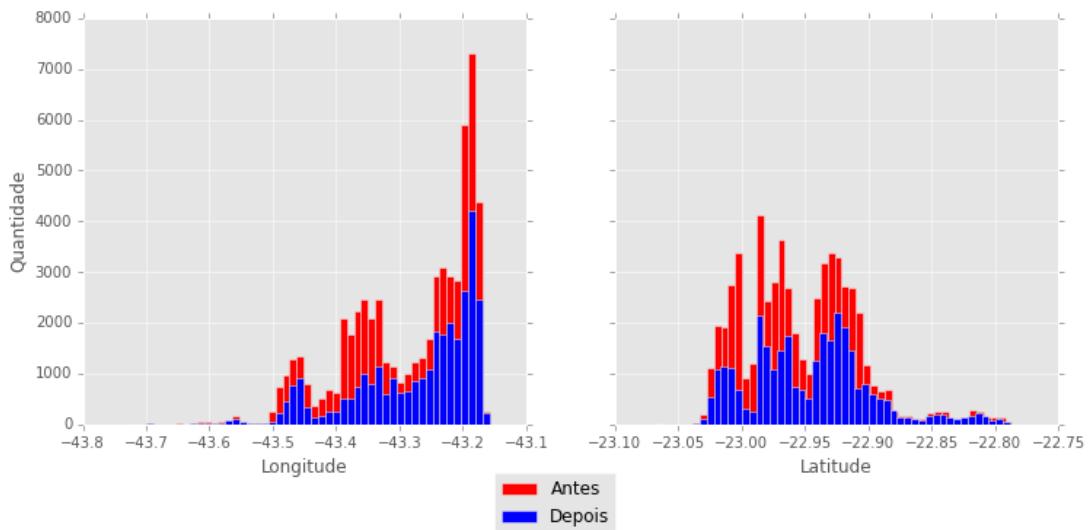


3.2.2 Variáveis de acessibilidade

As variáveis de interesse foram calculadas com base nas coordenadas de latitude e longitude aos respectivos imóveis. Identificamos em nosso conjunto de dados que muitos imóveis compartilhavam as mesmas coordenadas, mesmo localizados em ruas diferentes,

o que caracteriza um problema no cadastro. Optamos por tomar uma atitude conservadora e removemos do conjunto de observações todos os imóveis com coordenadas repetidas²⁶, levando nosso conjunto de dados ao número final de 28.819 observações. A fig. 3.9 apresenta distribuição das coordenadas longitude e latitude antes e depois da remoção dos imóveis com coordenadas repetidas.

Figura 3.9: Distribuição de antes e depois da remoção dos imóveis com coordenadas repetidas.



Dados que as variáveis de acessibilidade apresentadas na seção 3.1.3, p. 42, foram calculadas como a menor distância aos imóveis por um método determinístico, e os imóveis e pontos de interesse de acessibilidade foram verificados pertinentes ao limite da cidade do Rio de Janeiro, não esperamos obter valores que sejam considerados *outliers*. Entretanto, a fim de garantir a qualidade dos dados, apresentamos as visualizações que combinam boxplot com histograma para cada uma delas, listados no appendix G, Verificação das variáveis de acessibilidade, p. 168.

²⁶Um método alternativo é apesentado como uma extensão ao trabalho, no capítulo Conclusão, p. 93.

3.2.3 Variáveis socioambientais

Como o nível mais detalhado dos dados no Armazém de Dados é o de bairro, as variáveis *Média de Anos de Estudo*, *Percentual Alfabetização*, *Rendimento Nominal Médio*, *Saneamento* e *IDH* foram replicadas para os imóveis do respectivo bairro. Os valores ausentes de bairros para uma variável foram substituídos pela média aritmética dos bairros vizinhos conforme descrito abaixo:

Gericinó As variáveis *Média de Anos de Estudo* e *IDH* foram substituídas pela média aritmética dos bairros Bangu, Padre Miguel e Realengo, respectivamente.

Lapa As variáveis *Média de Anos de Estudo*, *Percentual de Alfabetização*, *Rendimento Nominal* e *IDH* foram substituídas pela média aritmética dos bairros Centro, Santa Teresa e Glória, respectivamente.

Vasco da Gama As variável *Média de Anos de Estudo* foi substituída pela média aritmética dos bairros São Cristóvão, Caju e Benfica.

As variáveis relacionadas a criminalidade, *Crimes de Roubo* e *Crimes Violentos*, não apresentaram valores ausentes para os bairros dos imóveis. De igual forma as demais, as visualizações das variáveis socioambientais encontram-se no appendix H - Verificação das variáveis socioambientais, p. 174.

3.3 Descrição das variáveis

Nesta seção, resumimos todas as variáveis obtidas e construídas neste capítulo a serem consideradas para o modelo de regressão da análise. Organizamos a descrição em uma tabela onde as variáveis são apresentadas em linhas e a coluna "Nome"exibe o nome da variável utilizado no modelo computacional que restringe o uso de caracteres de acentuação e espaços. A coluna "Característica"apresenta uma descrição textual da

característica do imóvel associada à variável. "Tipo" informa se é dicotômica, numérica inteira ou real. Finalmente, a coluna "Classificação" apresenta a classificação da variável baseada em (Long et al., 2007, p. 3) e utilizada ao longo do estudo .

Tabela 3.3: Descrição das variáveis.

Variável	Característica	Tipo	Classificação
preco	Preço anunciado do imóvel, em R\$ 1.000,00	Real	Var. dependente
andares	Quantidade de andares no condomínio	Inteiro	Estrutural
ano	Ano de construção do imóvel	Inteiro	Estrutural
area	Área do imóvel em m^2	Real	Estrutural
m2	Preço por m^2 , em R\$ 1.000,00/ m^2	Real	Estrutural
quartos	Quantidade de dormitórios	Inteiro	Estrutural
suites	Quantidade de suítes	Inteiro	Estrutural
vagas	Quantidade de vagas de garagem	Inteiro	Estrutural
unidades_andar	Quantidade de unidades por andar	Inteiro	Estrutural
lat	Latitude anunciada	Real	Acessibilidade
lng	Longitude anunciada	Real	Acessibilidade
dist_bombeiro	Distância ao Corpo de Bombeiros	Real	Acessibilidade
dist_centro	Distância ao Centro da Cidade	Real	Acessibilidade
dist_centro_lat	Distância latitudinal ao Centro da Cidade	Real	Acessibilidade
dist_centro_lng	Distância longitudinal ao Centro da Cidade	Real	Acessibilidade
dist_delegacia	Distância à delegacia de Polícia Civil	Real	Acessibilidade
dist_favela	Distância à favela	Real	Acessibilidade
dist_lagoa	Distância à lagoa	Real	Acessibilidade
dist_logradouro	Distância a um Principal Logradouro	Real	Acessibilidade
dist_metro	Distância à estação de metrô	Real	Acessibilidade
dist_praia	Distância à praia	Real	Acessibilidade
dist_saude_privada	Distância a um estabelecimento de saúde privado	Real	Acessibilidade
dist_saude_publica	Distância a um estabelecimento de saúde público	Real	Acessibilidade
dist_trem	Distância à estação de trem	Real	Acessibilidade
dm_andar_alto	Determina se localiza-se em andar alto	Dicotômica	Estrutural
dm_andar_baixo	Determina se localiza-se em andar baixo	Dicotômica	Estrutural
dm_andar_inteiro	Determina se ocupa o andar inteiro	Dicotômica	Estrutural
dm_armario	Presença de armário embutido	Dicotômica	Estrutural
dm_banheira	Presença de banheira ou hidromassagem	Dicotômica	Estrutural
dm_blindex	Presença de blox blindex ou similar	Dicotômica	Estrutural
dm_churrasqueira	Presença de churrasqueira	Dicotômica	Estrutural
dm_closet	Presença de closet	Dicotômica	Estrutural
dm_cobertura	Determina se é uma cobertura	Dicotômica	Estrutural
dm_copa	Presença de copa	Dicotômica	Estrutural
dm_creche	Presença de creche, ou bayb care, no condomínio	Dicotômica	Estrutural
dm_dep_empregada	Presença de dependências de empregada	Dicotômica	Estrutural
dm_duplex	Determina se é de dois andares	Dicotômica	Estrutural
dm_elevador	Presença de elevador no condomínio	Dicotômica	Estrutural
dm_elevador_privado	Presença de elevador privado	Dicotômica	Estrutural
dm_escritura	Determina se a vaga de garagem está na escritura	Dicotômica	Estrutural
dm_esquina	Determina se localiza em uma esquina	Dicotômica	Estrutural

Continua na próxima página

Tabela 3.3 –continuação da página anterior

Variável	Característica	Tipo	Classificação
dm_est_visitantes	Presença de estacionamento para visitantes	Dicotômica	Estrutural
dm_frente	Determina se é a frente do condomínio	Dicotômica	Estrutural
dm_fundos	Determina se está nos fundos do condomínio	Dicotômica	Estrutural
dm_granito	Presença de benfeitoria em granito	Dicotômica	Estrutural
dm_hidrometro	Determina se conta com hidrômetro individual	Dicotômica	Estrutural
dm_indevassavel	Determina se é indevassável	Dicotômica	Estrutural
dm_lateral	Determina localização na lateral do condomínio	Dicotômica	Estrutural
dm_linear	Determina se é linear	Dicotômica	Estrutural
dm_mezanino	Presença de mezanino	Dicotômica	Estrutural
dm_original	Determina se não sofreu reformas	Dicotômica	Estrutural
dm_piscina	Presença de piscina, próprio ou no condomínio	Dicotômica	Estrutural
dm_planejad	Presença de móveis planejados	Dicotômica	Estrutural
dm_play	Presença de playground	Dicotômica	Estrutural
dm_porcelanato	Presença de alguma benfeitoria em granito	Dicotômica	Estrutural
dm_portaria	Presença de portaria	Dicotômica	Estrutural
dm_recuado	Determina se é recuado em relação à rua	Dicotômica	Estrutural
dm_sala_jantar	Presença de sala de jantar	Dicotômica	Estrutural
dm_salao_festas	Presença de salão de festas no condomínio	Dicotômica	Estrutural
dm_salao_jogos	Presença de salão de jogos no condomínio	Dicotômica	Estrutural
dm_sauna	Presença de sauna, próprio ou no condomínio	Dicotômica	Estrutural
dm_segurança	Presença de segurança particular no condomínio	Dicotômica	Estrutural
dm_sol_manha	Determina se recebe sol apenas pela manhã	Dicotômica	Estrutural
dm_sol_tarde	Determina se recebe sol apenas à tarde	Dicotômica	Estrutural
dm_terraco	Presença de terraço	Dicotômica	Estrutural
dm_triplex	Determina se é triplex	Dicotômica	Estrutural
dm_varanda	Presença de varanda	Dicotômica	Estrutural
dm_vista_mar	Determina se há vista para o mar	Dicotômica	Estrutural
se_anos_estudo	Média de anos de estudo	Real	Socioambiental
se_crm_roubo	Roubos por 100 mil habitantes	Real	Socioambiental
se_crm_violento	Homicídios por 100 mil habitantes	Real	Socioambiental
se_idh	Índice de Desenvolvimento Humano	Real	Socioambiental
se_perc_alfabetizacao	Percentual de alfabetização	Real	Socioambiental
se renda	Renda média por bairro	Real	Socioambiental
se_saneamento	Percentual de ligação à rede pública de esgoto	Real	Socioambiental

3.3.1 Variáveis *quartos, suítes e garagem* tratadas como dicotômicas

Embora as variáveis *quartos, suítes e garagem* sejam numéricas inteiras, seu contexto de aplicação é a contagem de quantidade dos respectivos itens. Dessa forma recomenda-se que tais variáveis sejam tratadas como dicotômicas, para que os respectivos coeficientes da regressão indiquem individualmente o valor associado a quantidade presente de quartos, suítes e vagas de garagem, ao invés de um valor único correspondente ao

incremento unitário da variável (Andersen e Skovgaard, 2010, p. 137).

Ao transformar uma variável em dicotômica, precisamos remover uma das k classes derivadas para eliminar a colinearidade entre todo o conjunto, conforme apresentado na p. 20. Para a variável *quartos*, removemos a classe dicotômica que representa a quantidade 1, restando as variáveis para as quantidades 2, 3 e 4. Não é possível determinar se a variável removida em questão representa o quarto de casal ou quarto de solteiro. Para as variáveis *suítes* e *garagem*, removemos a classe dicotômica que representa a quantidade 0, ausência dessa característica no imóvel. Desta forma, o valor da interceptação da função de regressão \hat{y} do preço do imóvel considera que o mesmo possui 1 quarto, sem suítes e sem vagas de garagem.

3.4 Seleção do Modelo

A seleção do modelo é a etapa de busca do melhor modelo estatístico que descreva um determinado conjunto de dados com base em uma medida de performance. Em nosso caso, em que avaliamos um modelo hedônico dos preços de imóveis da cidade do Rio de Janeiro, que por definição é uma regressão linear, a seleção do modelo resume-se a determinar o conjunto de variáveis independentes apresentadas na tabela 3.3, p. 58, adequado à inferência da variável dependente *preco*, com base em uma medida de performance .

Primeiro, buscamos determinar se as variáveis independentes²⁷ não sejam uma combinação linear entre si, de forma que $X^T X$ não seja singular e os coeficientes da regressão tenham solução única (Hastie et al., 2013, p. 46). Segundo (Baltagi, 2008, p. 74), para que as variáveis independentes possam fazer parte do modelo final de regressão, basta que não haja perfeita multicolinearidade entre elas. Desta forma, tomamos as variáveis duas a duas e aplicamos o coeficiente de correlação de Pearson²⁸, onde determinamos

²⁷O termo "variáveis independentes" tem seu significado apresentado na seção Regressão Linear, p. 18.

²⁸Coeficiente de Correlação de Pearson: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

arbitrariamente que, se o valor absoluto do coeficiente for inferior a 0,8, as variáveis são consideradas linearmente independentes e serão aplicadas ao modelo de regressão. Se o valor absoluto do coeficiente de Pearson for superior ou igual a 0,8, consideramos as variáveis linearmente dependentes e excluiremos uma das duas do modelo final. O segundo passo é a remoção iterativa das variáveis cujo limite de significância estatística referente a hipótese nula de que o coeficiente resultante da regressão linear seja zero ultrapasse o limite de 0,05. Dessa forma, partimos de um modelo inicial com todas as variáveis disponíveis para um final com somente aquelas que satisfazem as condições propostas, implementando uma variação do método de *backward-stepwise selection* discutido em ([Hastie et al., 2013](#), p. 59).

Utilizamos a técnica de validação cruzada *K-Fold* apresentada na página [22](#) para avaliar a performance dos modelos propostos. Optamos por utilizar o *RMSE*, p. [24](#), em conjunto com o R^2 , como medida de performance para a avaliação de cada modelo. O modelo final é aquele com o menor *RMSE* e maior R^2 , que por sua vez é tomado em todo o conjunto de observações para construção dos coeficientes $\hat{\beta}$.

3.4.1 Seleção do modelo hedônico para toda a cidade do Rio de Janeiro

Os seguintes modelos foram avaliados para a regressão de toda a cidade do Rio de Janeiro.

Todos os dados disponíveis Modelo tradicional construído a partir de todas as variáveis disponíveis no conjunto de dados. A utilização de todas as variáveis pode implicar em problemas de multicolinearidade mas mantivemos essa escolha para termos uma referência do impacto ao retirar-se as variáveis linearmente dependentes. Identificamos esse modelo com o código "m1".

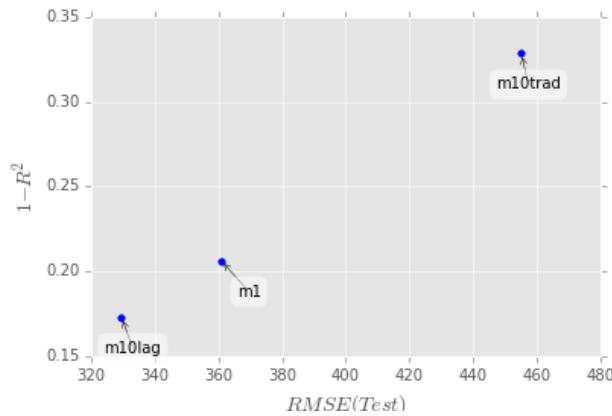
product-moment_correlation_coefficient.

Sem variáveis correlacionadas entre si Modelo tradicional sem as variáveis altamente correlacionadas entre si, aquelas com coeficiente de correlação de Pearson maior ou igual a 0,8. A decisão de qual das variáveis excluir foi empírica, buscando manter as que fossem de interesse do estudo. As variáveis removidas foram: *dist_bombeiro, dist_centro_lat, dist_centro_lng, dist_delegacia, dist_lagoa, dist_metro, dist_saude_publica, dist_trem, dm_banheira, dm_churrasqueira, dm_duplex, dm_est_visitantes, dm_play, dm_salao_festas, dm_salao_jogos, dm_sauna, se_anos_estudo, se_crm_roubo, se_crm_violento, se_perc_alfabetizacao, se_renda e se_saneamento*. Esse modelo é identificado pelo código "m10trad".

Lag espacial Mesma lista de variáveis do modelo anterior, com a adição da variável *preco_lag* que representa o preço médio dos 100 imóveis mais próximos, como forma de atenuar os efeitos da autocorrelação espacial sobre as outras variáveis independentes. Identificamos esse modelo com o código "m10lag".

Para a escolha do modelo consideramos duas medidas de performance: *RMSE* e R^2 . Plotamos esses valores obtidos de cada modelo em um gráfico, fig. 3.10, calculando-se o complemento de R^2 para facilitar identificação visual do melhor modelo, aquele mais próximo da origem do gráfico.

Figura 3.10: Comparativo entre os 3 modelos para seleção.



Pelo gráfico vemos que o modelo "m10lag" é o que melhor atende nossa definição de melhor modelo para a cidade do Rio de Janeiro. A discussão a respeito da diferença dos resultados entre esses exemplos está detalhado na seção [4.2 Modelo hedônico para a cidade do Rio de Janeiro, p. 70](#).

A implementação das regressões lineares utilizaram extensivamente as bibliotecas Pandas e Statsmodels, da linguagem Python. A primeira serviu para o armazenamento das observações. A segunda foi utilizada para a execução das regressões e visualização dos resultados. A implementação completa em IPython Notebook está disponível no apêndice [Listagem do IPython Notebook Reg_Linear_-_Modelo_Cidade.ipynb](#), p. 179.

3.4.2 Seleção do modelo hedônico para cada bairro da cidade do Rio de Janeiro

Para a seleção do modelo a ser aplicado exclusivamente a cada bairro da cidade do Rio de Janeiro, optamos por construir uma rotina que automatizasse o processo de seleção das variáveis com respeito a correlações e significância estatística, utilizando inicialmente todas as variáveis disponíveis. Para cada bairro no conjunto de dados, criamos um subconjunto com seus respectivos imóveis e primeiramente removemos as variáveis de

valor constante ao longo do subconjunto, tais como as variáveis socioambientais, que de acordo com as restrições de disponibilidade encontradas durante sua construção, apresentadas na seção [Variáveis socioambientais](#), p. 48, são replicadas aos imóveis dos respectivos bairros, dentre outras variáveis que podem ser constantes na seleção de um subconjunto por bairro. Sobre a significância estatística, para cada bairro, repetimos a regressão caso alguma variável ultrapasse o limite de significância de 0,05 até que nenhuma mais ocorra. Optamos por não remover variáveis altamente correlacionadas com outras pois a determinação de qual deve ser removida depende mais do interesse da modelagem em si do que uma avaliação numérica. Como nosso objetivo é conhecer as ocorrências das variáveis entre os modelos dos bairros, deixamos a cargo somente da significância estatística a determinação de qual deve ser o conjunto final.

O apêndice [Listagem do IPython Notebook Reg_Linear_-_Modelo_Cidade.ipynb](#), p. 179, apresenta a implementação em Python para a construção das regressões para os bairros utilizando as bibliotecas Statsmodels e Pandas da linguagem Python na ferramenta IPython Notebook.

No capítulo a seguir apresentamos os resultados obtidos pela avaliação dos diferentes modelos propostos, um para toda a cidade do Rio de Janeiro, e outros específicos a alguns bairros selecionados.

Capítulo 4

Apresentação e análise dos resultados

4.1 Análise descritiva das variáveis

A análise descritiva das variáveis nos permite conhecermos o domínio dos valores das mesmas, sua distribuição ao longo dos quartis, valores extremos, a média, desvio padrão e coeficiente de correlação com a variável preço. Tal conhecimento é fundamental para a correta interpretação dos coeficientes resultantes da regressão linear.

A tabela 4.1, p. 67, mostra uma análise descritiva das variáveis propostas na página 58 para a composição do modelo. A coluna "mean", "std", "min", "25%", "50%", "75%", "max" e "Corr(Preco)" representam respectivamente a média, desvio padrão, valor mínimo, valor do primeiro, segundo e terceiro quartil, valor máximo e correlação com a variável *preco*. Por exemplo, a distribuição da variável *dist_centro* informa que apenas 25% dos imóveis em nosso conjunto de dados estão a uma distância superior a 17,67 Km do ponto atribuído como centro da cidade, p. 48, o que mostra uma extrema concentração de imóveis nessa proximidade ao centro da cidade, dado que sua extensão longitudinal tem cerca de 65 Km. Supondo-se que o uso de classificados pagos como o ZAP Imóveis

seja utilizado por proprietários e demais agentes imobiliários para a venda de imóveis monetariamente relevantes, essa concentração da distribuição espacial já é um indício de que os preços possuem algum tipo de dependência com o centro da cidade, ou algum outro ponto de interesse próximo ao centro.

Tabela 4.1: Análise descritiva das variáveis propostas a modelagem.

	mean	std	min	25%	50%	75%	max	Corr(Preco)
area	102.34	52.08	16.00	67.00	87.00	124.00	300.00	0.69
condominio	0.55	0.42	0.00	0.26	0.49	0.80	2.00	0.64
dist_bombeiro	1.54	1.14	0.01	0.73	1.17	2.04	9.12	-0.20
dist_centro	12.48	9.09	0.10	6.28	8.96	17.67	55.87	-0.15
dist_centro_lat	5.33	3.81	0.00	1.93	4.65	8.25	17.64	0.38
dist_centro_lng	10.02	9.76	0.00	1.64	6.64	16.27	55.82	-0.28
dist_delegacia	2.71	4.06	0.00	0.69	1.22	2.06	17.70	-0.06
dist_favela	0.55	0.47	0.00	0.26	0.45	0.68	6.08	0.23
dist_lagoa	4.74	4.41	0.00	1.06	4.18	6.03	27.76	-0.50
dist_logradoiro	0.28	0.39	0.00	0.03	0.15	0.36	6.54	-0.12
dist_metro	5.76	7.50	0.01	0.58	1.86	9.29	39.02	-0.11
dist_praia	3.87	3.70	0.00	0.63	2.23	6.69	22.22	-0.53
dist_saude_privada	0.28	0.32	0.00	0.09	0.19	0.36	7.60	-0.15
dist_saude_publica	0.89	0.82	0.00	0.36	0.62	1.08	5.99	0.07
dist_trem	5.89	4.56	0.00	1.77	5.57	8.25	17.70	0.33
dm_andar_alto	0.02	0.15	0.00	0.00	0.00	0.00	1.00	0.04
dm_andar_baixo	0.00	0.03	0.00	0.00	0.00	0.00	1.00	-0.01
dm_andar_inteiro	0.02	0.15	0.00	0.00	0.00	0.00	1.00	0.16
dm_armario	0.32	0.47	0.00	0.00	0.00	1.00	1.00	0.17
dm_banheira	0.05	0.22	0.00	0.00	0.00	0.00	1.00	0.04
dm_blindex	0.08	0.27	0.00	0.00	0.00	0.00	1.00	-0.08
dm_churrasqueira	0.21	0.40	0.00	0.00	0.00	0.00	1.00	-0.08
dm_closet	0.06	0.23	0.00	0.00	0.00	0.00	1.00	0.19
dm_cobertura	0.03	0.18	0.00	0.00	0.00	0.00	1.00	0.09
dm_copa	0.16	0.36	0.00	0.00	0.00	0.00	1.00	0.21
dm_creche	0.03	0.17	0.00	0.00	0.00	0.00	1.00	-0.02
dm_dep_empregada	0.36	0.48	0.00	0.00	0.00	1.00	1.00	0.29
dm_duplex	0.02	0.13	0.00	0.00	0.00	0.00	1.00	0.08
dm_elevador	0.17	0.37	0.00	0.00	0.00	0.00	1.00	0.01
dm_elevador_privado	0.01	0.09	0.00	0.00	0.00	0.00	1.00	0.01
dm_escritura	0.09	0.28	0.00	0.00	0.00	0.00	1.00	0.08
dm_esquina	0.02	0.15	0.00	0.00	0.00	0.00	1.00	-0.01
dm_est_visitantes	0.05	0.22	0.00	0.00	0.00	0.00	1.00	0.01
dm_frente	0.05	0.22	0.00	0.00	0.00	0.00	1.00	0.05
dm_fundos	0.02	0.13	0.00	0.00	0.00	0.00	1.00	0.02
dm_granito	0.02	0.14	0.00	0.00	0.00	0.00	1.00	0.01
dm_hidrometro	0.00	0.04	0.00	0.00	0.00	0.00	1.00	-0.01
dm_indevassavel	0.04	0.20	0.00	0.00	0.00	0.00	1.00	-0.02
dm_lateral	0.01	0.11	0.00	0.00	0.00	0.00	1.00	0.02

Continua na próxima página

Tabela 4.1: Análise descritiva das variáveis propostas a modelagem.

	mean	std	min	25%	50%	75%	max	Corr(Preco)
dm_linear	0.01	0.09	0.00	0.00	0.00	0.00	1.00	0.08
dm_mezanino	0.00	0.06	0.00	0.00	0.00	0.00	1.00	-0.01
dm_original	0.02	0.14	0.00	0.00	0.00	0.00	1.00	0.07
dm_piscina	0.20	0.40	0.00	0.00	0.00	0.00	1.00	-0.01
dm_planejad	0.07	0.26	0.00	0.00	0.00	0.00	1.00	0.05
dm_play	0.22	0.41	0.00	0.00	0.00	0.00	1.00	0.01
dm_porcelanato	0.04	0.20	0.00	0.00	0.00	0.00	1.00	-0.01
dm_portaria	0.09	0.29	0.00	0.00	0.00	0.00	1.00	0.03
dm_recuado	0.00	0.05	0.00	0.00	0.00	0.00	1.00	0.03
dm_sala_jantar	0.13	0.33	0.00	0.00	0.00	0.00	1.00	0.17
dm_salao_festas	0.25	0.43	0.00	0.00	0.00	0.00	1.00	-0.06
dm_salao_jogos	0.08	0.27	0.00	0.00	0.00	0.00	1.00	-0.03
dm_sauna	0.14	0.35	0.00	0.00	0.00	0.00	1.00	0.03
dm_seguranca	0.05	0.22	0.00	0.00	0.00	0.00	1.00	0.00
dm_servico	0.20	0.40	0.00	0.00	0.00	0.00	1.00	0.10
dm_sol_manha	0.00	0.07	0.00	0.00	0.00	0.00	1.00	-0.01
dm_sol_tarde	0.00	0.01	0.00	0.00	0.00	0.00	1.00	-0.01
dm_terraco	0.03	0.17	0.00	0.00	0.00	0.00	1.00	0.09
dm_triplex	0.00	0.04	0.00	0.00	0.00	0.00	1.00	0.02
dm_varanda	0.30	0.46	0.00	0.00	0.00	1.00	1.00	0.05
dm_vista_mar	0.05	0.22	0.00	0.00	0.00	0.00	1.00	0.09
garagem	1.05	0.81	0.00	1.00	1.00	1.00	4.00	0.32
lat	-22.94	0.05	-23.07	-22.98	-22.94	-22.92	-22.79	-0.48
lng	-43.27	0.10	-43.72	-43.34	-43.24	-43.19	-43.16	0.28
m2	9.31	5.13	0.47	5.51	7.82	11.95	30.14	0.72
preco	981.48	790.44	45.00	420.16	717.11	1285.19	4943.05	1.00
quartos	2.47	0.78	1.00	2.00	2.00	3.00	4.00	0.50
se_anos_estudo	10.71	1.88	4.59	9.85	11.28	11.77	13.89	0.56
se_crm_roubo	1212.31	648.01	374.15	809.53	1167.36	1432.40	4871.75	-0.25
se_crm_violento	745.71	383.43	423.45	567.54	644.47	764.41	3306.23	-0.14
se_idh	0.91	0.05	0.71	0.89	0.93	0.96	0.97	0.53
se_perc_alfabetizacao	98.50	1.14	91.95	98.16	98.80	99.24	100.00	0.39
se_renda	3594.34	1772.62	685.63	2144.67	3357.40	4269.21	8286.46	0.69
se_saneamento	0.94	0.09	0.30	0.89	0.98	1.00	1.00	0.20
suites	0.66	0.76	0.00	0.00	1.00	1.00	4.00	0.39

Nota-se que a distância mínima a uma favela é zero, indicando que há anúncio de imóveis localizados em uma dessas áreas. Não obstante, o terceiro quartil abriga imóveis a menos de 700 metros dessas regiões, cerca de 8 minutos de caminhada¹. O mapa de uso do solo na p. 10 mostra que as favelas estão razoavelmente espalhadas ao longo de toda a cidade, mas com a concentração de imóveis próximos ao centro, podemos

¹Precisamente 8,4 minutos de caminhada para uma velocidade média de 5Km/h.

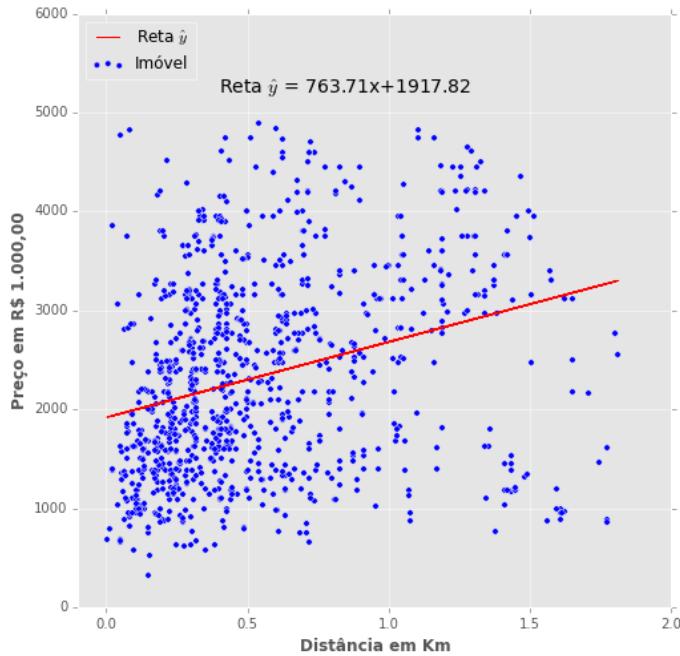
considerar uma certa saturação da ocupação urbana ao redor das favelas, comumente utilizado pelos estudos sociais como exemplo visual da extrema desigualdade na cidade. Buscando-se conhecer o relacionamento entre essa distância e a variação de preços, selecionamos as dez favelas com maior quantidade de imóveis próximos e aplicamos uma regressão de ordem 1. A tabela 4.2 resume os resultados encontrados, apresentando para cada uma das favelas a quantidade de imóveis próximos, a menor e maior distância em metros de um imóvel, o coeficiente da regressão para a variável *dist_favela* em R\$/m, o nível de significância estatística, a estatística R^2 e o(s) bairro(s) onde a favela está localizada:

Tabela 4.2: Dez favelas com maior número de imóveis próximos.

Nome	Qtd	Min_dist	Max_dist	Coef	P> t	R^2	Bairro
Morro do Cantagalo	1078	0.00	1.81	763.71	0.00	0.09	Ipanema/Copacabana
Morro Azul	1008	0.06	1.04	360.87	0.00	0.01	Flamengo
Ladeira dos Tabajaras, no 248	895	0.04	0.88	1040.21	0.00	0.08	Copacabana
Pavao-Pavaozinho	851	0.00	0.78	1200.75	0.00	0.06	Copacabana
Salgueiro	843	0.04	1.11	-114.05	0.01	0.01	Tijuca
Chacara do Ceu	757	0.12	1.79	-500.50	0.00	0.05	Vidigal
Rua Silva Teles, no 110	675	0.00	1.12	361.61	0.00	0.08	Andarai
Tirol	646	0.00	0.93	168.64	0.00	0.02	Jacarepagua
Babilonia	573	0.01	0.80	31.03	0.84	0.00	Leme
Morro Santa Marta	497	0.01	1.07	401.03	0.00	0.02	Botafogo

Temos então que para o Morro do Cantagalo, a que possui maior quantidade de imóveis próximos ao redor, o coeficiente indica uma valorização de R\$ 763,71/m de distância no intervalo de 0 a 1,81 Km. Interessante notar que esse resultado não é homogêneo para todas as ocorrências, dado que Salgueiro tem resultado inverso. A respeito da comunidades Babilônia, o resultado é inconclusivo devido ao limite de significância de 0,05 ter sido ultrapassado.

Figura 4.1: Regressão de preço pela distância à favela Morro do Cantagalo.



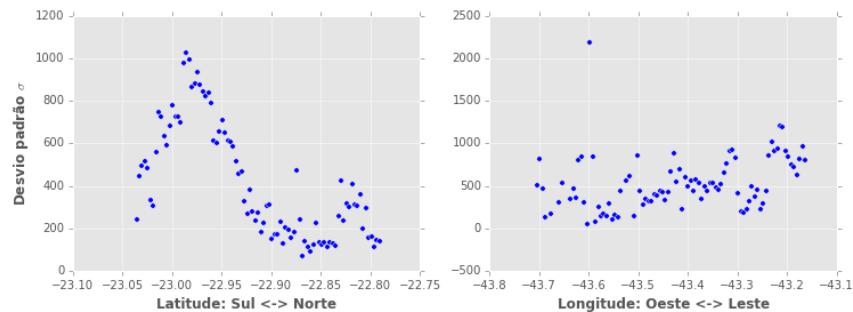
Outro ponto a destacar é a maioria das variáveis dicotômicas não apresentarem ocorrência da respectiva benfeitoria no imóvel ou item de serviço ou lazer no condomínio até o terceiro quartil, com exceção apenas de ***dm_armario***, ***dm_dep_empregada*** e ***dm_varanda***. Observa-se que aproximadamente metade dos imóveis possuem 2 quartos, menos de 25% mais de uma suíte e mais de uma vaga de garagem. A variável ***area*** nos mostra que apenas 25% dos imóveis possuem área inferior a $67\ m^2$, usualmente o tamanho de novos lançamentos.

Com base na análise descritiva e execução de alguns modelos iniciais de regressão para teste, adquirimos conhecimentos para excluir as variáveis ***latitude***, ***longitude***, ***dist_centro_lat***, ***dist_centro_lng***, ***condominio***, ***m2*** e ***se_anos_estudo***, dos modelos finais pelos motivos explicados a seguir. O objetivo da inclusão das variáveis ***latitude*** e ***longitude*** era quantificar a influência da posição geográfica latitudinal

e longitudinal ao preço do imóvel, entretanto observamos que tais variáveis não são bons regressores devido a distribuição dos imóveis ao longo desses eixos não adequar-se ao requisito de homoscedasticidade para a variável *preco*, como pode ser visto na fig. 4.2. O mesmo pode ser dito a respeito das variáveis *dist_centro_lat*, *dist_centro_lng*. A variável *m2* também apresentou-se como um mau regressor pela sua óbvia dependência com *area* e *preco*, entretanto pode ser usada como variável independente em extensão a esse estudo.

A respeito da variável *condominio*, embora possua uma correlação de 0.64 com *preco*, arbitramos por removê-la por entendermos que, como eliminamos apartamentos em uma mesma coordenada, sua atuação como variável socioambiental ficou bastante reduzida, além do fato de estar mais relacionada aos itens de serviços e lazer presentes no condomínio, já representados no modelo como variáveis dicotômicas, do que ao imóvel. A variável *se_anos_estudo* foi removida em função da correlação com *se_idh* ser 0.95.

Figura 4.2: Desvio padrão da variável *preco* ao longo da Latitude e Longitude da cidade.



4.2 Modelo hedônico para a cidade do Rio de Janeiro

Iniciamos a discussão dos resultados da modelagem hedônica para a cidade do Rio de Janeiro com um primeiro modelo que relaciona todas as variáveis independentes

disponíveis após a remoção das mencionadas na seção anterior e, adicionalmente, sem considerar a autocorrelação espacial da variável dependente *preco*, apresentado na tabela 4.3 em ordem alfabética para fácil localização. A coluna "coef" contém o valor dos coeficientes $\hat{\beta}_i$ resultante da regressão, "std err" indica o desvio padrão da variável, "t" a estatística de teste de significância estatística, "P>|t|" a probabilidade de "t" ocorrer por acaso, "95% Conf Int" o intervalo de confiança a 95% e "pos" indica o ranking da variável na ordem decrescente dos coeficientes, onde o valor 1 indica a primeira posição, 2 a segunda e assim sucessivamente.

Tabela 4.3: Resultado do modelo com todas as variáveis, ordenado por nome. Valores em R\$ 1.000,00.

	coef	std err	t	P> t	[95% Conf. Int.]	pos
Intercept	7004.37	354.31	19.77	0.00	6309.918 7698.829	1
area	6.57	0.07	92.70	0.00	6.434 6.712	32
dist_bombeiro	8.39	2.99	2.80	0.01	2.522 14.252	31
dist_centro	49.67	1.14	43.39	0.00	47.427 51.914	20
dist_delegacia	-40.73	1.34	-30.35	0.00	-43.361 -38.101	42
dist_favela	71.57	6.08	11.76	0.00	59.645 83.494	16
dist_lagoa	-24.93	1.03	-24.18	0.00	-26.949 -22.907	37
dist_logradouro	92.54	7.11	13.02	0.00	78.604 106.473	12
dist_metro	-51.57	1.34	-38.43	0.00	-54.202 -48.942	45
dist_praia	-63.39	1.62	-39.23	0.00	-66.553 -60.219	46
dist_saude_privada	72.50	9.14	7.93	0.00	54.579 90.426	15
dist_saude_publica	-108.38	4.39	-24.66	0.00	-116.989 -99.764	51
dist_trem	-15.22	1.88	-8.09	0.00	-18.907 -11.529	36
dm_andar_alto	-37.69	14.51	-2.60	0.01	-66.140 -9.240	41
dm_andar_inteiro	95.42	14.48	6.59	0.00	67.033 123.811	11
dm_armario	13.07	5.19	2.52	0.01	2.903 23.231	30
dm_banheira	40.42	11.97	3.38	0.00	16.961 63.882	22
dm_blinindex	-33.55	8.55	-3.92	0.00	-50.322 -16.788	39
dm_churrasqueira	-65.32	7.55	-8.65	0.00	-80.118 -50.523	47
dm_closet	62.58	10.07	6.21	0.00	42.838 82.317	19
dm_cobertura	-91.99	14.54	-6.33	0.00	-120.483 -63.498	50
dm_copa	20.74	6.45	3.21	0.00	8.088 33.389	28
dm_elevador	-35.49	6.46	-5.49	0.00	-48.150 -22.825	40
dm_est_visitantes	-85.78	12.27	-6.99	0.00	-109.829 -61.732	49
dm_frente	29.30	9.98	2.94	0.00	9.745 48.850	24
dm_fundos	-49.91	16.42	-3.04	0.00	-82.093 -17.720	44
dm_indevassavel	-26.39	11.40	-2.31	0.02	-48.734 -4.044	38
dm_linear	78.03	25.78	3.03	0.00	27.496 128.566	14
dm_piscina	21.04	7.95	2.65	0.01	5.458 36.616	27
dm_play	30.82	6.33	4.87	0.00	18.419 43.231	23

Continued on next page

Tabela 4.3: Resultado do modelo com todas as variáveis, ordenado por nome. Valores em R\$ 1.000,00.

	coef	std err	t	P> t	[95% Conf. Int.]	pos
dm_portaria	19.74	8.02	2.46	0.01	4.025 35.464	29
dm_sala_jantar	43.28	7.37	5.87	0.00	28.838 57.717	21
dm_salao_jogos	-46.98	10.74	-4.37	0.00	-68.040 -25.924	43
dm_servico	25.26	5.89	4.29	0.00	13.714 36.802	25
dm_triplex	-121.15	61.24	-1.98	0.05	-241.174 -1.126	52
dm_vista_mar	70.83	10.11	7.00	0.00	51.011 90.659	17
garagem_1	126.91	5.95	21.31	0.00	115.240 138.584	10
garagem_2	153.16	8.45	18.14	0.00	136.608 169.713	8
garagem_3	157.34	14.20	11.08	0.00	129.516 185.174	7
garagem_4	214.80	28.73	7.48	0.00	158.493 271.110	4
quartos_2	23.08	8.43	2.74	0.01	6.559 39.592	26
quartos_3	63.68	9.53	6.69	0.00	45.011 82.354	18
quartos_4	172.78	13.64	12.67	0.00	146.057 199.512	5
se_crm_roubo	-0.02	0.00	-4.06	0.00	-0.029 -0.010	35
se_crm_violento	0.02	0.01	2.29	0.02	0.002 0.030	34
se_idh	-634.90	123.02	-5.16	0.00	-876.019 -393.772	53
se_perc_alfabetizacao	-73.43	4.33	-16.95	0.00	-81.927 -64.939	48
se_renda	0.19	0.00	60.21	0.00	0.180 0.192	33
se_saneamento	520.61	53.63	9.71	0.00	415.493 625.718	2
suites_1	80.37	5.47	14.70	0.00	69.658 91.090	13
suites_2	129.99	10.17	12.78	0.00	110.051 149.927	9
suites_3	157.41	16.03	9.82	0.00	125.986 188.827	6
suites_4	227.86	30.15	7.56	0.00	168.776 286.950	3

Para esse modelo, a estatística R^2 é de 0,793 e o $RMSE$ igual a 358,843. Sabendo que o maior preço em nosso conjunto de dados é próximo a R\$ 5 milhões, o valor expressivo da constante ***Intercept*** parece indicar um *overfit* nessa variável, a ser compensado pelas variáveis com coeficiente negativo, principalmente as variáveis de distância, que aqui servem para tentar mitigar a autocorrelação entre os preços de imóveis.

Mas esse modelo utiliza todas as variáveis independentes disponíveis, e sabemos que pode haver problemas de multicolinearidade entre elas. Ao proceder essa investigação, identificando variáveis que possuem um coeficiente de correlação superior ou igual a 0,8 a alguma outra, construímos a seguinte lista para ser removida do próximo modelo: ***dist_bombeiro, dist_delegacia, dist_lagoa, dist.metro, dist_saude_publica, dist_trem, dm_banheira, dm_churrasqueira, dm_duplex, dm_est_visitantes,***

dm_play, dm_salao_festas, dm_salao_jogos, dm_sauna, se_crm_roubo, se_crm_violento, se_perc_alfabetizacao, se_renda, se_saneamento. Com isso, temos o seguinte resultado para um segundo modelo, sem as variáveis correlacionadas entre si pelo coeficiente de Pearson, na tabela 4.4.

Tabela 4.4: Resultado do modelo sem variáveis correlacionadas, ordenado por nome. Valores em R\$ 1.000,00.

	coef	std err	t	P> t	[95% Conf. Int.]	pos
Intercept	-2498.26	73.32	-34.07	0.00	-2641.979 -2354.544	42
area	7.17	0.09	82.46	0.00	6.996 7.337	25
dist_centro	-6.42	0.41	-15.52	0.00	-7.233 -5.611	26
dist_favela	82.79	6.40	12.94	0.00	70.245 95.331	12
dist_logradouro	-124.77	8.32	-14.99	0.00	-141.074 -108.456	38
dist_praia	-50.89	0.94	-54.28	0.00	-52.727 -49.051	33
dist_saude_privada	204.60	10.83	18.88	0.00	183.366 225.838	2
dm_andar_inteiro	131.83	18.27	7.22	0.00	96.029 167.641	6
dm_armario	35.64	6.69	5.33	0.00	22.524 48.751	22
dm_blindex	-81.50	10.93	-7.46	0.00	-102.914 -60.083	36
dm_closet	93.86	12.86	7.30	0.00	68.648 119.069	10
dm_cobertura	-157.79	16.76	-9.41	0.00	-190.646 -124.933	39
dm_copa	50.25	8.23	6.11	0.00	34.127 66.372	21
dm_creche	-40.13	20.00	-2.01	0.04	-79.336 -0.926	29
dm_dep_empregada	32.14	6.66	4.82	0.00	19.079 45.205	23
dm_elevador	-30.11	8.30	-3.63	0.00	-46.384 -13.835	28
dm_elevador_privado	-69.43	30.42	-2.28	0.02	-129.053 -9.797	35
dm_escritura	21.92	10.84	2.02	0.04	0.660 43.171	24
dm_esquina	-101.80	20.72	-4.91	0.00	-142.411 -61.188	37
dm_fundos	-49.90	20.73	-2.41	0.02	-90.530 -9.280	32
dm_indevassavel	-46.67	14.34	-3.25	0.00	-74.777 -18.555	31
dm_mezanino	-357.23	49.47	-7.22	0.00	-454.195 -260.262	41
dm_original	69.54	20.32	3.42	0.00	29.705 109.371	15
dm_piscina	-19.72	8.05	-2.45	0.01	-35.494 -3.943	27
dm_porcelanato	-40.15	13.88	-2.89	0.00	-67.356 -12.951	30
dm_portaria	54.52	10.36	5.26	0.00	34.213 74.818	19
dm_sala_jantar	57.35	9.24	6.21	0.00	39.244 75.462	18
dm_servico	62.49	7.51	8.32	0.00	47.771 77.205	17
dm_triplex	-192.29	77.19	-2.49	0.01	-343.580 -40.995	40
dm_varanda	-58.77	7.01	-8.38	0.00	-72.515 -45.026	34
dm_vista_mar	114.65	12.68	9.04	0.00	89.796 139.500	8
garagem_1	157.35	7.38	21.33	0.00	142.895 171.812	4
garagem_2	107.53	10.50	10.24	0.00	86.947 128.106	9
garagem_3	71.44	17.72	4.03	0.00	36.710 106.165	14
garagem_4	77.41	36.15	2.14	0.03	6.552 148.267	13
quartos_3	52.86	7.09	7.46	0.00	38.970 66.753	20
quartos_4	204.53	13.50	15.15	0.00	178.063 230.995	3

Continued on next page

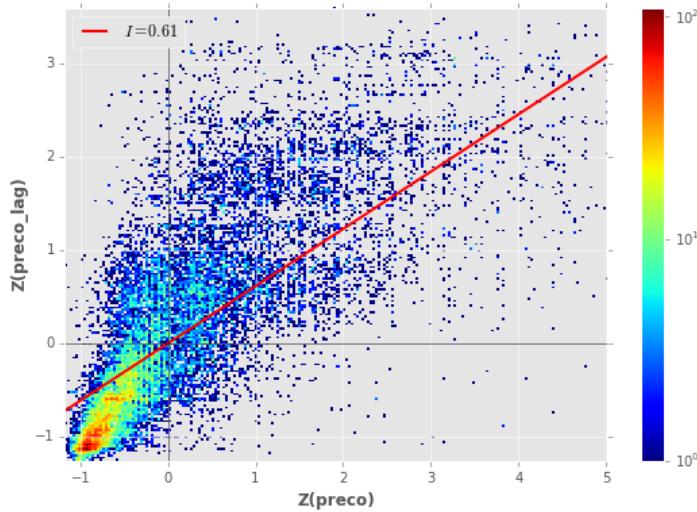
Tabela 4.4: Resultado do modelo sem variáveis correlacionadas, ordenado por nome. Valores em R\$ 1.000,00.

	coef	std err	t	P> t	[95% Conf. Int.]	pos
se_idh	3009.45	77.65	38.76	0.00	2857.247 3161.644	1
suites_1	83.60	6.83	12.24	0.00	70.214 96.981	11
suites_2	119.47	12.83	9.31	0.00	94.327 144.615	7
suites_3	66.16	20.16	3.28	0.00	26.653 105.676	16
suites_4	146.10	37.99	3.85	0.00	71.624 220.567	5

Como era de se esperar, as estatísticas R^2 e $RMSE$ pioraram em relação ao primeiro, pela remoção de informações - ainda que autocorrelacionadas - do modelo, para os valores 0,671 e 452,850 respectivamente. A constante **Intercept** tomou um valor ainda mais expressivo, assim como **se_idh**, mas agora, em posições contrárias no ranking. A remoção das variáveis de distância mencionadas no parágrafo acima elevou a importância das outras que restaram, e para compensar o *overfit* da constante, as variáveis dicotômicas assumiram valores maiores, ainda que com baixa frequência de ocorrência no conjunto de dados, como visto na análise descritiva de variáveis.

Faz-se então necessário considerar o efeito da autocorrelação espacial da variável **preco**. A hipótese nula H_o de que não há autocorrelação espacial é rejeitada pelo índice de Moran de autocorrelação espacial $I = 0,614$ para uma matriz W de pesos espaciais dos imóveis construída pelo algoritmo $Knn = 100$, cuja constatação é visivelmente explícita na avaliação do gráfico de Moran, que nós estendemos para também representar a acumulação de imóveis em um mesmo ponto numa escala log normal, fig. 4.3.

Figura 4.3: Gráfico de Moran para Knn=100.



Dado que nossa matriz W foi construída tendo a entrada $w_{(i,j)} = 1$ para os 100 imóveis na coluna j mais próximos ao imóvel na linha i , e 0 caso contrário, cada observação i tem sua nova variável ***preco_lag*** calculada pela fórmula 4.1:

$$preco_lag_i = \frac{\sum_j w_{(i,j)} preco_j}{\sum_j 1_{w_{(i,j)}=1}} \quad (4.1)$$

No gráfico 4.3, os pontos nos quadrantes superior direito e inferior esquerdo representam associação espacial positiva, ou seja, a acumulação de pontos nestes quadrantes indicam que preços próximos são acompanhados pela média espacial dos vizinhos acima e abaixo da média amostral, respectivamente. A coloração da escala *Log Normal* nos permite identificar uma grande concentração de imóveis concentrados abaixo da média amostral. Reciprocamente, os quadrantes superior esquerdo e inferior direito são áreas de baixa associação espacial com seus vizinhos. A construção da matriz W e cálculo do índice de Moran foram obtidos com a utilização da biblioteca PySAL de [Rey e Anselin \(2010\)](#).

Desta forma, atualizamos o modelo anterior adicionando a variável *preco_lag*, que após execução apresenta os seguintes resultados na tabela 4.5.

Tabela 4.5: Resultado do modelo *lag* espacial, ordenado por nome. Valores em R\$ 1.000,00.

	coef	std err	t	P> t	[95% Conf. Int.]	pos
Intercept	279.92	56.30	4.97	0.00	169.564 390.284	1
area	6.05	0.06	93.43	0.00	5.922 6.176	25
dist_centro	-11.78	0.30	-39.14	0.00	-12.371 -11.191	29
dist_favela	28.19	4.65	6.06	0.00	19.073 37.313	19
dist_logradouro	-14.60	6.07	-2.40	0.02	-26.507 -2.700	30
dist_praia	4.42	0.77	5.74	0.00	2.909 5.927	26
dist_saude_privada	47.61	7.92	6.01	0.00	32.098 63.131	14
dm_andar_inteiro	75.09	13.25	5.67	0.00	49.119 101.064	11
dm_closet	50.29	9.26	5.43	0.00	32.135 68.439	13
dm_cobertura	-85.97	12.09	-7.11	0.00	-109.675 -62.268	35
dm_copa	22.19	5.91	3.75	0.00	10.607 33.776	20
dm_dep_empregada	-24.07	4.73	-5.08	0.00	-33.347 -14.789	31
dm_elevador	-26.05	5.83	-4.47	0.00	-37.474 -14.627	32
dm_esquina	-48.24	14.31	-3.37	0.00	-76.286 -20.187	34
dm_frente	30.36	9.08	3.34	0.00	12.559 48.157	18
dm_fundos	-89.08	15.00	-5.94	0.00	-118.485 -59.682	36
dm_granito	-32.30	14.18	-2.28	0.02	-60.086 -4.518	33
dm_mezanino	-158.74	35.86	-4.43	0.00	-229.028 -88.452	38
dm_piscina	21.95	5.73	3.83	0.00	10.730 33.175	21
dm_portaria	17.19	7.42	2.32	0.02	2.646 31.725	23
dm_sala_jantar	39.26	6.55	5.99	0.00	26.418 52.110	17
dm_seguranca	19.34	9.51	2.03	0.04	0.698 37.986	22
dm_servico	13.59	5.33	2.55	0.01	3.146 24.039	24
dm_tríplex	-140.08	55.99	-2.50	0.01	-249.816 -30.347	37
dm_varanda	-9.97	5.05	-1.97	0.05	-19.872 -0.064	28
dm_vista_mar	53.20	9.17	5.80	0.00	35.217 71.177	12
garagem_1	123.01	5.34	23.05	0.00	112.546 133.464	6
garagem_2	124.57	7.58	16.43	0.00	109.714 139.429	5
garagem_3	115.14	12.82	8.98	0.00	90.005 140.266	8
garagem_4	183.62	26.21	7.00	0.00	132.238 234.993	4
preco_lag	0.83	0.01	159.70	0.00	0.817 0.837	27
quartos_2	47.18	7.68	6.14	0.00	32.127 62.237	15
quartos_3	81.34	8.70	9.35	0.00	64.291 98.394	10
quartos_4	200.88	12.45	16.14	0.00	176.477 225.280	3
se_idh	-897.75	61.46	-14.61	0.00	-1018.214 -777.292	39
suites_1	42.93	4.97	8.64	0.00	33.198 52.668	16
suites_2	93.05	9.30	10.00	0.00	74.822 111.286	9
suites_3	118.08	14.60	8.09	0.00	89.462 146.704	7
suites_4	231.39	27.54	8.40	0.00	177.412 285.358	2

Para esse modelo temos as seguintes estatísticas: $R^2 = 0.83$ e $RMSE = 328.50$,

melhor que os dois anteriores, o que indica que a perda de informação com a remoção das variáveis correlacionadas entre si no segundo modelo foi completamente substituída com a adição do lag espacial. O coeficiente da variável ***preco_lag*** indica que o preço médio dos imóveis no conjunto de dados é constituído em 83% da média de preços dos imóveis espacialmente relacionados na matrix W de pesos espaciais. Essa proporção demonstra o tamanho do efeito da autocorrelação espacial dos preços de imóveis e o resultado prático é a atenuação dos coeficientes das demais variáveis, que agora destituídos em parte da parcela de influência dos imóveis vizinhos, representam mais acuradamente a contribuição de suas naturezas à formação do preço.

Com esse resultado, podemos determinar as características mais valorizadas observando os valores dos coeficientes das respectivas variáveis. Primeiro, ao comparar as variáveis dentro de suas classificações, temos que entre as estruturais a variável ***suites_4*** representa a característica mais valorizada, seguida por ***quartos_4, garagem_4, garagem_2, garagem_1, suites_3, garagem_3, suites_2, quartos_3, quartos_2*** e finalmente ***suites_1***. A variável estrutural dicotômica mais valorizada é ***dm_andar_inteiro***, seguida por ***dm_vista_mar, dm_closet, dm_sala_jantar***. Dentre as variáveis de acessibilidade, ***dist_saude_privada*** precede ***dist_favela, dist_praia, dist_centro*** e ***dist_logradouro***, nessa ordem. As demais variáveis dicotômicas podem ser vistas pela numeração na coluna "pos". Embora a área ocupe a 25^a posição pela ordem decrescente do coeficiente, lembramos que conforme pode ser visto na tabela 4.1, p. 67, 75% dos imóveis possuem área superior ou igual a $67m^2$. Logo 75% dos imóveis tem custo mínimo de $R\$6,05/m^2 * 67m^2 = R\$405,35$, valor superior a qualquer coeficiente da tabela 4.5.

Desta forma, comparando-se as variáveis em conjunto, consideramos que a localização é a característica mais valorizada, seguida pela área, quantidade de cômodos e vagas de garagem, características específicas a imóveis de alto valor como andar inteiro e vista para o mar e *closet*, seguidos pela distância à localização de unidades de saúde

privada, favelas, centro da cidade e logradouro principal. O conjunto de unidades de saúde privada contém todas as empresas registradas junto à Prefeitura Municipal como clínicas, consultórios, laboratórios de exame, o que nos leva a interpretar que a maior parte destes estabelecimentos estão localizados em áreas de comércio dos bairros. Logo, a precedência dessa variável em relação a distância a favelas pode significar uma preferência por afastar-se das áreas comerciais, seja em função da poluição sonora devido ao trânsito de veículos e/ou aglomeração de pedestres, seja pela concentração de furtos e outros eventos associados à segurança pública em lugares de grande movimentação. Como estamos tratando de imóveis do tipo apartamento, o fato de que a maioria dos condomínios possuem acesso controlado pode atenuar a sensação de insegurança mesmo próximo a favelas.

Dentre as posições de um apartamento em relação a rua de acesso, a preferência são os localizados de frente, seguidos pelo de esquina e por último, os localizados nos fundos. A ausência da posição lateral, variável *dm_lateral*, indica que foi removida por não ser estatisticamente significante e dessa forma, a inferência de uma nova observação onde essas variáveis de posição em relação a rua de acesso fossem 0 indicaria o valor do imóvel para um apartamento lateral ou de posição desconhecida. A presença de varanda incorre em uma redução de R\$ 9.970,00, possivelmente associado a falta de privacidade do morador se a varanda tem vista livre a ruas.

A única variável socioambiental que restou ao modelo, *se_idh*, ocupa a última posição no ranking e com um valor que contraria o senso comum ao informar um decréscimo de R\$ 8.977,50 por ponto percentual de avanço no IDH. Lembrando que tal variável assume comportamento de determinação espacial dos imóveis devido a sua distribuição ser por bairro, a expressiva mantissa indica que a presença de variáveis de acessibilidade e *lag* espacial não são suficientes para a explicação completa da dependência da variável *preco* e que há iteração entre essas variáveis para a minimização do erro médio quadrado.,

Desta forma, verificamos que a modelagem hedônica para uma cidade, uma aplicação de regressão linear de mínimos quadrados ordinária, necessita abordar adequadamente o efeito da autocorrelação espacial a fim de construir resultados confiáveis. Com base no modelo apresentado, o preço do imóvel é drasticamente influenciado pela média dos imóveis vizinhos, onde o índice I de Moran de autocorrelação espacial é de 0.61 para uma média aritmética dos preços dos 100 imóveis mais próximos. Em seguida, temos as características estruturais como área e de quantificação de suítes, vagas de garagem e quartos como de maior interferência no preço. A distância a favelas interfere menos do que as características estruturais, e menos também do que a proximidade às unidades públicas de saúde, em que interpretamos como áreas comerciais de movimento relevante.

4.3 Modelo hedônico para os bairros

A seção anterior fez uma regressão considerando todos os imóveis da cidade. Uma das desvantagens dessa abordagem ampla é o impacto da heterocedasticidade das variáveis ao longo da cidade, o que pode prejudicar uma avaliação específica a uma região de menor área de interesse como bairros. Desta forma, propomos como resultado auxiliar desse estudo a geração de modelos hedônicos específicos aos bairros, tomando-se como base as variáveis utilizadas para o modelo de lag espacial da cidade. A decisão de utilizar as variáveis do modelo anterior restringe nosso conjunto inicial de bairros para aqueles que possuem uma quantidade de imóveis superior ao número de variáveis a serem usadas, mas de outra forma seria necessário uma análise individual do contexto de cada bairro, o que demandaria um tempo elevado. Utilizaremos a mesma variável ***preco_lag*** calculada anteriormente para considerar plenamente a autocorrelação espacial nos imóveis limítrofes, incorporando a vinhança com aqueles localizados em bairro contíguo.

Dos 155 bairros registrados em nosso conjunto de dados, somente 73 puderam ser modelados, ainda assim, uma quantidade considerável para ser feito manualmente. Automatizamos a execução das regressões, com a preocupação em apenas remover vairáveis com $p\text{-value} > 0,05$ de confiança. Os resultados para todos os bairros estão disponíveis na Internet², e replicamos nesse trabalho somente os 5 maiores bairros em quantidade de imóveis, disponíveis no apêndice K, **Resultados do Modelo Hedônico para os 5 bairros em maior quantidade de observações**, p. 216, que são Copacabana, Barra da Tijuca, Recreio dos Bandeirantes, Botafogo e Tijuca. Destes, somente Botafogo não manteve a variável ***preco_lag*** no resultado final. Em todos observa-se a predominância da valorização dos imóveis pelo número de suítes.

Uma análise detalhada do resultado por bairro está além do escopo desse estudo. Entretanto, buscamos realizar uma breve análise exploratória a fim de verificar as variáveis mais presentes entre os modelos resultantes dos bairros e quais bairros cujos modelos finais apresentam mais variáveis. Por fim, apresentamos os valores máximos e mínimos para as variáveis independentes entre os modelos e o respectivo bairro de ocorrência desses valores extremos.

Esses resultados são apresentados em três tabelas. A primeira delas é a tabela 4.6 que lista as variáveis ordenadas pela quantidade de ocorrência em bairros. Não é surpresa verificar que a variável ***area*** é a de maior ocorrência pois tanto é um atributo obrigatório no cadastro do classificados ZAP Imóveis, quanto é a segunda característica mais valorizada, após a localização, no modelo para a cidade. Em seguida posicionam-se variáveis estruturais, indicadoras da presença de uma única suíte no imóvel e quantidade de vagas de garagem. No fim da tabela encontramos a variável dicotômica indicadora da presença hidrômetro em última posição, com uma única ocorrência. É válido destacar que a variável ***preco_lag*** permaneceu em apenas 23 dos 73 bairros modelados, o que nos leva a conjecturar que para 50 bairros, a abordagem da autocorrelação espacial

²https://github.com/srodriguez/fgv_dissertacao/tree/master/modelos_bairro

como a média de preços de imóveis vizinhos não é estatisticamente significante.

Tabela 4.6: Variáveis por quantidade de ocorrência em bairros

	Bairros	Perc(%)		Bairros	Perc(%)
area	72	98	dm_portaria	12	16
suites_1	50	68	dist_centro	11	15
garagem_2	42	57	dm_duplex	11	15
dist_metro	38	52	suites_4	11	15
garagem_1	38	52	dm_segurança	11	15
suites_2	32	43	dm_salao_festas	10	13
quartos_4	30	41	dist_saude_privada	10	13
garagem_3	26	35	dist_trem	10	13
dist_favela	25	34	dm_armario	10	13
dm_sauna	25	34	dm_salao_jogos	9	12
quartos_2	25	34	dm_banheira	9	12
dm_varanda	23	31	dm_porcelanato	9	12
suites_3	23	31	dm_creche	9	12
preco_lag	23	31	dm_elevador	9	12
dist_praia	22	30	dm_vista_mar	9	12
dist_bombeiro	21	28	dm_lateral	7	9
dm_terraco	21	28	dm_linear	7	9
dm_play	21	28	dist_delegacia	7	9
quartos_3	20	27	dm_servico	7	9
dm_churrasqueira	18	24	dm_fundos	7	9
dm_piscina	17	23	dm_sol_manha	7	9
dist_saude_publica	16	21	dm_indevassavel	5	6
dm_closet	16	21	dm_triplex	4	5
dm_cobertura	16	21	dist_lagoa	4	5
dm_escritura	16	21	dm_esquina	4	5
dm_planejad	15	20	dm_original	3	4
dm_sala_jantar	14	19	dm_recuado	2	2
dm_andar_inteiro	14	19	dm_andar_alto	2	2
dist_logradouro	14	19	dm_est_visitantes	2	2
dm_dep_empregada	14	19	dm_granito	2	2
garagem_4	14	19	dm_elevador_privado	2	2
dm_frente	13	17	dm_mezanino	1	1
dm_copa	12	16	dm_hidrometro	1	1
dm_index	12	16			

A tabela 4.7 lista os bairros ordenados pela quantidade de variáveis presentes no respectivo modelo final, após a eliminação iterativa de variáveis acima do limite de significância de 5%. Observamos que não há uma relação direta entre vizinhança e quantidade de variáveis no modelo final, dado que bairros contíguos aparecem em posições distintas da tabela. Ademais, de um conjunto inicial de 69 variáveis disponíveis

originalmente, a primeira posição conta com apenas 29, cerca de 42% do total.

Tabela 4.7: Bairros por quantidade de variáveis

Variaveis	Perc(%)	Variaveis	Perc(%)		
Copacabana	29	42	Grajau	14	20
Vila Isabel	28	41	Del Castilho	14	20
Freguesia (Jacarepagua)	27	39	Pechincha	14	20
Recreio dos Bandeirantes	26	38	Penha Circular	14	20
Jardim Sulacap	26	38	Cachambi	14	20
Barra da Tijuca	23	33	Maracana	13	19
Urca	23	33	Encantado	13	19
Taquara	22	32	Penha	13	19
Ipanema	22	32	Riachuelo	13	19
Vila Valqueire	22	32	Rocha	13	19
Todos os Santos	21	30	Catete	13	19
Tijuca	20	29	Bras de Pina	12	17
Lagoa	19	27	Vila da Penha	12	17
Vargem Pequena	19	27	Praca da Bandeira	12	17
Flamengo	19	27	Piedade	12	17
Jacarepagua	18	26	Lapa	11	16
Praca Seca	18	26	Freguesia (Ilha)	11	16
Leme	18	26	Estacio	11	16
Centro	18	26	Jardim Botanico	11	16
Santa Teresa	18	26	Cascadura	11	16
Sao Conrado	18	26	Higienopolis	11	16
Botafogo	18	26	Curicica	11	16
Campo Grande	17	25	Cosme Velho	11	16
Bonsucesso	17	25	Sao Cristovao	10	14
Portuguesa	17	25	Lins de Vasconcelos	10	14
Tanque	17	25	Quintino Bocaiuva	10	14
Leblon	17	25	Anil	9	13
Engenho de Dentro	17	25	Irajá	9	13
Laranjeiras	16	23	Ramos	9	13
Jardim Guanabara	16	23	Sao Francisco Xavier	8	11
Rio Comprido	16	23	Gloria	8	11
Taua	16	23	Camorim	8	11
Meier	15	22	Jardim Carioca	8	11
Gavea	15	22	Olaria	8	11
Humaita	14	20	Sampaio	6	8
Andarai	14	20	Madureira	6	8
			Engenho Novo	6	8

Finalmente, a tabela 4.8 mostra os valores máximo e mínimo para cada variável e o respectivo bairro, comparando-se os resultados dos modelos para os 73 bairros.

Tabela 4.8: Máximos e mínimos das variáveis com respectivos bairros.

	Coef_max	Bairro_max	Coef_min	Bairro_min
Intercept	5084.16	Jardim Sulacap	-41465.21	Leme
area	19.04	Curicica	1.14	Jardim Sulacap
dist_bombeiro	1820.82	Portuguesa	-1395.05	Vila Valqueire
dist_centro	6218.75	Leme	-2344.91	Vila Valqueire
dist_delegacia	194.21	Vargem Pequena	-216.24	Sao Conrado
dist_favela	4255.75	Leme	-448.96	Lagoa
dist_lagoa	166.68	Barra da Tijuca	-991.03	Ipanema
dist_logradouro	1221.48	Sao Conrado	-1522.14	Sao Francisco Xavier
dist_metro	1771.36	Leme	-2948.95	Vila Valqueire
dist_praia	6034.29	Vila Valqueire	-2264.03	Portuguesa
dist_saude_privada	2107.78	Ipanema	-250.32	Laranjeiras
dist_saude_publica	751.98	Sao Francisco Xavier	-191.27	Bonsucesso
dist_trem	1094.92	Sao Francisco Xavier	-344.87	Vargem Pequena
dm_andar_alto	87.13	Taquara	-53.96	Copacabana
dm_andar_inteiro	584.54	Gavea	-406.30	Cachambi
dm_armario	150.37	Leme	-63.50	Jardim Sulacap
dm_banheira	279.37	Santa Teresa	-306.16	Madureira
dm_blindex	229.33	Jardim Sulacap	-103.08	Portuguesa
dm_churrasqueira	786.22	Urca	-243.26	Gavea
dm_closet	2228.91	Urca	-68.86	Taquara
dm_cobertura	1880.86	Sao Cristovao	-2060.70	Urca
dm_copa	205.81	Bras de Pina	-240.50	Campo Grande
dm_creche	691.51	Santa Teresa	-431.74	Praca Seca
dm_dep_empregada	430.74	Campo Grande	-118.08	Leme
dm_duplex	391.51	Jacarepagua	-357.78	Lagoa
dm_elevador	66.43	Jardim Sulacap	-220.49	Urca
dm_elevador_privado	310.12	Riachuelo	139.03	Vila Isabel
dm_escritura	162.05	Gloria	-1008.95	Urca

Continua na próxima página

Tabela 4.8: Máximos e mínimos das variáveis com respectivos bairros.

	Coef_max	Bairro_max	Coef_min	Bairro_min
dm_esquina	255.45	Flamengo	-623.67	Urca
dm_est_visitantes	-34.47	Freguesia (Jacarepagua)	-1410.09	Centro
dm_frente	484.63	Urca	-265.29	Sao Conrado
dm_fundos	126.08	Centro	-782.52	Curicica
dm_granito	252.00	Taua	122.81	Vargem Pequena
dm_hidrometro	122.81	Vargem Pequena	122.81	Vargem Pequena
dm_indevassavel	131.39	Freguesia (Ilha)	-339.64	Urca
dm_lateral	222.73	Vila Isabel	-325.87	Vila Valqueire
dm_linear	1614.10	Sao Cristovao	-669.01	Santa Teresa
dm_mezanino	-100.85	Vargem Pequena	-100.85	Vargem Pequena
dm_original	864.57	Taquara	-1054.50	Curicica
dm_piscina	236.00	Bonsucesso	-169.70	Bras de Pina
dm_planejad	334.46	Jardim Sulacap	-258.45	Jardim Carioca
dm_play	451.29	Urca	-210.86	Higienopolis
dm_porcelanato	674.90	Urca	25.68	Freguesia (Jacarepagua)
dm_portaria	245.28	Sao Conrado	-290.90	Bonsucesso
dm_recuado	649.33	Jacarepagua	-221.02	Vila Isabel
dm_sala_jantar	214.43	Leme	-232.92	Jardim Sulacap
dm_salao_festas	229.53	Leme	-1552.11	Urca
dm_salao_jogos	195.08	Praca da Bandeira	-482.40	Santa Teresa
dm_sauna	1300.09	Centro	-198.63	Jardim Sulacap
dm_seguranca	975.60	Urca	-494.13	Jardim Sulacap
dm_servico	239.60	Estacio	-110.38	Jardim Sulacap
dm_sol_manha	648.99	Jardim Sulacap	-440.53	Campo Grande
dm_terraco	479.40	Sao Conrado	-1904.39	Sao Cristovao
dm_triplex	814.03	Leblon	-554.02	Humaita
dm_varanda	210.26	Sao Conrado	-296.76	Leme
dm_vista_mar	343.01	Leme	-163.78	Vargem Pequena

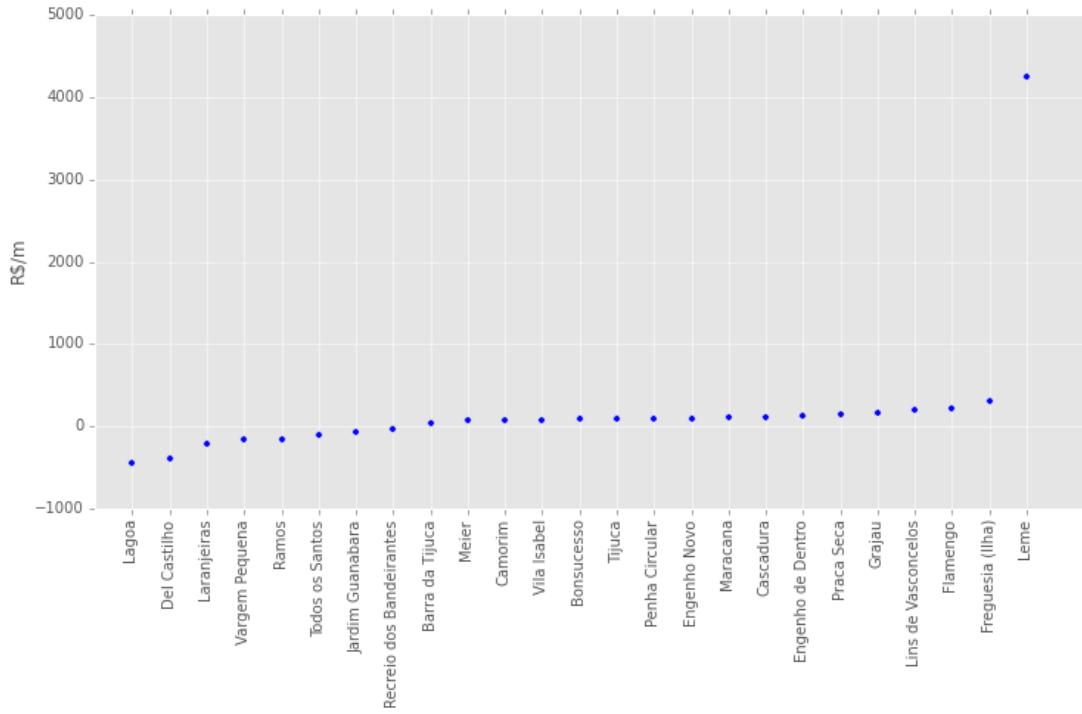
Continua na próxima página

Tabela 4.8: Máximos e mínimos das variáveis com respectivos bairros.

	Coef_max	Bairro_max	Coef_min	Bairro_min
garagem_1	368.61	Leblon	-48.12	Jardim Guanabara
garagem_2	731.90	Ipanema	-2171.15	Sao Cristovao
garagem_3	1727.10	Santa Teresa	-201.03	Praca Seca
garagem_4	1521.13	Ipanema	-1549.09	Urca
preco_lag	2.30	Riachuelo	-6.33	Rocha
quartos_2	89.83	Freguesia (Ilha)	-228.83	Urca
quartos_3	326.33	Ipanema	-308.46	Sao Conrado
quartos_4	477.05	Cosme Velho	-351.54	Sao Conrado
suites_1	307.31	Jardim Botanico	22.56	Jardim Sulacap
suites_2	629.34	Leme	-681.01	Urca
suites_3	997.55	Sao Conrado	-1818.83	Urca
suites_4	3390.31	Tijuca	-1891.82	Gavea

A variável *dist_favela* apresenta valores interessantes. O máximo é para o bairro do Leme, cuja distância largura é restrita em cerca de 350 metros entre a praia e a favela da Babilônia ao longo de seu comprimento de 1 Km. O mínimo ocorre, inesperadamente, para um bairro da "Zona Sul", Lagoa, onde cada metro de distância reduz o preço em R\$ 448,96. A figura 4.4 apresenta os valores dessa variável nos bairros em que ocorre. O bairro do Leme, que detém o segundo maior valor, apresenta semelhança geográfica com São Conrado, em que a área residencial é limitada em cerca de 120 metros ao longo de sua extensão entre o litoral e Morro da Babilônia.

Figura 4.4: Variável dist_favela em bairros



Dessa forma, verificamos que a automatização de um modelo de regressão excluindo-se variáveis apenas com base na significância estatística e multicolinearidade para comparar a importância relativa das variáveis entre os diferentes bairros demanda uma análise mais criteriosa a fim de escolher adequadamente quais das variáveis correlacionadas será eliminada do modelo.

Os bairros que não puderam ser avaliados por insuficiência de observações estão listados abaixo, onde o número entre parênteses indica a quantidade de observações.

Abolicao(42), Agua Santa(39), Alto da Boa Vista(36), Anchieta(3), Bancarios(18), Bangu(18), Barra de Guaratiba(2), Barros Filho(8), Benfica(15), Bento Ribeiro(25), Caucia(30), Caju(5), Campinho(30), Campo dos Afonsos(7), Catumbi(26), Cavalcanti(8), Cidade Nova(25), Cidade Universitaria(5), Cidade de Deus(24), Cocota(24), Coelho Neto(10), Colegio(20), Complexo do Alemao(7), Cordovil(26), Cosmos(31), Costa Barros(16), Deodoro(2), Engenho da Rainha(30), Galeao(10), Gamboa(9), Gardenia Azul(16),

Guadalupe(10), Guaratiba(15), Honorio Gurgel(6), Inhauma(32), Inhoaiba(18), Itanhangá(48), Jacare(17), Jacarezinho(4), Jardim America(13), Joa(2), Magalhaes Bastos(5), Manguinhos(3), Mare(5), Marechal Hermes(20), Maria da Graca(33), Monero(26), Osvaldo Cruz(18), Paciencia(12), Padre Miguel(7), Parada de Lucas(7), Pavuna(25), Pilares(33), Pitangueiras(34), Praia da Bandeira(36), Realengo(26), Ribeira(38), Rocha Miranda(18), Rocinha(5), Santa Cruz(35), Santissimo(6), Santo Cristo(12), Senador Ca-mara(14), Senador Vasconcelos(7), Sepetiba(2), Tomas Coelho(28), Turiacu(20), Vargem Grande(45), Vasco da Gama(12), Vaz Lobo(13), Vicente de Carvalho(38), Vidigal(17), Vigario Geral(13), Vila Kosmos(33), Vista Alegre(27), Zumbi(15)

Capítulo 5

Conclusão

Este estudo teve como objetivo principal revisitar a modelagem hedônica para cidade de Rio de Janeiro, como proposta de atualização do trabalho anterior realizado por [Neto \(2002\)](#), em função da recente importância da cidade como vitrine do país sediando os maiores eventos do esporte mundial, o que acarretou um fluxo de investimentos em infraestrutura, mobilidade urbana e segurança pública, que aliado com um maior acesso ao crédito imobiliário promovido pelo Governo Federal, gerou perceptível impacto no mercado de imóveis da cidade.

A modelagem hedônica é uma técnica da Econométrica para a avaliação da precificação de um bem de consumo como uma combinação linear dos coeficientes de suas partes constituintes inferidos a partir de uma regressão linear onde o preço do bem é a variável dependente e suas partes constituintes as variáveis independentes. A importância das partes constituintes é considerada em função do valor dos respectivos coeficientes das variáveis independentes resultantes da regressão linear. A utilização de regressão linear demanda atenção a certas premissas à sua utilização como não colinearidade entre as variáveis independentes, independência e homoscedasticidade dos erros residuais. Destas premissas, as duas últimas são as que mais merecem cuidado pois é verificado que os preços de imóveis não são determinados por processos inteiramente independentes

em que se consideram apenas as variáveis explicativas, mas sofrem influência dos preços dos imóveis vizinhos em função de sua localização próximos a pontos de interesse seja positivo ao preço, como proximidade a centros de trabalho, acesso de modais de transporte rápido, áreas de cultura e lazer, seja negativo, como proximidade a áreas degradadas, sensíveis a segurança pública, difícil acesso ou poluição. Pela influência dos vizinhos, temos então uma autocorrelação da variável dependente e, devido a essa influência ocorrer em função da localização, denomina-se autocorrelação espacial.

Nesse trabalho, uma primeira abordagem para a mitigação da autocorrelação espacial da variável preço foi a introdução de variáveis de acessibilidade, que representam distâncias do imóvel aos pontos de interesse, e variáveis socioambientais, que refletem as características regionais compartilhadas entre imóveis vizinhos. Uma segunda abordagem foi a introdução de uma nova variável que representa a influência dos imóveis vizinhos, modelada como sendo a média dos preços dos 100 imóveis mais próximos.

O conjunto de dados foi construído a partir de um *web scrapping* de classificados na Internet ZAP Imóveis, do qual inicialmente capturamos 91.091 anúncios de apartamentos usados na cidade do Rio de Janeiro publicados entre outubro de 2013 e outubro de 2014. Após tratamento de *outliers* e repetições, o conjunto de observações final resultou em 28.111 imóveis. As informações básicas como quantidade de cômodos, vagas de garagem e preço estavam facilmente disponíveis na estrutura HTML do anúncio, mas as demais informações foram capturadas com o processamento de texto da descrição do imóvel feita pelo anunciante. Fontes de dados oficiais como o Armazém de Dados da Prefeitura do Rio de Janeiro e Instituto de Segurança Pública do Estado para a construção das demais variáveis espaciais e socioambientais, mas alguns pontos de interesse como praia foram obtidos de uma base de dados de acesso público conhecida como OpenStreetMap.

O principal resultado da modelagem hedônica para os imóveis da cidade foi a verificação da grande influência dos imóveis vizinhos para a formação do preço, seguida

da estrutura do apartamento em termos de área, quantidade de suítes, quartos e vagas de garagem, após a análise de 3 modelos apresentados. A utilização de diversas variáveis de acessibilidade e socioambientais como forma de mitigar os efeitos da autocorrelação espacial não foi suficiente para a explicação completa da variável *preco*, como pode ser visto na lista de variáveis resultantes do modelo com *lag* espacial, tabela 4.5, p. 76, que contém apenas cinco das treze variáveis de acessibilidade inicialmente propostas e uma socioambiental. Sobre os resultados auxiliares, a análise descritiva das variáveis independentes modeladas permite-nos conhecer seus domínios de valores de forma a auxiliar uma interpretação mais adequada dos coeficientes resultantes da regressão. Não obstante, nosso conjunto de dados apresenta 75% dos imóveis anunciados a uma distância máxima de 18 Km do centro da cidade. Outro resultado auxiliar foi a verificação do aumento do preço à medida que os imóveis distanciam-se de favelas quando essa variável é a única do modelo, mas quando em conjunto com as demais, não apresenta o mesmo impacto. Finalmente, as análises específicas a bairros resumidas na tabela 4.6 mostra-nos a distribuição de ocorrência das variáveis independentes estatisticamente significantes nos resultados finais das regressões, onde verificamos que as variáveis mais comuns entre os bairros são as relacionadas a estrutura do imóvel e distância a pontos de interesse. Entretanto, alertamos que uma correta abordagem da relevância de variáveis entre os bairros fomenta uma análise específica considerando as particularidades de cada um.

5.1 Limitações e possíveis extensões

A principal fonte de dados para a captura das informações dos imóveis foi o classificados online ZAP Imóveis, utilizando a técnica de *web scrapping* processando em memória as páginas dos anúncios a medida que eram baixadas. Uma das principais desvantagens dessa técnica é sua total dependência com a estrutura dos dados onde residem as informações de interesse. Qualquer alteração na estrutura original demanda uma

reavaliação do algoritmo de captura para adaptação das modificações realizadas na origem. De fato, o serviço ZAP Imóveis sofreu uma atualização de sua interface com os usuários e alterou drasticamente a estrutura das páginas HTML onde as informações estão armazenadas. Desta forma, o algoritmo de captura listado no appendix B, p. 111 está obsoleto e precisa ser adaptado à nova disposição das informações.

O presente estudo considerou apenas imóveis usados. Essa decisão foi tomada com o objetivo de obter um grande número de observações, dado que a quantidade de imóveis já ocupados colocados à venda é superior à quantidade de imóveis em lançamento. Entretanto, cuidando-se para que o número de variáveis independentes não supere o número de observações, a modelagem abordada nesse estudo é inteiramente compatível para os novos lançamentos, cabendo-se atualizar o algoritmo de captura e realizar nova busca no ZAP Imóveis usando como URL inicial a lista de imóveis em lançamento na cidade. Pode-se também adicionar uma variável indicativa do estado do imóvel, novo ou usado, a fim de verificar a contribuição dessa característica à formação do preço.

Por descuido na construção da variável *dist_saude_privada* acabamos utilizamos todos os estabelecimentos de saúde privada fornecidos no Portal de Dados Abertos da Prefeitura da Cidade do Rio de Janeiro. Entretanto, essa fonte de dados permite a identificação de estabelecimentos de saúde com respeito a natureza dos seus serviços oferecidos à população como hospital geral ou especializado, consultório médico, pronto socorro e laboratório. Uma filtragem dessa informação pode permitir uma avaliação mais apurada da relevância dessa distância com base na natureza do estabelecimento.

As variáveis socioambientais utilizadas, com exceção das duas a respeito de criminalidade, foram tomadas do Armazém de Dados da Prefeitura do Rio de Janeiro, mantido pelo Instituto Pereira Passos, e a menor agregação regional disponível foi a nível de bairro, sendo dados derivados dos Censos realizados pelo Governo Federal. Dessa forma, essas variáveis são replicadas para os imóveis de um mesmo bairro o que as torna variáveis inviáveis para a avaliação de modelos delimitados a bairro. Outra desvantagem

observada é que as variáveis construídas com base nessas informações apresentam baixa variabilidade, carregando pouco poder explicativo para a variável dependente *preco*. As variáveis sociambientais a respeito de criminalidade foram obtidas do Instituto de Segurança Pública do Estado do Rio de Janeiro, e seu nível de agregação por AISP também é limitado a nível de bairro.

Cerca de 22.000 anúncios não foram utilizados devido a suas coordenadas serem repetidas. Verificamos que na grande maioria dos casos os imóveis possuem ruas diferentes, o que determina que suas coordenadas deveriam ser diferentes. Uma solução a alternativa de ignorá-los é descobrir suas coordenadas com base no endereço, um método conhecido como geolocalização e disponível em Python pela biblioteca Geopy¹, e calcular as variáveis de acessibilidade com base na nova coordenada. Possivelmente encontrar-se-á a repetição de coordenadas para imóveis de uma mesma rua. Pode-se distribuir esses imóveis ao longo da rua interpolando-se as coordenadas. Caso decida-se manter as coordenadas iguais, alertamos que isso impede a construção da matriz de pesos espaciais W com base no inverso da distância.

Por último, uma aplicação prática da utilização da modelagem hedônica é a construção de um produto de software que permita ao usuário, por exemplo um agente público interessado em verificar o mercado imobiliário, ao escolher um ponto de interesse em uma interface do mapa da cidade do Rio de Janeiro, obter diversas inferências do preço com base nas características definidas. Uma inferência seria com base no modelo para toda a cidade, outra com base no modelo do bairro, outra a partir de um modelo limitado a uma área espacial com raio configurável, e um último como a média dos últimos preços determinados.

¹Biblioteca Geopy: <https://github.com/geopy/geopy>.

5.2 Últimas palavras

Nas palavras de George Box, "*all models are wrong, but some are useful*", o que nos leva a considerar, com base nos esforço investido na elaboração desse estudo, avaliação da literatura pertinente e verificação dos exemplos construídos, que não haverá um modelo único que possa seguramente representar toda a complexidade da realidade que, embora não exclusiva, é peculiar à cidade do Rio de Janeiro pelo contexto histórico de sua ocupação e as limitações que os diversos acidentes geográficos impõem ao planejamento urbano e acesso a centros de emprego, comércio e entretenimento.

A realização desse trabalho permitiu uma revisão da modelagem hedônica para a cidade do Rio de Janeiro em uma tentativa de melhorar a inferência anterior, [Neto \(2002\)](#), com um conjunto maior de variáveis e observações aproveitando-se o estágio atual de acessibilidade a tecnologias e fontes de dados não disponíveis àquela época, entretanto, terminamos esse trabalho com a constatação de que a localização ainda é o maior fator na formação de preço da venda de um imóvel usado, um processo inteiramente humano de interesse da Economia, em que o proprietário de um bem procura valorizá-lo conforme avaliação de exemplos similares próximos.

Referências Bibliográficas

- Andersen, P. K. e Skovgaard, L. T. (2010). *Regression with Linear Predictors*. Springer.
- Baltagi, B. H. (2008). *Econometrics*. Springer, 4th edition.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- de Castro Pereira, M. F. (2009). A evolução habitacional-urbana na cidade do rio de janeiro. *Revista em Debate*.
- Favero, L. P. L., Belfiore, P. P., e de Lima, G. A. S. F. (2008). Modelos de precificação hedônica de imóveis residenciais na região metropolitana de são paulo: uma abordagem sob as perspectivas da demanda e da oferta. *Estudos Econômicos.[online]*, 38(1):73–96.
- Hastie, T., Tibshirani, R., e Friedman, J. (2013). *The Elements of Statistical Learning*. Springer, 2nd edition.
- Ismail, S. (2006). Spatial autocorrelation and real estate studies: A literature review. *Malaysian Journal of Real Estate*, 1(1):1–13.
- Ismail, S. e MacGregor, B. (2006). Hedonic modelling of housing markets using geographical information system and spatial statistics: Glasgow, scotland. Em *International Real Estate Symposium (IRERS) 2006, INSPEÑ 11-13 April 2006 PWTC, Kuala Lumpur*.

- Long, F., Páez, A., e Farber, S. (2007). Spatial effects in hedonic price estimation: A case study in the city of toronto. *Working Paper Series*.
- Macedo, P. B. R. (1999). Hedonic price models with spatial effects: an application to the housing market of belo horizonte, brazil. *XIV Latin American Meeting of the Econometric Society*.
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, 1st edition.
- Murphy, K. P. (2012). *Machine Learning, a probabilistic perspective*. Massachusetts Institute of Technology.
- Neto, E. F. (2002). Estimação do preço hedônico: uma aplicação para o mercado imobiliário da cidade do rio de janeiro.
- Rey, S. e Anselin, L. (2010). Pysal: A python library of spatial analytical methods. Em Fischer, M. M. e Getis, A., editores, *Handbook of Applied Spatial Analysis*, pgs. 175–193. Springer Berlin Heidelberg.
- Rubens A. Dantas, André M. Magalhães, J. R. d. O. V. (2007). Avaliação de imóveis: a importância dos vizinhos no caso de recife. *Economia Aplicada*, 11:231 – 251.
- Schultz, K. (2008). Perfeita civilização: a transferência da corte, a escravidão e o desejo de metropolizar uma capital colonial, rio de janeiro, 1808-1821. *Tempo*, 12:5 – 27.

Apêndice A

Mapas da cidade do Rio de Janeiro

Município do Rio de Janeiro - Divisões Administrativa

Figura A.1: Mapa das divisões administrativas da cidade do Rio de Janeiro. Fonte: Instituto Pereira Passos, 2008.

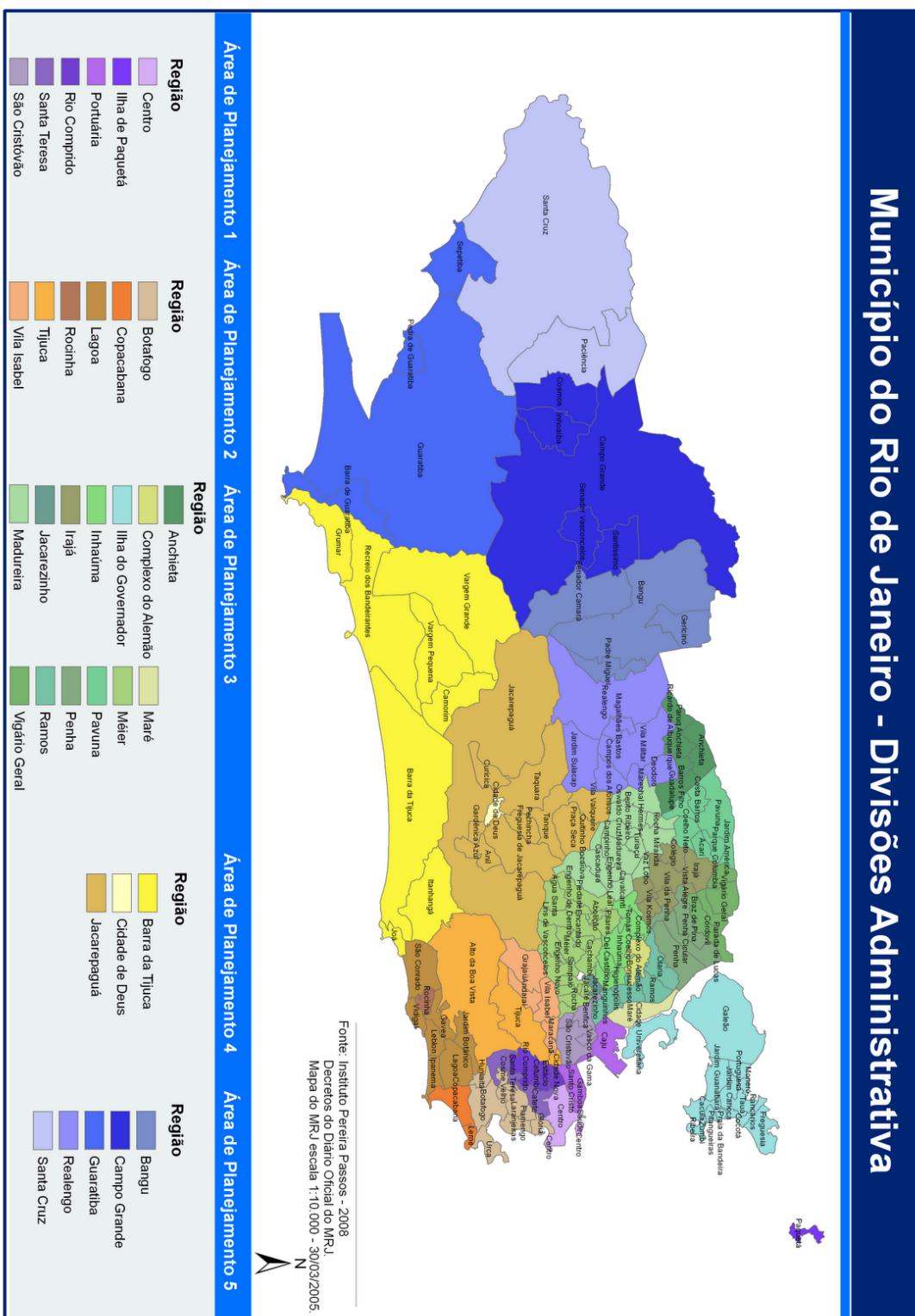
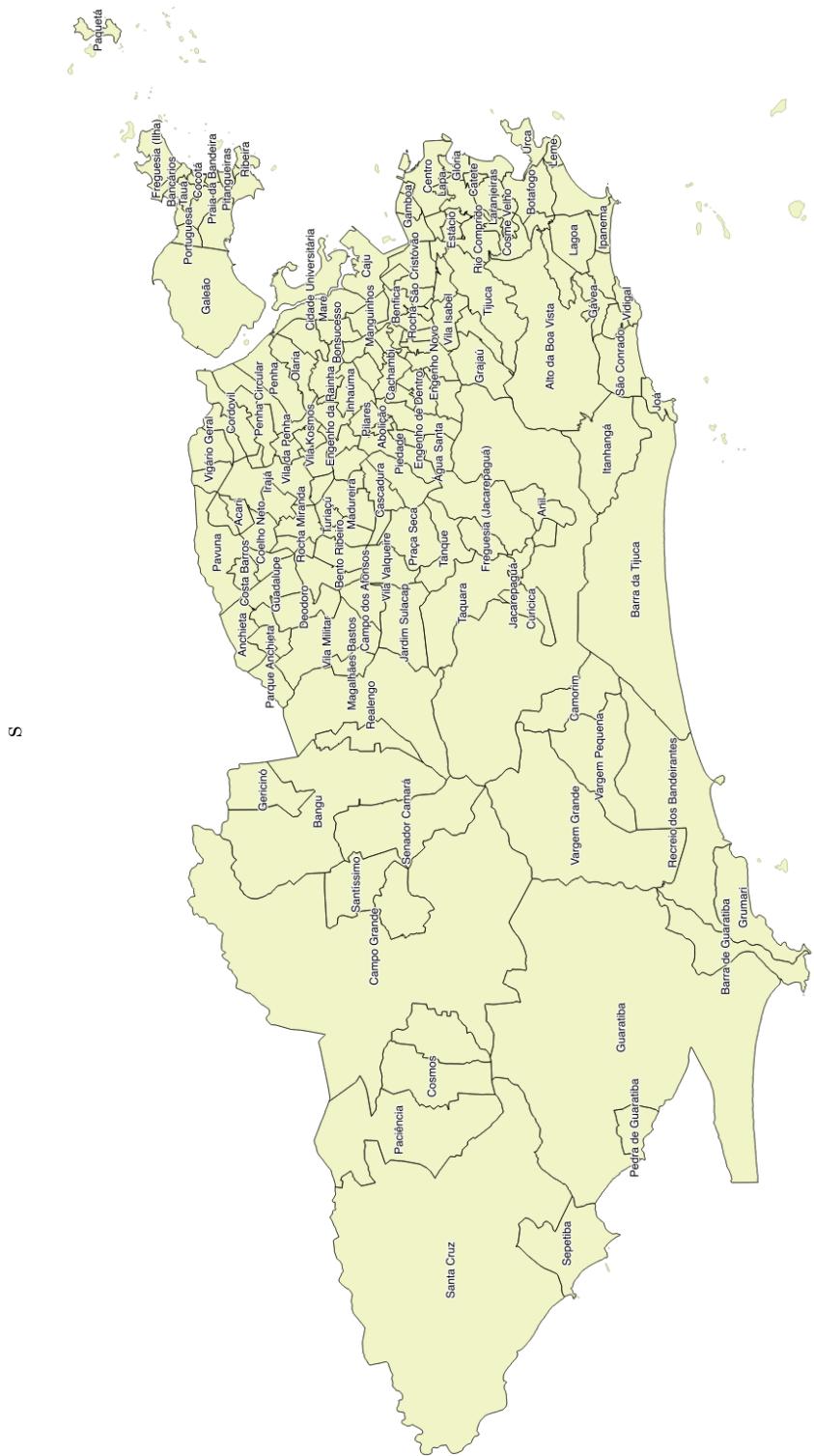


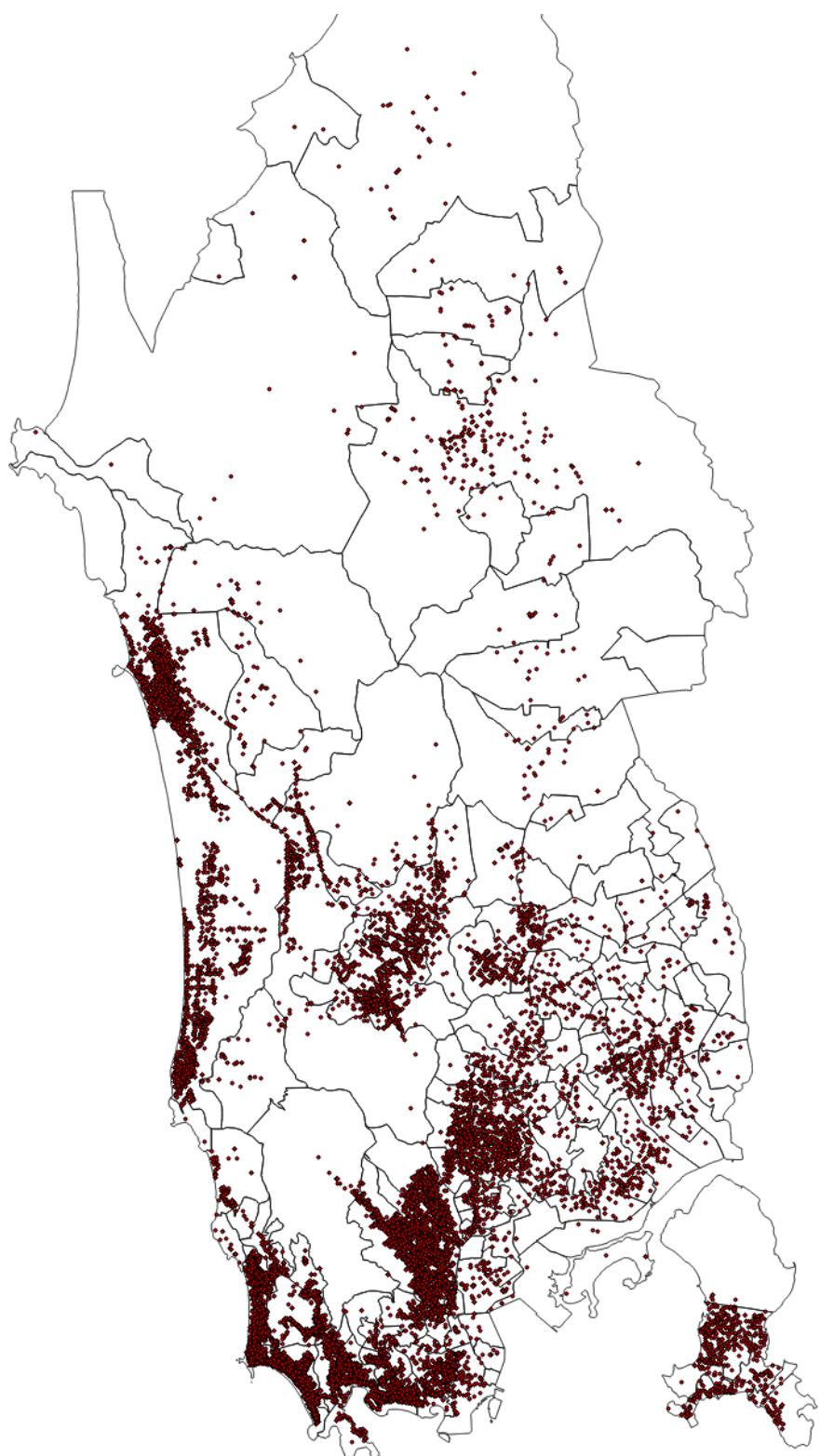
Figura A.2: Nomes dos bairros. Fonte: Mapa Digital do Rio de Janeiro [a](#).



^aMapa Digital do Rio de Janeiro: http://portalgéo.rio.rj.gov.br/mapa_digital_rio/?config=config=ipp/basegeoeweb.xml. Acessado em 5 de janeiro de 2015.

Figura A.3: Localização dos imóveis. Fonte: ZAP Imóveis^a.

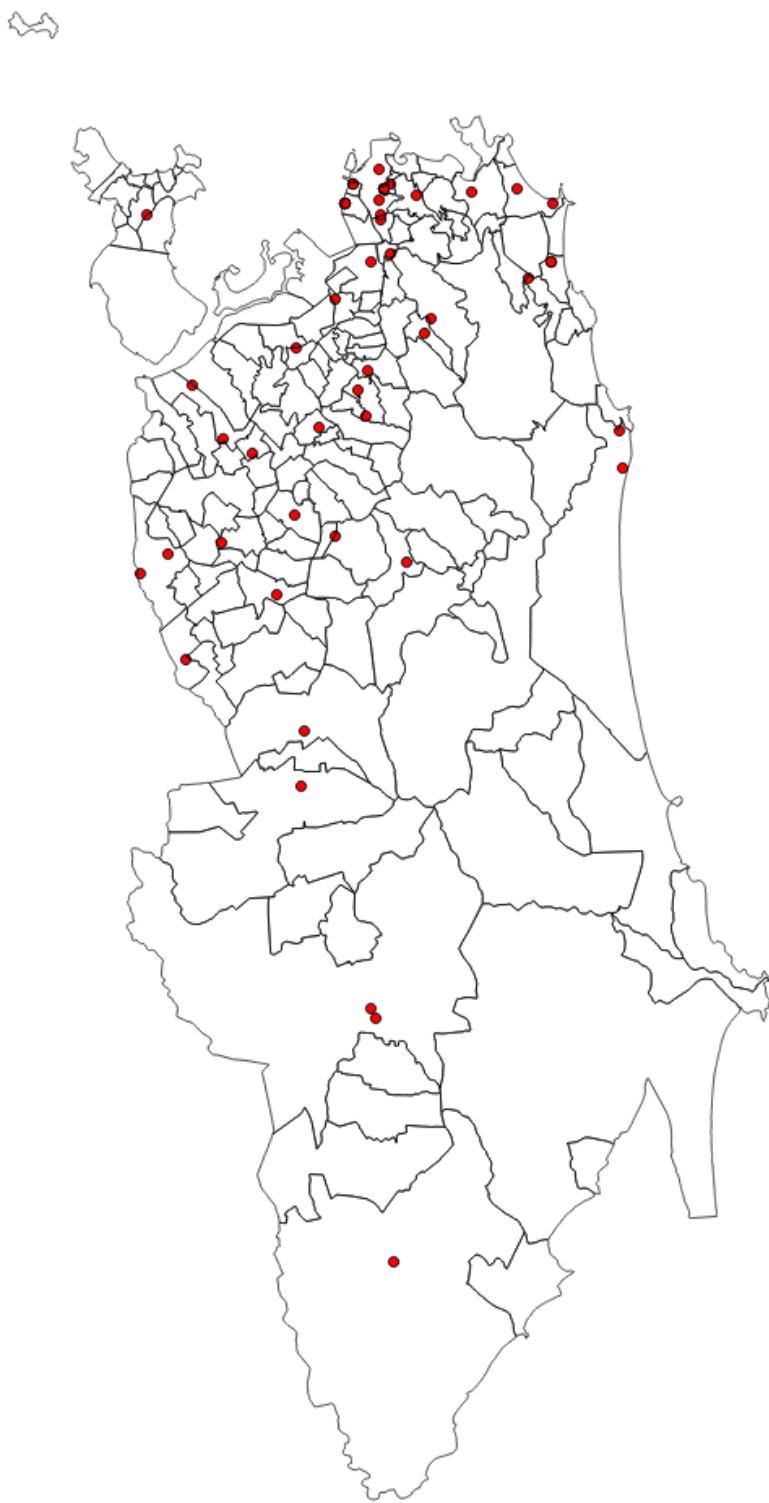
s



^aZAP Imóveis: <http://www.zapimoveis.com.br/>. Acessado entre outubro de 2013 a outubro de 2014.

Figura A.4: Localização das Delegacias de Polícia Civil.Fonte: Mapa Digital do Rio de Janeiro ^a.

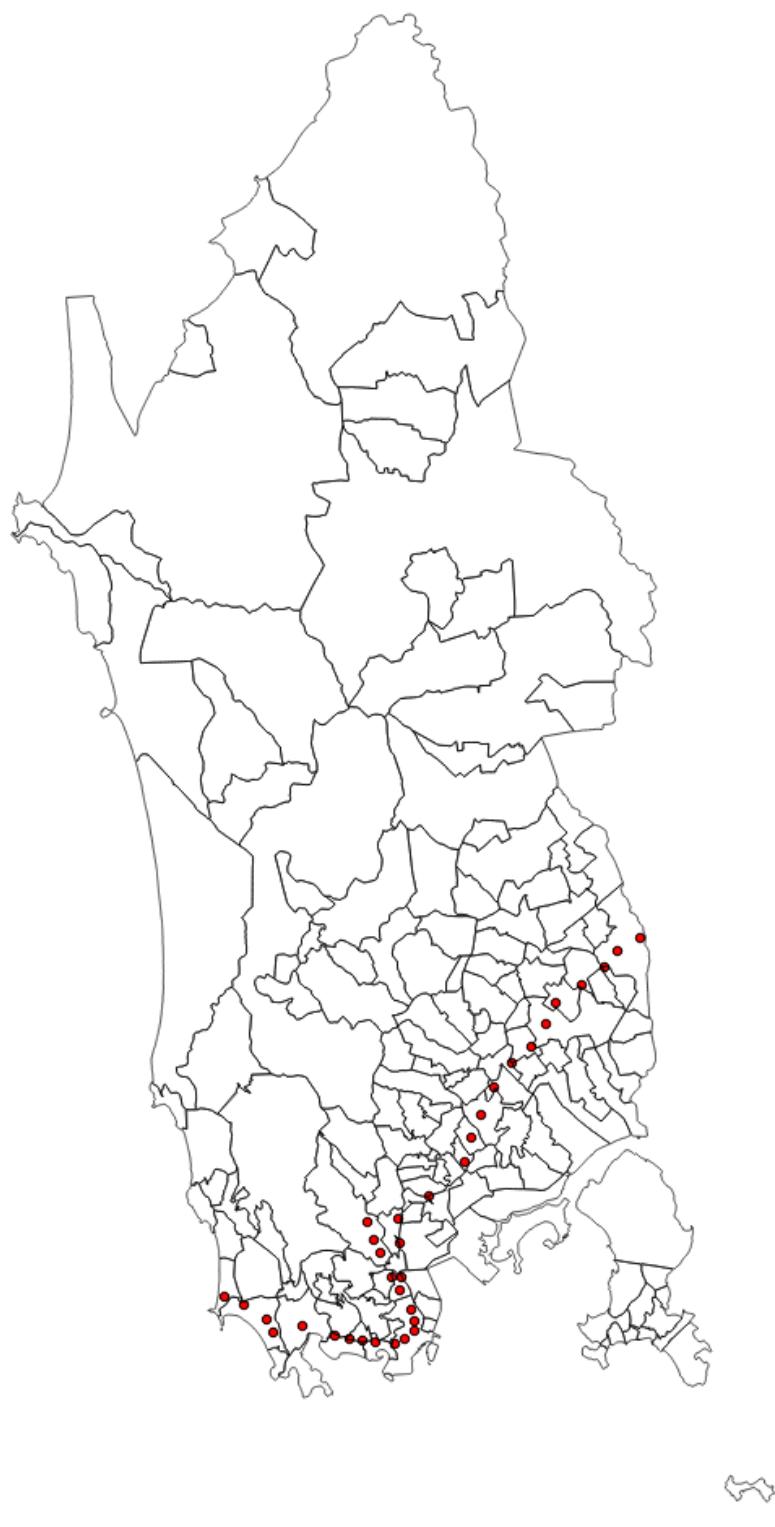
S



^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config=ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

Figura A.5: Localização das estações do metrô. Fonte: Mapa Digital do Rio de Janeiro^a.

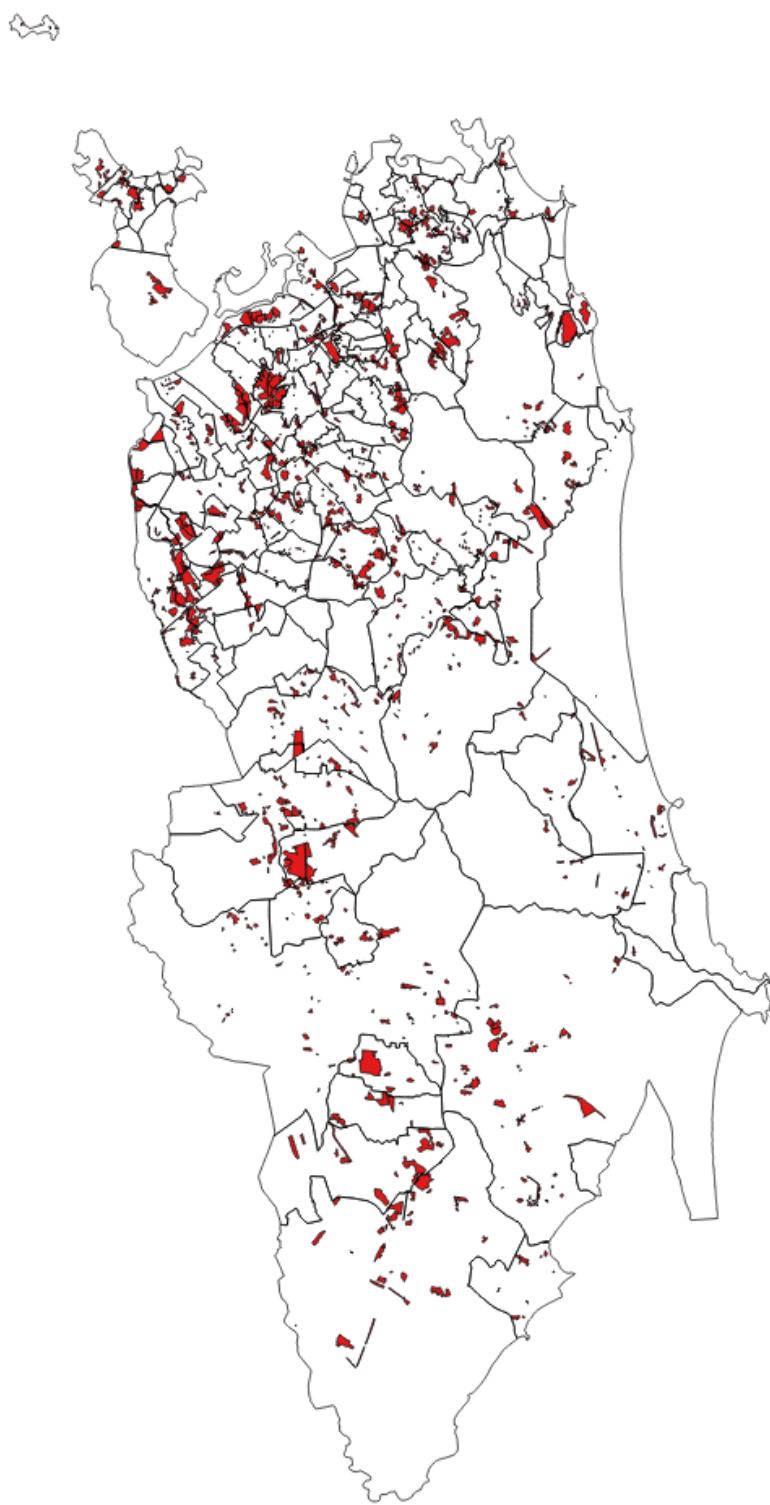
S



^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml. Acessado em 5 de janeiro de 2015.

Figura A.6: Localização dos aglomerados subnormais, favelas. Fonte: Mapa Digital do Rio de Janeiro.^a.

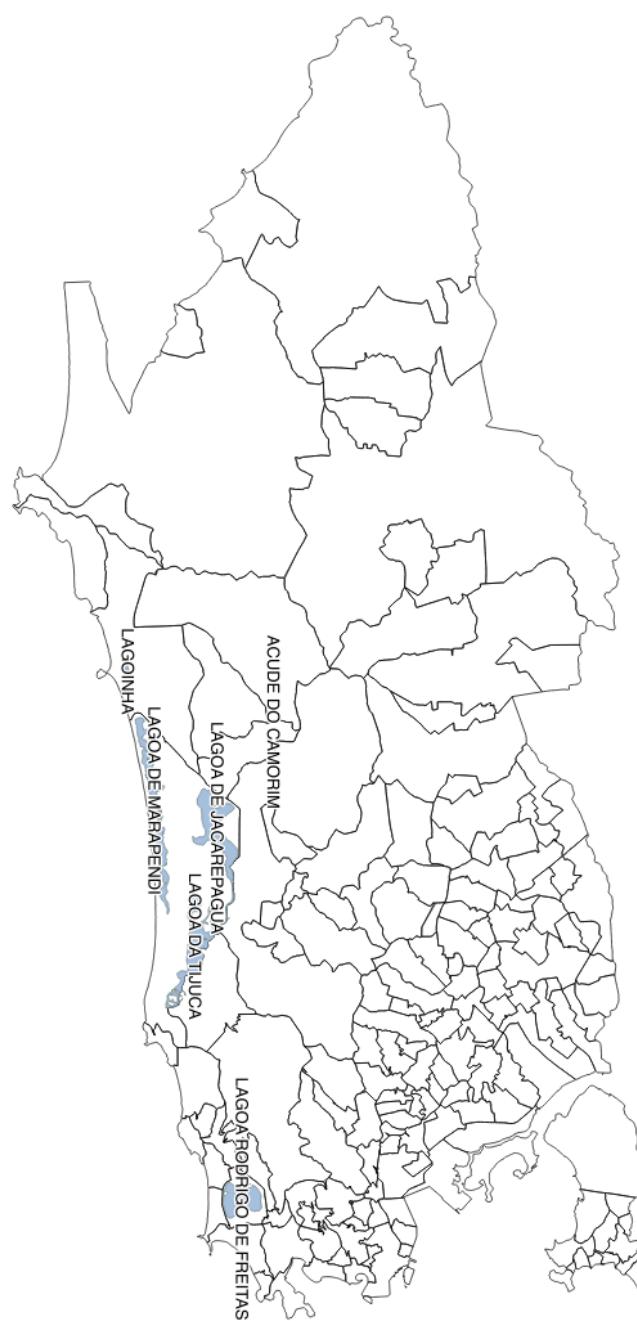
S



^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config=ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

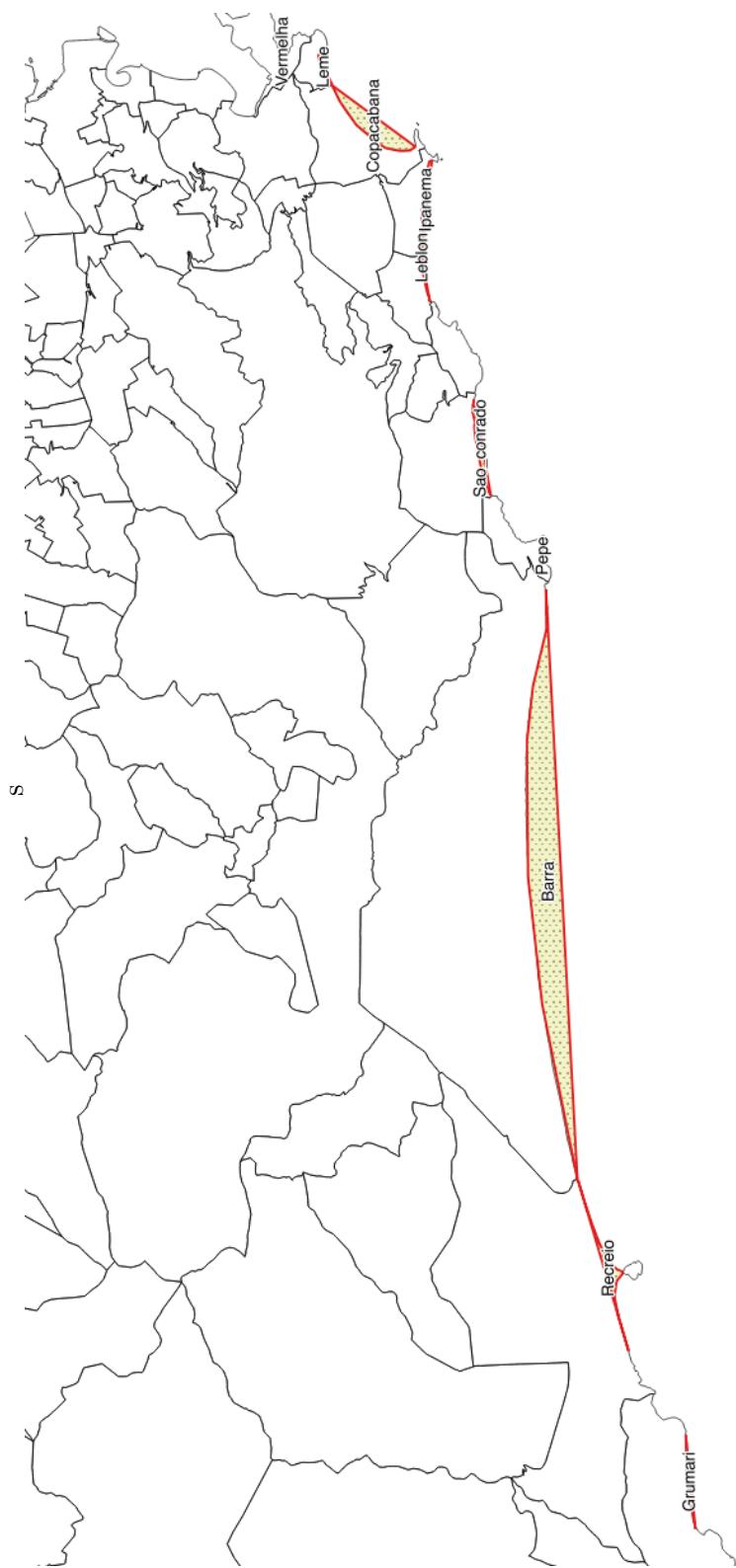
Figura A.7: Localização das lagoas.Fonte: Mapa Digital do Rio de Janeiro^a.

S



^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

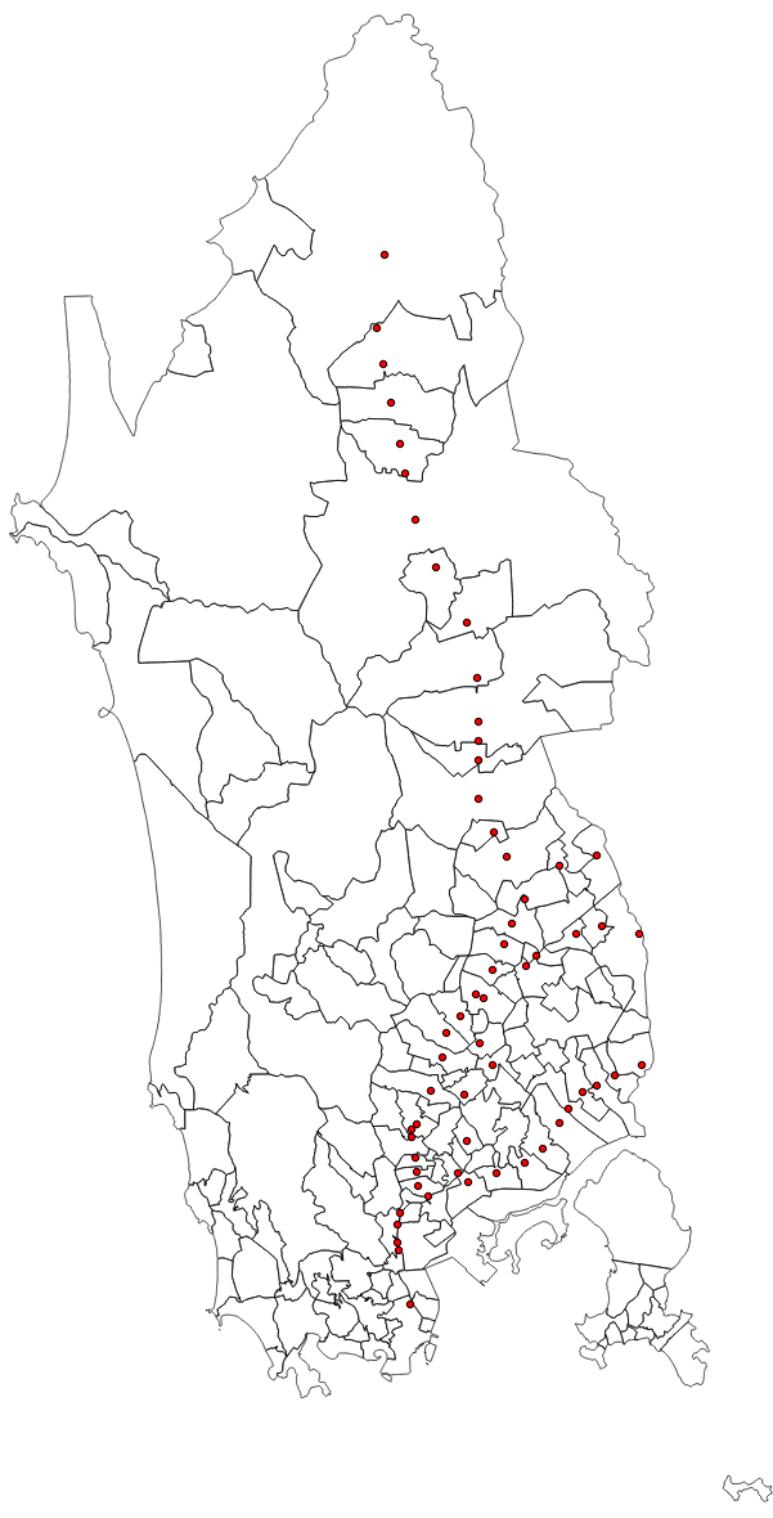
Figura A.8: Localização das praias.Fonte: OpenStreetMap^a.



^aOpenStreetMap: <http://www.openstreetmap.org>. Acessado em 6 de janeiro de 2015.

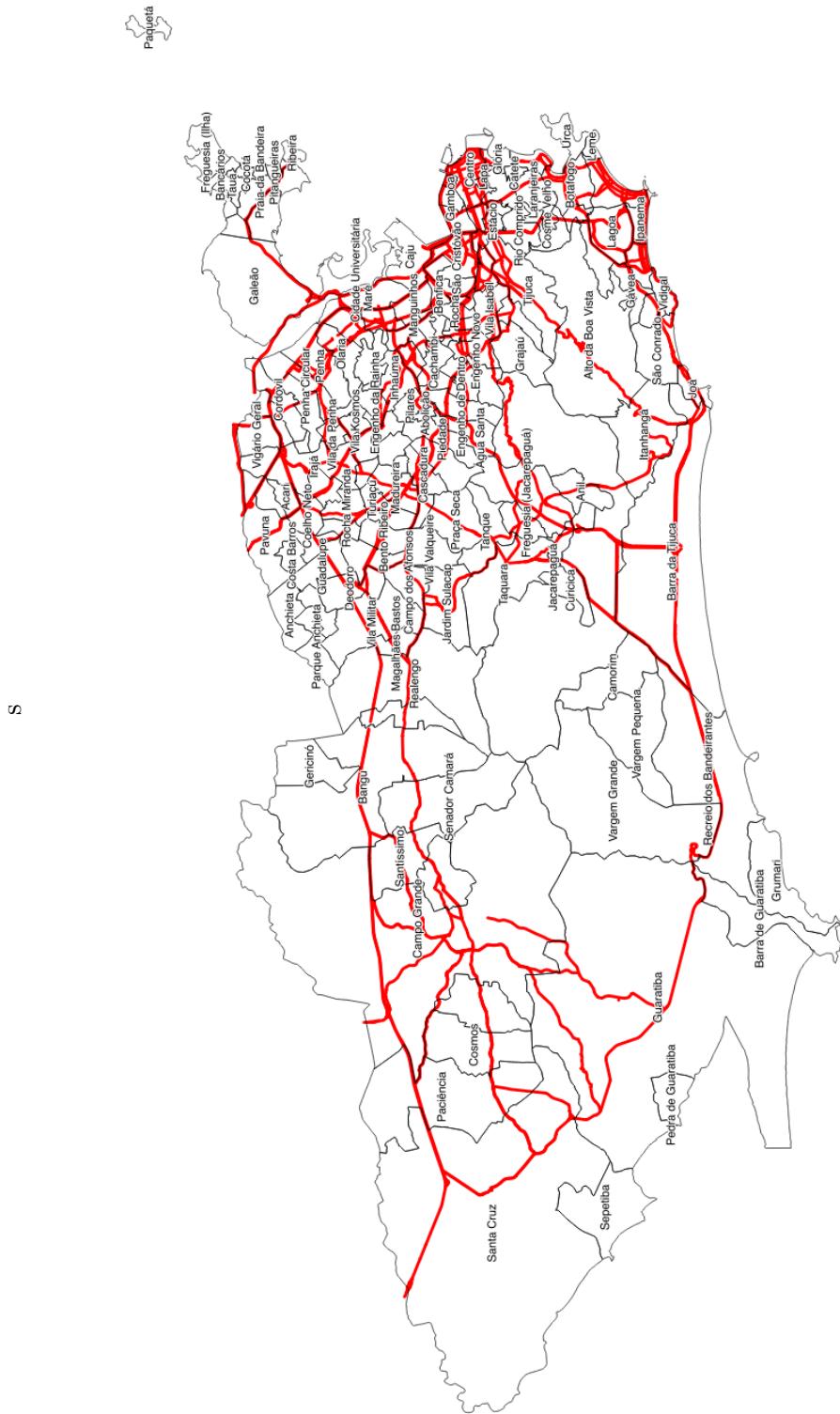
Figura A.9: Localização das estações de trem. Fonte: Mapa Digital do Rio de Janeiro^a.

S



^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

Figura A.10: Geometria dos principais logradouros da Cidade do Rio de Janeiro.^a



^aMapa Digital do Rio de Janeiro: http://portalgéo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml. Acessado em 5 de janeiro de 2015.

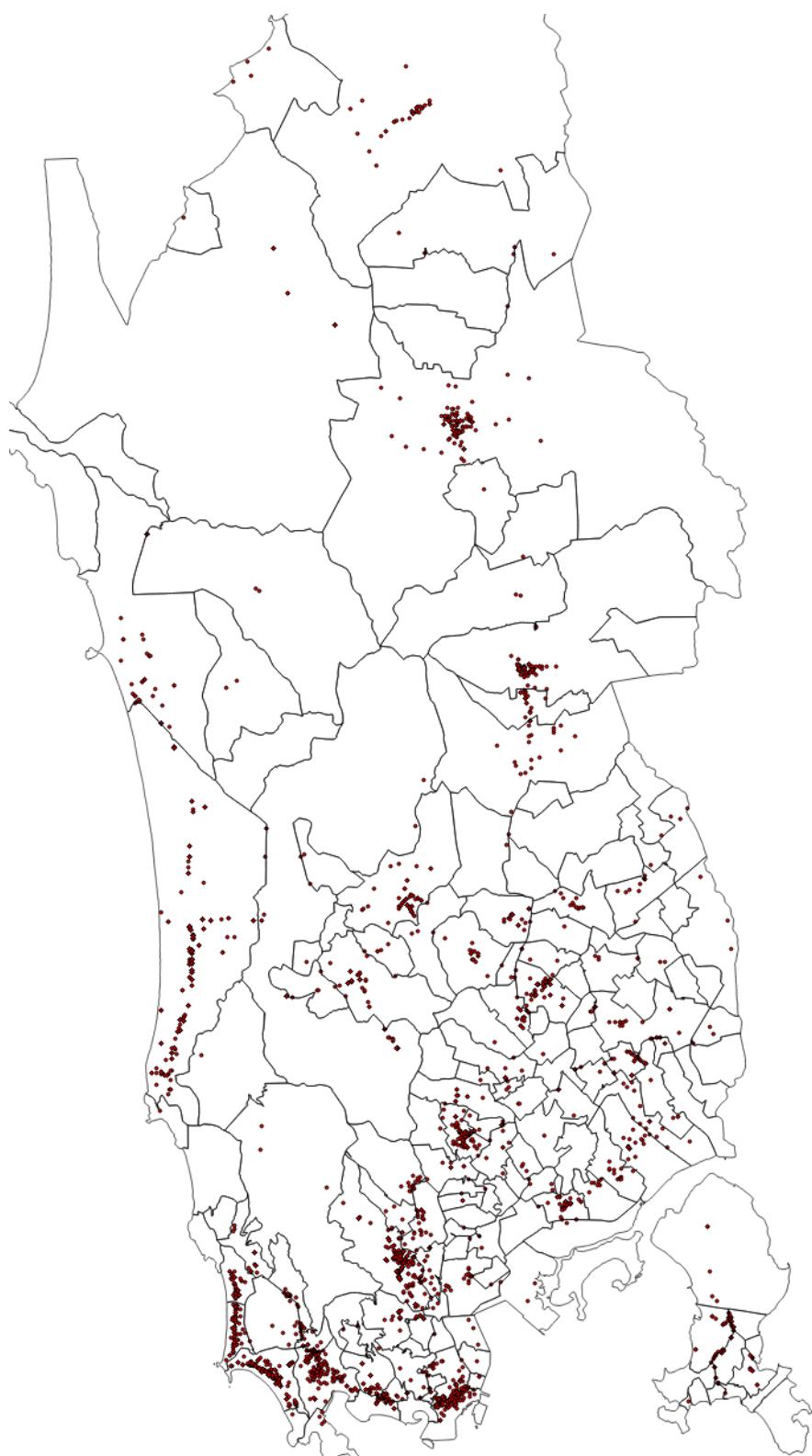
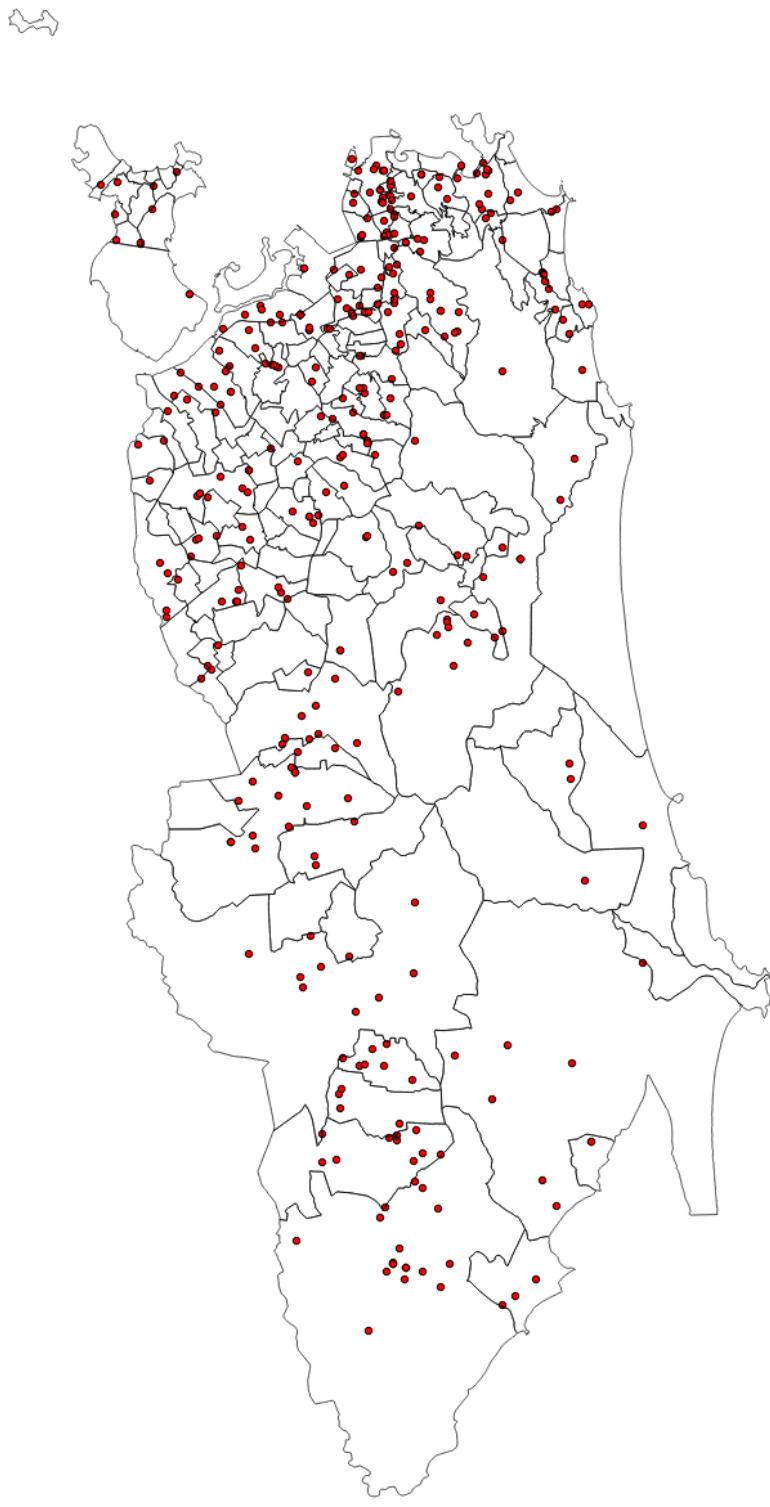


Figura A.11: Localização das unidades de saúde privada. Fonte: Mapa Digital do Rio de Janeiro [a](#).

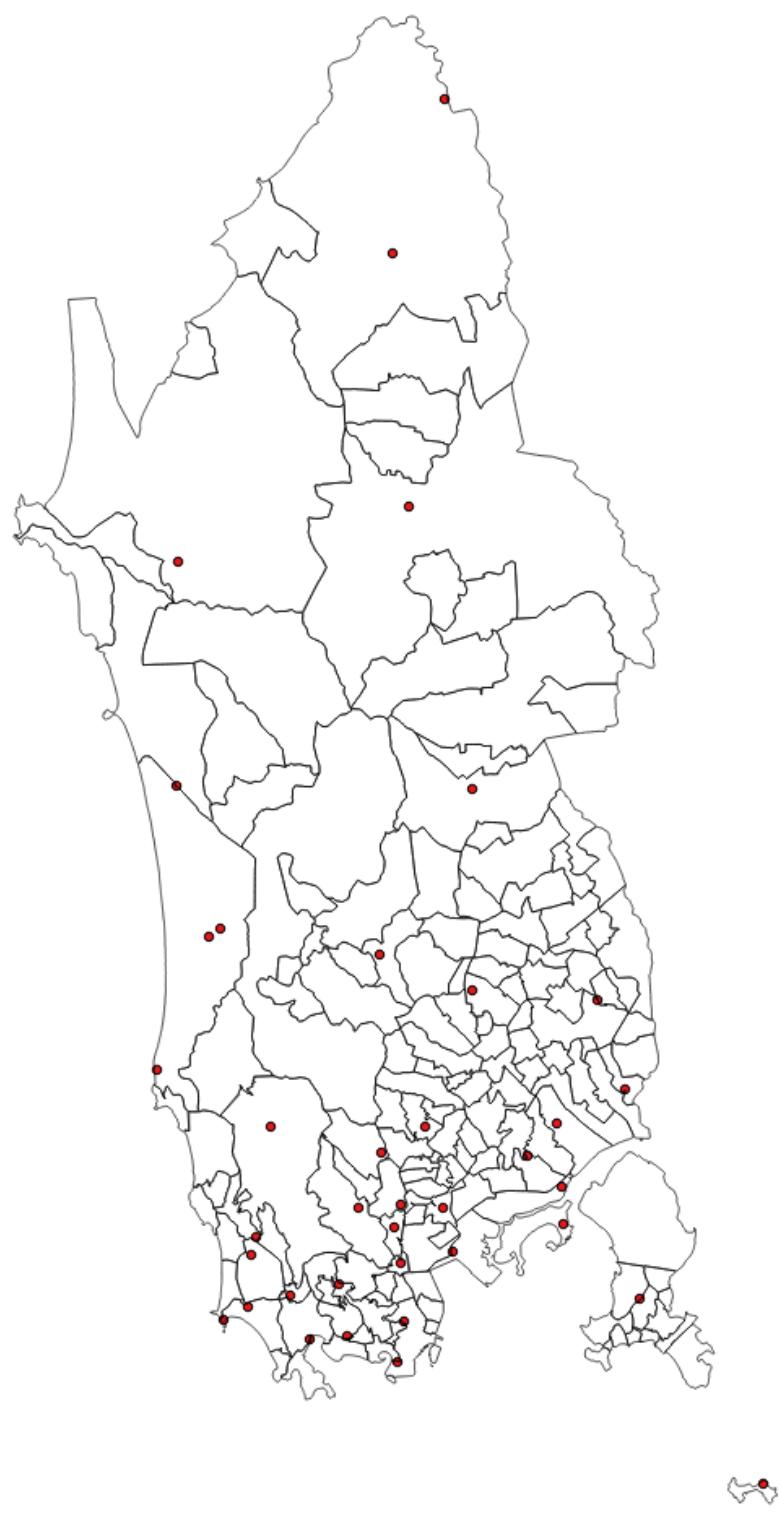
^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

Figura A.12: Localização das unidades de saúde pública.Fonte: Mapa Digital do Rio de Janeiro ^a.



^aMapa Digital do Rio de Janeiro: http://portalego.rio.rj.gov.br/mapa_digital_rio/?config=config=ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

Figura A.13: Localização das unidades do Corpo de Bombeiros. Fonte: Mapa Digital do Rio de Janeiro ^a.



^aMapa Digital do Rio de Janeiro: http://portalgeo.rio.rj.gov.br/mapa_digital_rio/?config=config/ipp/basegeoweb.xml.
Acessado em 5 de janeiro de 2015.

Apêndice B

Listagem do módulo captura.py

Código fonte do módulo de captura das informações do ZAP Imóveis.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fqv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

```

# -*- coding: utf-8 -*-

"""
Created on 06/05/2014

@author: sergio
"""

import re
import requests
from bs4 import BeautifulSoup
import thread
import dataset as d
from zap_logging import log
import threading

def __processa_html(html):
    """
    Retorna objeto Beautiful Soup que processa o conteudo do parâmetro "html".
    @param html - Conteudo HTML.
    """
    return BeautifulSoup(html, 'html.parser')

def __extrair_minificha(div):
    """
    Retorna informacoes de um apartamento de um elemento DIV da classe
    'content-minificha' de uma pagina HTML de anuncios do ZAP Imoveis.
    @param div - Elemento DIV da pagina de classe CSS 'content-minificha' que
    contem informacoes de um apartamento.
    """

apartamento = {}

# Obtem url do anuncio especifico do imovel.
apartamento['url'] = div.a['href']

# Obtm url da foto anunciada do imovel.
apartamento['url_foto'] = div.a.img['src']

# Neighborhood, city, state and price tags.
span_location = div.find('span', class_='location')
sublocal = span_location.findAll('a')

# Indexadres de tags.
IDX_NEIGH = 0
IDX_CITY = 1
IDX_STATE = 2
IDX_PRICE = 1

# Obtem bairro.
apartamento['bairro'] = sublocal[IDX_NEIGH].string

# Obtem cidade do imovel.
apartamento['cidade'] = sublocal[IDX_CITY].string

# Obtem estado do imovel.
apartamento['estado'] = sublocal[IDX_STATE].string

# Obtem preco do imovel.
apartamento['preco'] = div.find('a', class_='valorOferta').contents[IDX_PRICE]

details = div.find('a', class_='itemCaracteristicas')
list_details = [s for s in details.strings]
for i in range(len(list_details)):
    # Area.
    if list_details[i] == u'Area':
        area = list_details[i+1]
        apartamento['area'] = re.findall(r'\d+', area)[0]

    # Garagem.
    if list_details[i] == u'Vagas':
        apartamento['garagem'] = list_details[i+1]

    # Suites.
    if list_details[i] == u'Suites':

```

```

apartamento['suites'] = list_details[i+1]

# Quartos.
if list_details[i] == u'Dorms':
    apartamento['quartos'] = list_details[i+1]

# Data de publicacao.
span_date = div.find('span', class_='dtlisted')
apartamento['dt_publicacao'] = re.findall(r'(\d+/\d+/\d+)', span_date.string)[0]

# Extrai codigo identificador do imovel.
id = re.findall(r'ID-\d+',apartamento['url'])[0]
apartamento['id'] = re.findall(r'\d+',id)[0]

# Retorna as informacoes do imovel.
return apartamento

def __processar_pagina(url):
    """
    Processa uma pagina de anuncios do ZAP Imoveis e retorna uma lista de
    apartamentos e lista de links para outras paginas de anuncios.

    @param url - Link URL de uma pagina de anuncios do ZAP Imoveis a ser
                 processada.
    ...
    apartamentos,paginas = [],[]
    response = None

    response = requests.get(url).text
    doc = __processa_html(response)

    divs = doc.findAll('div', class_='content-minificha' )
    for div in divs:
        apartamentos.append(__extrair_minificha(div))
    next_pages = doc.findAll('a', class_='paginacao_link')
    for next_page in next_pages:
        if 'ativo' in next_page['class']:
            continue
        url = next_page['href']

        # Remove 'request number' from url.
        if url.find('&rn')>0:
            url = url.split('&rn')[0]
        if url != None:
            paginas.append({'url':url})

    if response != None:
        return apartamentos,paginas
    return None

def captura_apartamentos():
    """
    Executa a captura online dos imoveis no classificados ZAP a partir de uma
    pagina inicial cadastrada no banco de dados.
    """

    log.info('Iniciando captura de apartamentos do zap. Veja log em "log.txt".')
    paginas = d.paginas_nao_visitadas()

    # Executa o loop enquanto houver paginas a visitar.
    while len(paginas) > 0:
        try:
            url = paginas[0]['url']
            log.info('Processando pagina "{}".format(url)')
            apartamentos,paginas = __processar_pagina(url)
            log.info('Pagina "{}" processada.'.format(url))

            d.insere_paginas(paginas)
            d.insere_imoveis(apartamentos)
            d.registra_visita_pagina(url)
        except Exception as e:
            log.error(e)
        finally:
            paginas = d.paginas_nao_visitadas()
    log.info('Nao ha paginas a visitar. Captura de apartamentos terminado.')

```

```

def __captura_thread(apts,tmp=None):
    """
    Metodo interno para captura dos detalhes dos imoveis em modo Thread.

    @param apts: lista com informacoes de imoveis.
    """

    log.debug('Tread com {} apartamentos criada.'.format(len(apts)) )
    for ap in apts:
        try:
            response = requests.get(ap['url'])
        except Exception as e:
            log.error('Erro apt ID = {}. Msg: {}'.format(ap['id'],e))
            #d.insere_erro(ap['id'],e)
            continue

        text = response.text
        response.close()

        if text != None:
            try:
                doc = __processa_html(text)
                det = extrai_detalhes(doc)
                det['id'] = ap['id']
                det['dt_visita'] = d.dia_hoje()
                d.atualiza_imoveis([det])
            except Exception as e:
                log.error('Erro apt ID = {}. Msg: {}'.format(ap['id'],e))
                d.insere_erro(ap['id'],e)

    def captura_detalhes_thread():
        """
        Cria e executa três threads para captura das informacoes de imoveis do
        ZAP Imoveis.
        """

        apts = d.imoveis_nao_visitados()
        qtd = len(apts)
        log.debug('Capturando detalhes de {} apartamentos por TRHREAD.'.format(len(apts)) )
        #thread.start_new_thread(__captura_thread, (1,))
        th1=threading.Thread( target=__captura_thread, args = ( apts[:qtd/3], ) )
        th1.start()
        th2=threading.Thread( target=__captura_thread, args = ( apts[qtd/3:qtd*2/3], ) )
        th2.start()
        th3=threading.Thread( target=__captura_thread, args = ( apts[qtd*2/3:qtd], ) )
        th3.start()
        log.debug('Fim da captura de detalhes dos apartamentos por TRHREAD.')
        return th1,th2,th3

    def captura_detalhes():
        """
        Verifica os imoveis que ainda nao tenham sido visitados para recuperar suas
        informacoes especificas, nao disponiveis na lista inicial de anuncios.
        """

        apts = d.imoveis_nao_visitados()
        while len(apts)>0:
            log.debug('Capturando detalhes de {} apartamentos.'.format(len(apts)) )
            for ap in apts:
                response = requests.get(ap['url'])
                text = response.text
                response.close()

                if text != None:
                    try:
                        doc = __processa_html(text)
                        det = extrai_detalhes(doc)
                        det['id'] = ap['id']
                        det['dt_visita'] = d.dia_hoje()
                        d.atualiza_imoveis([det])
                    except Exception as e:
                        log.error('Erro apt ID = {}. Msg: {}'.format(ap['id'],e))
                        d.insere_erro(ap['id'],e)

```

```

apts = d.imoveis_nao_visitados()
log.debug('Fim da captura de detalhes dos apartamentos.')

def extrai_detalhes(doc):
    """
    Retorna os detalhes de um imovel.

    @param doc - extrato de texto HTML de um imovel.
    """

    def __is_number(s):
        try:
            float(s)
            return True
        except Exception:
            return False

    description, lat, lng, condo = None, None, None, None

    img = doc.find('img', class_='fichaMapa')
    if img != None:
        url_geo = img['src']
        start = url_geo.rindex('/') + 1
        end = url_geo.rindex('.png')
        coord = url_geo[start:end].split('_')
        if (len(coord) == 2 and
            __is_number(coord[0]) and __is_number(coord[1])):
            lat, lng = coord
        else:
            onclick = img['onclick']
            start = onclick.index(u'abreMapa')+len(u'abreMapa')
            end = start + 29
            coord = onclick[start:end].split(',')
            if (len(coord) == 2 and
                __is_number(coord[0]) and __is_number(coord[1])):
                lat, lng = coord

    h3_desc = doc.find('h3', attrs={'itemprop':'description'})
    if h3_desc != None:
        description = unicode(h3_desc.text)

    li = doc.find(id='ctl00_ContentPlaceHolder1_resumo_liCondominio')
    if li != None:
        condo = unicode(li.find('span', class_='featureValue').string).replace('R$', '').strip()

    det_imovel = ''
    for uls in doc.findAll('ul', class_='fc-detalhes'):
        det_imovel = det_imovel + uls.text.strip()+'\n'
    det_imovel = None if det_imovel.strip() == '' else det_imovel

    carac_imovel = None
    carac_condo = None
    for divs in doc.findAll('div', class_='fc-maisdetalhes'):
        if divs.p.string.find(u'Outras caracteristicas do imovel'):
            carac_imovel = divs.ul.text.strip()

        if divs.p.string.find(u'Caracteristicas das areas comuns'):
            carac_condo = divs.ul.text.strip()

    # Anuncio desativado.
    if len(doc.findAll('div', class_='erro_tit')) > 0:
        #carac_imovel = doc.find('div', class_='colunainfos').h3.text.strip()
        carac_imovel = doc.find('div', class_='colunainfos').text.strip()

    return {'descricao':description, 'lat':lat, 'lng':lng, 'condominio':condo, \
            'det_imovel':det_imovel, 'carac_imovel':carac_imovel, \
            'carac_condo': carac_condo}

```

Apêndice C

Listagem do módulo zap_util.py

Código fonte do módulo de utilidades que contém diversas funções utilizadas ao longo do estudo.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fgv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

```

# -*- coding: iso-8859-1 -*-

import sys,os;
import pandas as pd
import psycopg2
import unicodedata
from scipy.stats import norm as gauss, probplot, cumfreq
from matplotlib import rcParams
from matplotlib.pyplot import figure, xlim, ylim, pcolor, colorbar, xticks, \
    yticks, subplots, subplot, style, xlabel, ylabel, setp
from numpy import linspace, arange, std, polyfit, polyval, sqrt, mean, NaN
from matplotlib.colors import LogNorm
from matplotlib.gridspec import GridSpec
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Nomes das tabelas no banco de dados.
TBL_VAR_SOCIOAMB = '_var_socioambiental'
TBL_VAR_ESTRUTURAL = '_var_estrutural'
TBL_VWIMOVEL = 'vw_imovel'
VW_DISTANCIA = 'vw_distancia'

# Adiciona o diretório "capturar_zap" ao "path" do Python para poder acessar
# o módulo "dataset" nesse diretório.
x = os.getcwd()
l = x.rfind('/')
path = x[:l+1] + 'capturar_zap'
sys.path.append(path)

import dataset as d

IMOVEIS_CSV = './imoveis_dataset.csv'
def get_imoveis_from_csv():
    return pd.read_csv(IMOVEIS_CSV, index_col='id')

def set_style(sty='ggplot'):
    """
    Define estilo visual dos gráficos da biblioteca matplotlib.

    @param sty - nome do estilo dos gráficos a ser aplicado.
    """
    style.use(sty)

def exec_sql(sql, con=None):
    """
    Executa um SQL diretamente ao banco de dados e retorna seu resultado.

    @param sql - Código SQL a ser executado no banco de dados.
    @param con - Objeto de conexão ao banco de dados.
    """
    if con == None: con = d.conecta_db()
    return d._executar(sql, con=con)

def get(sql, id_='id', con=None):
    """
    Executa um SQL do tipo SELECT no banco de dados e retorna seu resultado como
    um pandas.DataFrame.

    @param sql - Código SQL do tipo SELECT a ser executado.
    @param id_ - Nome do campo a ser tratado como "index" do objeto
        pandas.DataFrame.
    @param con - Objeto de conexão ao banco de dados.
    """
    if con == None: con = d.conecta_db()
    return pd.io.sql.read_sql(sql, con, index_col=id_)

# Variável global para armazenar tamanho original das figuras da biblioteca
# matplotlib.
tam_fig_original = None
def tam_figura(largura=None, altura=None):
    """
    Define o tamanho das figuras dos gráficos da biblioteca matplotlib.
    Caso nenhum valor seja passado, a configuração do tamanho da figura é
    atribuído ao valor original.
    """

```

```

@param largura - Tamanho da largura da figura.
@param altura - Tamanho da altura da figura.
'''
if tam_fig_original == None:
    tam_fig_original = rcParams['figure.figsize']

if largura == None and altura == None:
    rcParams['figure.figsize'] = tam_fig_original
if largura != None and altura == None:
    rcParams['figure.figsize'] = [largura*i for i in tam_fig_original]
if largura != None and altura == None:
    rcParams = [largura, altura]
if largura == None and altura != None:
    raise Exception('Defina apenas "lagura" ou nenhuma.')

def remove_acento(str_or_list, codec='utf-8'):
    '''
    Retorna uma string ou uma lista de texto com todos os caracteres acentuados
    substituídos pelos seus respectivos caracteres sem acento.

    @param str_or_list - String ou lista de strings.
    @param codec - CÓDIGO da string ou lista de strings.
    '''
    if type(str_or_list) == str:
        s = str_or_list.decode(codec)
        return unicodedata.normalize('NFKD', s).encode('ascii', 'ignore')

    if type(str_or_list) == list:
        return [remove_acento(item) for item in str_or_list]

    if type(str_or_list) == unicode:
        return unicodedata.normalize('NFKD', str_or_list).encode('ascii',
            'ignore')

    msg = 'ERRO: ''unicode'', ''str'' ou ''list'' esperado. ' + \
          str(type(str_or_list)) + ' encontrado.'
    raise Exception(msg)

def prep_formula(dataframe, var_dep='preco', func=None, cat=[], ignore=[]):
    '''
    Retorna uma tupla contendo a fórmula para uso de regressão linear da
    biblioteca "statsmodels", uma lista de variáveis utilizadas na construção
    da fórmula e uma lista das variáveis rejeitadas para a utilizar na fórmula.

    As variáveis rejeitadas são aquelas que possuem valores nulos, junto com a
    lista de variáveis definidas no parâmetro "ignore".

    @param dataframe - Objeto pandas.DataFrame com o modelo para a regressão
        linear.
    @param var_dep - Nome da variável dependente da regressão linear.
    @param func - Função a ser aplicada na variável dependente.
    @param cat - Lista de variáveis independentes a serem tratadas como
        categóricas.
    @param ignore - Lista de variáveis independentes a serem ignoradas na
        construção da fórmula.
    ...
    cat=cat[:]
    ignore=ignore[:]

    # Determinar colunas que são identificadores para serem removidos.
    cols_ids = set([c for c in dataframe.columns if c.find('id_') > -1])

    # Identificar colunas que contém valores ausentes.
    s = dataframe.isnull().sum()
    cols_nulos = set(s[s > 0].index.tolist())

    # Colunas que são variáveis dependentes.
    cols_dep = set(['m2', 'preco'])

    # Juntar todas as colunas a serem removidas do modelo.
    cols_excluded = cols_ids.union(cols_nulos).\
        union(cols_dep).union(set(ignore))

    # Definir as variáveis para o modelo.
    cols = list(set(dataframe.columns) - cols_excluded)

```

```

# Construir a fórmula.
if func != None:
    yname = '{}({})'.format(func, var_dep)
else:
    yname = '{}'.format(var_dep)
formula = yname + ' ~ ' + ' + '.join(cols)

for c in cat:
    formula = formula.replace('{}'.format(c),
        'C({})'.format(c))

return formula, cols, cols_excluded

def plot_corrmatrix(matrix, figsize=(8*1.05, 6*1.05), **args):
    """
    Plota uma matriz de correlação em conjunto com uma legenda de cores.

    @param matrix - Matriz com os valores a terem a correlação verificada.
    @param figsize - Tamanho da figura a ser gerada.
    @param **args - Demais argumentos a serem passados para o método "pcolor".
    """
    figure(figsize=figsize)
    pcolor(matrix, **args);
    colorbar();

    max_ = len(matrix);
    yticks(arange(0.5,max_+0.5), range(0,max_));
    xticks(arange(0.5,max_+0.5), range(0,max_));

def vars_corr(matrix, threshold = 0.69999):
    """
    Retorna uma lista de variáveis correlacionadas entre si, onde o grau de
    correlação é dado pelo parâmetro "threshold".

    @param matrix - Matriz de correlação obtida pelo método
        pandas.DataFrame.corr().
    @param threshold - Limite inferior de determinação de correlação entre duas
        variáveis.

    ...
    from numpy import tril
    matrix.loc[:,:] = tril(matrix, k=-1) # borrowed from Karl D's answer

    already_in = set()
    result = {}
    for col in matrix:
        perfect_corr = matrix[col][abs(matrix[col]) > threshold].index.tolist()
        #if perfect_corr and col not in already_in:
        #    already_in.update(set(perfect_corr))
        #    perfect_corr.append(col)
        #    cols_autocorr.append(perfect_corr)
        if len(perfect_corr) > 0:
            result[col] = [i+'{:2f}'.format(matrix[col][i]) for i in perfect_corr]

    return result

def print_autocorr(dataframe, cols_excluded=[], threshold = 0.79999):
    """
    Imprime na saída padrão um relatório exibindo as variáveis correlacionadas
    entre si.

    @param dataframe - Objeto pandas.DataFrame que contém as variáveis a serem
        verificadas a correlação.
    @param cols_excluded - Lista de variáveis a serem excluídas da verificação
        de correlação.
    ...
    if type(cols_excluded) != set:
        cols_excluded = set(cols_excluded)
    # Selecionar as colunas do modelo.
    valid_cols = set(dataframe.columns.tolist()) - cols_excluded
    valid_cols = list(valid_cols)

    matrix_corr = dataframe[valid_cols].corr()
    cols_autoc = vars_corr(matrix_corr, threshold)

```

```

if len(cols_autoc) == 0:
    print(u'Nao ha colunas correlacionadas entre si.')
else:
    print 'Coluna'.ljust(20), '|', 'Autocorrelacionada com '.ljust(50)
    for k,c in cols_autoc.items():
        print str(k).ljust(20), ':', str(c).ljust(50)

def scatter_distancia(dfx,suptitle=None):
    '''
    Plota uma visualização customizada que exibe scatter plot para cada variável
    que representa uma distância no eixo X e o preço correspondente no eixo Y,
    ajustando uma reta para verificação da tendência de Y ao longo de X.

    @param dfx - Objeto pandas.DataFrame com as variáveis de interesse.
    @param suptitle - Título superior do gráfico.
    '''

    # Colunas que representam distâncias.
    dist_columns = sorted([c for c in dfx.columns if c.find('dist_') > -1])

    # Definir tamanho do gráfico.
    w,h = rcParams['figure.figsize']
    f,a = subplots(len(dist_columns), 2)
    f.set_size_inches(w*2, h*len(dist_columns))

    # Título central da figura.
    if suptitle != None:
        f.suptitle(suptitle)

    # Plotar gráficos.
    for i in range(len(dist_columns)):
        col_name = dist_columns[i]
        x = dfx[col_name]
        fx = linspace(x.min(),x.max(),50)

        a[i, 0].scatter(x, dfx.preco)
        p = polyfit(x,dfx.preco,1)
        a[i, 0].plot(fx, polyval(p,fx), linewidth=2)
        a[i, 0].set_title(col_name + u' x preço')

        a[i, 1].scatter(x,dfx.m2)
        p = polyfit(x,dfx.m2,1)
        a[i, 1].plot(fx, polyval(p,fx), linewidth=2)
        a[i, 1].set_title(col_name + u' x $R\$/m^2$')

def plot_boxhist(x,titulo=None,xlabel=None):
    '''
    Plota uma visualização customizada que consiste em um boxplot acima de um
    histograma, para visualizar a distribuição de uma variável "x" entre esses
    dois tipos de gráficos.

    @param x - Variável de interesse de visualização.
    @param titulo - Título do gráfico.
    @param xlabel - Rótulo a ser exibido para o eixo X.
    '''

    #f,a = subplots(2,1)
    #a0,a1 = a.ravel()

    fig = figure()
    gs = GridSpec(3,1)
    a0 = subplot(gs[0,:])
    a1 = subplot(gs[1:,:], sharex=a0)

    bp = a0.boxplot(x,vert=False);
    a0.text(max(x)*0.8,1.2,
            s='3$\sigma$={:.2f}'.format(3*std(x)),
            bbox={'facecolor':'w', 'pad':10, 'alpha':0.5},
            style='italic', fontsize=13);
    setp(a0.get_xticklabels(), visible=False);

    a1.hist(x,bins=30);

    if titulo != None:
        a0.set_title(titulo);
    if xlabel != None:

```

```

        a1.set_xlabel(xlabel);
gs.tight_layout(fig)

def rmse(resid):
    """
    Retornar o erro médio quadrático de uma lista de erros residuais de uma
    regressão.

    @param resid - Lista de erros residuais de uma regressão linear.

    return sqrt((resid**2/len(resid)).sum())

def get_imoveis_dataframe(with_missing_data=True, bairro_g=None):
    """
    Retorna um objeto pandas.DataFrame contendo todas as observações de imóveis
    com as variáveis de interesse do estudo.

    @param with_missing_data - Determina se o dataframe retornado trata as
    variáveis com valores ausentes ou não.

    @param bairro_g - Filtra pelo nome do bairro.

    ...

    # Variáveis intrínsecas básicas.
    sql = 'select id,area,p.preco_ajustado as preco,condominio,garagem,' + \
        ' quartos,suites,lat,lng,bairro_g,p.m2_ajustado as m2, '+'\
        'b.gid as id_bairro_g ' + \
        ' from {} i inner join preco_ajustado p '.format(TBL_VWIMOVEL) + \
        ' using(id) +' \
        ' inner join bairro b on b.nome = i.bairro_g'
    if bairro_g != None:
        sql += ' where bairro_g = \'{}\''.format(bairro_g)
    df = get(sql, None)
    df['bairro_g'] = df.bairro_g.apply(lambda x: remove_acento(x))

    # Variáveis sócio econômicas, por bairro.
    df_soc = get('select * from {}'.format(TBL_VAR_SOCIOAMB), None)
    del df_soc['nome']

    # Unir imóveis com variáveis socioeconômicas por bairro.
    df = pd.merge(df,df_soc, left_on='id_bairro_g', right_on='gid', how='inner')
    del df_soc

    # Variáveis espaciais.
    df_dist = get('select id, dist_favela,dist_bombeiro,dist_centro,' + \
        'dist_centro_lng,dist_centro_lat,dist_delegacia,dist_lagoa,' + \
        'dist_logradouro,dist_metro,dist_praia,dist_saude_publica,' + \
        'dist_saude_privada,dist_trem from {}'.format(VW_DISTANCIA), None)

    # Demais variáveis intrínsecas dummies.
    #var_not_dummies = ['andar','elevadores','ano','unidades']
    df_int = get('select * from {}'.format(TBL_VAR_ESTRUTURAL), None)

    # Unir com variáveis espaciais e intrínsecas.
    df_int.index = df_int.id
    df_dist.index = df_dist.id
    df.index = df.id
    del df['id'], df_dist['id'], df_int['id']
    df = pd.concat([df,df_dist,df_int], axis=1, join='inner')
    del df['gid'], df['bairro_g'],df_dist, df_int

    if not with_missing_data:
        df.condominio.fillna(0, inplace=True)
        df.garagem.fillna(0, inplace=True)
        df.suites.fillna(0, inplace=True)
        del df['andares'],df['ano'],df['unidades_andar']

    return df

def plot_residual(predicted,actual):
    """
    Plota uma visualização customizada com gráficos exibindo informações dos
    valores residuais de uma regressão linear.

```

```

@param smres - Lista de valores residuais de uma regressão linear.

delta = 100
# Gráfico superior esquerdo.
resid = predicted - actual
resid_std = (resid-mean(resid))/std(resid)
fig = figure()
w,h = rcParams['figure.figsize']
fig.set_size_inches(w*2,h*2)
ax = fig.add_subplot(221)
probplot(resid_std, plot=ax);
xlim(-5,5)

# Gráfico superior direito.
ax = fig.add_subplot(222)
ax.scatter(predicted,resid);
ax.axhline(y=mean(resid),xmin=min(predicted),xmax=max(predicted),color='r');
xlim(min(predicted)-delta,max(predicted)+delta)
ylim(min(resid)-delta,max(resid)+delta)
xlabel(u'Inferência')
ylabel('Residual')

# Gráfico inferior esquerdo.
ax = fig.add_subplot(223)
ax.hist(resid_std,100);
xlim(-5,5)

# Gráfico inferior direito.
ax = fig.add_subplot(224)
ax.hexbin(predicted,resid, norm=LogNorm());
ax.axhline(y=mean(resid),xmin=min(predicted),xmax=max(predicted),color='r');
xlim(min(predicted)-delta,max(predicted)+delta)
ylim(min(resid)-delta,max(resid)+delta)

def ols(dataframe,cat=[], ignore=[], avoid_corr=False, avoid_plow=True,
       alpha=0.05,corr_limit=0.8, remove_plow_by_step=False):
    ignore = ignore[:]
    cat = cat[:]
    dataframe = dataframe.copy()
    lm = None
    c,e = None, None

    # Proteção contra loop infinito.
    MAX_LOOP = dataframe.shape[1] +1

    cols = list(set(dataframe.columns.tolist())-set(ignore))

    # Remover variáveis correlacionadas entre si.
    if avoid_corr:
        while True:
            cols_autocorr = vars_corr( dataframe[cols].corr(),corr_limit).keys()
            if len(cols_autocorr) > 0:
                ignore = list(set(ignore).union(set(cols_autocorr)))
                cols = list(set(dataframe.columns.tolist())-set(ignore))
            else:
                break

    # Remover colunas estatisticamente insignificante.
    ignore_anterior = -1
    if avoid_plow:
        loop = 0
        while True:
            loop += 1

            # Evitar mais loops que variáveis.
            if loop > MAX_LOOP:
                raise Exception('Loop maior que variaveis!')

            f,c,e = prep_formula(dataframe,cat=cat,ignore=ignore)
            lm = sm.formula.ols(f, dataframe).fit()
            plow = lm.pvalues[lm.pvalues > alpha]
            plow.sort(ascending=False, inplace=True)

            # Identifica variáveis de baixo pvalue
            cols_plow = [i for i in plow.index.tolist()
                         if i.find('Intercept')==-1]

```

```

#print 'cols_plow: {}'.format(cols_plow)
if len(cols_plow) > 0:
    if remove_plow_by_step:
        ignore = list(set(ignore).union(set([cols_plow[0]])))
    else:
        if len(ignore) == ignore_anterior:
            raise Exception(u'Quantidade de variaveis a ignorar plow repetiu:
{}'.\
                format(len(ignore)))
        ignore = list(set(ignore).union(set(cols_plow)))
#print 'len ignore: {}'.format(ignore)
else:
    e = ignore
    break
else:
    f,c,e = prep_formula(dataframe,cat=cat,ignore=ignore)
    lm = sm.formula.ols(f, dataframe).fit()

del cat,ignore,dataframe
vars_used = [i for i in lm.params.index.tolist() if i != 'Intercept']
return lm, vars_used, e

def prep_statsmodels(dataframe,bairro_g=True,other_vars=True, unused_vars=False):
    '''
    Retorna dataframe preparado para ser executado no statsmodels.
    @param bairro_g - Determina se transforma bairros em dummies.
    @param demais - determina se transforma as variáveis quartos,suites e
    garagens em dummies.
    @param unused_vars - Determina se retorna variáveis não usadas no modelo:
    lat,lng,condominio e m2.
    '''

    if other_vars:
        # Varáveis categóricas
        for var_ in ['suites','quartos','garagem']:
            temp = pd.get_dummies(dataframe[var_])
            del temp[temp.columns.tolist()[0]]
            temp.columns = [var_+'_'+str(int(i)) for i
                            in temp.columns.tolist()]

            dataframe = pd.concat([dataframe,temp], axis=1)
            # Eliminar dependência linear.
            del dataframe[var_]

    if bairro_g:
        # Variável bairro.
        # Alterar nome para compatibilidade com stasmodels.
        dataframe['bairro_g'] = dataframe.bairro_g.apply(
            lambda x : x.replace(' ', '_').replace('(',')').replace(')', ''))

        db = pd.get_dummies(dataframe.bairro_g)
        # Eliminar dependência linear.
        del db['Centro']
        dataframe = pd.concat([dataframe,db],axis=1)
        del dataframe['bairro_g']

    if not unused_vars:
        for i in ['lat','lng','condominio','m2','dist_centro_lat',\
                  'dist_centro_lng','se_anos_estudo']:
            del dataframe[i]

    return dataframe

def run_model(dataframe, n_folds=5):
    '''
    Avalia a performance de X e retorna a média do RMSE de treino, teste e R^2.
    @param dataframe - pandas.DataFrame com o conjunto de dados.
    @param n_folds - Número de folds para o K_Fold. Se n_folds = 1, executa uma
    única iteração com todo o conjunto de dados.
    '''

    from sklearn import cross_validation as cv

```

```

rmse_train = [];
rmse_test = [];
r2 = [];

if n_folds == 1:
    lm,_,_ = ols(dataframe)
    return rmse(lm.resid),NaN,lm.rsquared
else:
    kfold = cv.KFold(len(dataframe),n_folds,shuffle=False);

    for train,test in kfold:
        df_test = dataframe.iloc[test];
        df_train = dataframe.iloc[train];
        lm,_,_ = ols(df_train);
        predict = lm.predict(df_test);
        resid = predict - df_test.precio;
        rmse_test.append(rmse(resid));
        r2.append(lm.rsquared);
        rmse_train.append(rmse(lm.resid));

    return mean(rmse_train),mean(rmse_test),mean(r2);

def res_coef(lm_result):
    """
    Converte um statsmodels.ResultWaper em um pandas.DataFrame ordenado
    pelos coeficientes.

    @param lm_result: statsmodels.ResultWaper
    """

    tbl = lm_result.params.copy()
    tbl.sort(ascending=False)
    dfv = pd.DataFrame(tbl, columns=['Coef'])
    return dfv

def res_coef_bairros(lm_result):
    """
    Converte um statsmodels.ResultWaper em um pandas.DataFrame com somente
    os coeficientes de bairros.
    """

    import re
    coef = lm_result.params.copy()
    coef.sort(ascending=False)
    vars_ = coef.index.tolist()
    import re
    p = re.compile(r'[A-Z].*')
    idx = [vars_[i] for i in range(len(vars_)) if p.search(vars_[i])]
    bairros = pd.DataFrame(coef.loc[idx], columns=['Coef'])
    bairros.drop('Intercept',inplace=True)
    return bairros

def res_coef_intr(lm_result):
    """
    Converte um statsmodels.ResultWaper em um pandas.DataFrame com somente
    os coeficientes das variáveis intrínsecas.
    """

    import re
    coef = lm_result.params.copy()
    coef.sort(ascending=False)
    vars_ = coef.index.tolist()
    p = re.compile(r'[A-Z].*')
    idx_ = [vars_[i] for i in range(len(vars_)) if not p.search(vars_[i])]
    varint = pd.DataFrame(coef.loc[idx_], columns=['Coef'])
    return varint

def statsmodels_df(lm_):
    """
    Converte a tabela de parâmetros de um objeto
    statsmodels.ResultWaper em um pandas.DataFrame ordenado
    pelo valor do coeficiente.
    """

    from collections import OrderedDict as OD
    dic = OD()
    dic['coef']=lm_.params
    dic['std err']=lm_.bse
    dic['t']=lm_.tvalues
    dic['P>|t|']=lm_.pvalues
    dic[['95% Conf. Int.']] = lm_.conf_int().apply(

```

```
        lambda x: '%1.3f'%x[0]+ ' +' '%1.3f'%x[1], axis=1)
df = pd.DataFrame.from_dict(dic, )
# Alteração do nome para manter o 'Intercept' no topo.
df.index = [i.replace('Intercept','!Intercept')
            for i in df.index.tolist()]
df.sort(inplace=True)
# Alteração do nome para manter o 'Intercept' no topo.
df.index = [i.replace('!Intercept','Intercept')
            for i in df.index.tolist()]
#df.index.name = 'Variavel'
return df

def getW():
    import pysal
    BASE_DIR = './spreg'
    K_NN = 100
    W_FILE = '{}/W_cidade_knn_{}.gal'.format(BASE_DIR,K_NN)
    f = pysal.open(W_FILE, 'r')
    w = f.read()
    f.close()
    w.transform = 'r'
    return w
```

Apêndice D

Listagem do IPython Notebook Variaveis_estruturais.ipynb

Código de construção das variáveis estruturais do modelo hedônico.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fgv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

Variaveis_estruturais

May 13, 2015

1 Variáveis estruturais

Este Notebook tem o objetivo de construir as variáveis estruturais a partir da descrição dos anúncios de imóveis. As demais variáveis numéricas `preco`, `area`, `quartos`, `suites`, `garagem`, `condominio`, `latitude`, `longitude`, já foram recuperadas durante o *WebScraping* do ZAP Imóveis.

In [1]: # Seção de imports.
%pylab inline

```
import zap_util as z
import re
z.set_style()
```

Populating the interactive namespace from numpy and matplotlib

In [2]: # Cria pandas.DataFrame com as descrições dos anúncios.
df = z.get('select id, det_imovel,carac_imovel, carac_condo ' + \
'from imovel where carac_imovel is not null or ' + \
' carac_condo is not null or det_imovel is not null')
df.describe()

Out[2]:

	det_imovel	carac_imovel	\
count	64555	77414	
unique	63054	46571	
top	dormitórios:2 quartos/dts\nsuites:1 suíte\nvag...	Interfone	
freq	6	1700	

	carac_condo
count	41449
unique	6815
top	Varanda
freq	3652

In [3]: # Tratar NAs.
df.fillna(' ', inplace=True)

Converter todos os textos para minúsculo.
df['carac_condo'] = df.carac_condo.apply(str.lower)
df['carac_imovel'] = df.carac_imovel.apply(str.lower)
df['det_imovel'] = df.det_imovel.apply(str.lower)
df['carac'] = df.carac_imovel+\n+df.carac_condo+\n+df.det_imovel

Unir todas as descrições em um único campo.
del df['carac_condo'], df['carac_imovel'], df['det_imovel']

```
In [4]: # Remover acentuações.  
df.carac = df.carac.apply(z.remove_acento)
```

1.1 Variáveis dummies.

Esta seção irá construir variáveis estruturais *dummies*, que indicam a presença, valor 1, ou ausência, valor 0, de um característica do imóvel.

```
In [5]: import re  
import nltk  
  
# Expressão regular para localizar palavras somente com letras e  
# com mínimo de 4 letras.  
re_word = re.compile(r'[a-z]{3,}')  
  
# Stopwords do corpus NLTK.  
stopwords = z.remove_acento(nltk.corpus.stopwords.words('portuguese'))  
  
# Adiciona stopwords da análise das descrições.  
stopwords.extend(['dts', 'ser', 'alterado', 'sem', 'aviso', 'previo',  
'area', 'quarto', 'quartos', 'dormitorio', 'dormitorios', 'valor',  
'venda', 'vel', 'preco', 'vaga', 'vagas', 'suite', 'suites', 'total',  
'util', 'ligue', 'timo', 'dio', 'otima', 'bom', 'claro', 'arejado',  
'impecavel', 'lindo', 'pronto', 'morar', 'totalmente', 'shopping',  
'toda', 'junto', 'metro', 'estacao', 'ref.', 'vagasconstruido', 'iptu',  
'banheiro', 'unidades', 'construido', 'imovel', 'todo', 'predio', 'rua',  
'excelente', 'maravilhoso'])  
#', '@', '#', '£', '%', '„', '„', '*', '(', ')', '‘', '‘', '+', '=', '–', '.', ',', '‘', '‘', '„', '„', ';', '?', '!', '£'])
```

```
In [6]: # Transformar as descrições em bag of words.  
df['words'] = df.carac.apply(lambda x: [w for w in nltk.regexp_tokenize(x, re_word) if w not in stopwords])
```

```
In [7]: # Contagem de monogramas.  
fd_1gram = nltk.FreqDist()  
for words in df.words:  
    fd_1gram.update(words)
```

```
In [8]: # Exibe as "n" maiores ocorrências de monogramas.  
fd_1gram.most_common(n=20)
```

```
Out[8]: [('andares', 113767),  
         ('condominio', 96373),  
         ('sala', 61985),  
         ('cozinha', 50979),  
         ('salao', 40470),  
         ('empregada', 37358),  
         ('andar', 34064),  
         ('armario', 33886),  
         ('varanda', 32342),  
         ('apartamento', 29565),  
         ('festas', 24905),  
         ('piscina', 24263),  
         ('churrasqueira', 22354),  
         ('servico', 19018),  
         ('playground', 18836),
```

```
('piso', 18822),  
('social', 18063),  
('sauna', 17560),  
('elevadores', 16660),  
('armarios', 16064)]
```

```
In [9]: # Contagem de bigramas.  
fd_2gram = nltk.FreqDist()  
for words in df.words:  
    fd_2gram.update(nltk.bigrams(words))
```

```
In [10]: # Exibe as "n" maiores ocorrências de bigramas.  
fd_2gram.most_common(n=20)
```

```
Out[10]: [((‘andares’, ‘andares’), 43864),  
((‘andares’, ‘condominio’), 40147),  
((‘salao’, ‘festas’), 23513),  
((‘armario’, ‘cozinha’), 16660),  
((‘armario’, ‘embutido’), 15580),  
((‘andares’, ‘andar’), 14574),  
((‘empregada’, ‘andares’), 13949),  
((‘sala’, ‘jantar’), 13182),  
((‘cozinha’, ‘armario’), 13005),  
((‘condominio’, ‘andares’), 11574),  
((‘piscina’, ‘playground’), 11385),  
((‘sala’, ‘ginastica’), 10648),  
((‘salao’, ‘jogos’), 9709),  
((‘ginastica’, ‘salao’), 9604),  
((‘elevadores’, ‘andares’), 8791),  
((‘festas’, ‘salao’), 8190),  
((‘varanda’, ‘empregada’), 8044),  
((‘sala’, ‘almoco’), 8016),  
((‘varanda’, ‘andares’), 7700),  
((‘jantar’, ‘varanda’), 7438)]
```

```
In [11]: # Contagem de 3-gramas.  
fd_3gram = nltk.FreqDist()  
for words in df.words:  
    fd_3gram.update(nltk.ngrams(words,3))
```

```
In [12]: # Exibe as "n" maiores ocorrências de bigramas.  
fd_3gram.most_common(n=20)
```

```
Out[12]: [((‘andares’, ‘andares’, ‘condominio’), 40044),  
((‘empregada’, ‘andares’, ‘andares’), 13520),  
((‘armario’, ‘cozinha’, ‘armario’), 12818),  
((‘cozinha’, ‘armario’, ‘embutido’), 12812),  
((‘sala’, ‘ginastica’, ‘salao’), 9585),  
((‘condominio’, ‘andares’, ‘andar’), 9029),  
((‘ginastica’, ‘salao’, ‘festas’), 9012),  
((‘salao’, ‘festas’, ‘salao’), 8180),  
((‘festas’, ‘salao’, ‘jogos’), 8029),  
((‘varanda’, ‘andares’, ‘andares’), 7658),
```

```
(('sala', 'jantar', 'varanda'), 7381),
 (('salao', 'jogos', 'sauna'), 6913),
 (('sala', 'almoco', 'sala'), 6701),
 (('almoco', 'sala', 'jantar'), 6698),
 (('varanda', 'empregada', 'andares'), 6675),
 (('playground', 'salao', 'festas'), 6306),
 (('piscina', 'playground', 'quadra'), 5725),
 (('condicionado', 'armario', 'cozinha'), 5494),
 (('quadra', 'poliesportiva', 'sala'), 5169),
 (('poliesportiva', 'sala', 'ginastica'), 5157)]
```

```
In [13]: # Copiar lista dos n-gramas em memória para gerar tabelas Latex em
# http://www.tablesgenerator.com/latex_tables.
a = [i[0] for i in fd_1gram.most_common(n=20)]
b = [i[0][0]+ '+'+i[0][1] for i in fd_2gram.most_common(n=20)]
c = [i[0][0]+ '+'+i[0][1] + ' ' + i[0][2] for i in fd_3gram.most_common(n=20)]
x = z.pd.DataFrame(c)
print x.to_clipboard(index=0, )
```

None

```
In [14]: # DataFrame para as variáveis.
dfc = z.pd.DataFrame(index=df.index)

# Helper para processar a característica "nome" e
# armazenar na variável "dfc[nome]".
def caract(nome):
    dfc[nome] = df.carac.str.contains(nome)

features = [
    'armario',
    'banheira',
    'blindex',
    'churrasqueira',
    'closet',
    'cobertura',
    'copa',
    'duplex',
    'elevador',
    'escritura',
    'esquina',
    'frente',
    'fundos',
    'granito',
    'hidrometro',
    'linear',
    'mezanino',
    'original',
    'planejad',
    'piscina',
    'play',
    'porcelanato',
    'portaria',
    'recuado',
```

```

'sauna',
'servico',
'terraco',
'triplex',
'varanda'
]

for f in features:
    caract(f)

# Demais características que envolvam expressão regular.
dfc['dependencia_empregada'] = df.carac.str.\
    contains(r'[dependencia|quarto]\s*[da|de]?\s*empregada') | \
    df.carac.str.contains(r'dependencia')
dfc['estacionamento_visitantes'] = \
    df.carac.str.contains(r'(estacionamento|vaga(s?))\s*(d[aeiou])?\s*visitante')
dfc['seguranca'] = df.carac.str.contains(r'guarita|seguranca')
dfc['indevassavel'] = df.carac.str.contains(r'indevassavel|vista\s*livre')
dfc['vista_mar'] = df.carac.str.contains(r'vista\s*(livre|para)?\s*[mar|praia]')
dfc['sol_manha'] = df.carac.str.contains(r'sol\s*[da|de|di|do|du]?\s*manh')
dfc['sol_tarde'] = df.carac.str.contains(r'sol\s*[da|de|di|do|du]?\s*tarde')
dfc['banheira'] = df.carac.str.contains(r'hidromassagem|banheira')
dfc['elevador_privado'] = df.carac.str.contains(r'elevador\s*[privado|particular]')
dfc['andar_alto'] = df.carac.str.contains(r'andar\s*alto')
dfc['andar_baixo'] = df.carac.str.contains(r'andar\s*baixo')
dfc['andar_inteiro'] = df.carac.str.contains(r'andar\s*inteiro') | \
    df.carac.str.contains(r'1 unidade por andar') | \
    df.carac.str.contains(r'1 por andar')
dfc['creche'] = df.carac.str.contains(r'creche|children\s*care')
dfc['sala_jantar'] = df.carac.str.contains(r'sala\s*(de)?\s*jantar')
dfc['salao_festas'] = df.carac.str.contains(r'salao\s*(de)?\s*festa')
dfc['salao_jogos'] = df.carac.str.contains(r'salao\s*(de)?\s*jogos')
dfc['lateral'] = df.carac.str.contains('lateral|de\s*lado')

# Adicionar prefixo "dm_" para identifica variáveis como dummies.
dfc.columns = ['dm_' + i for i in dfc.columns.tolist()]

```

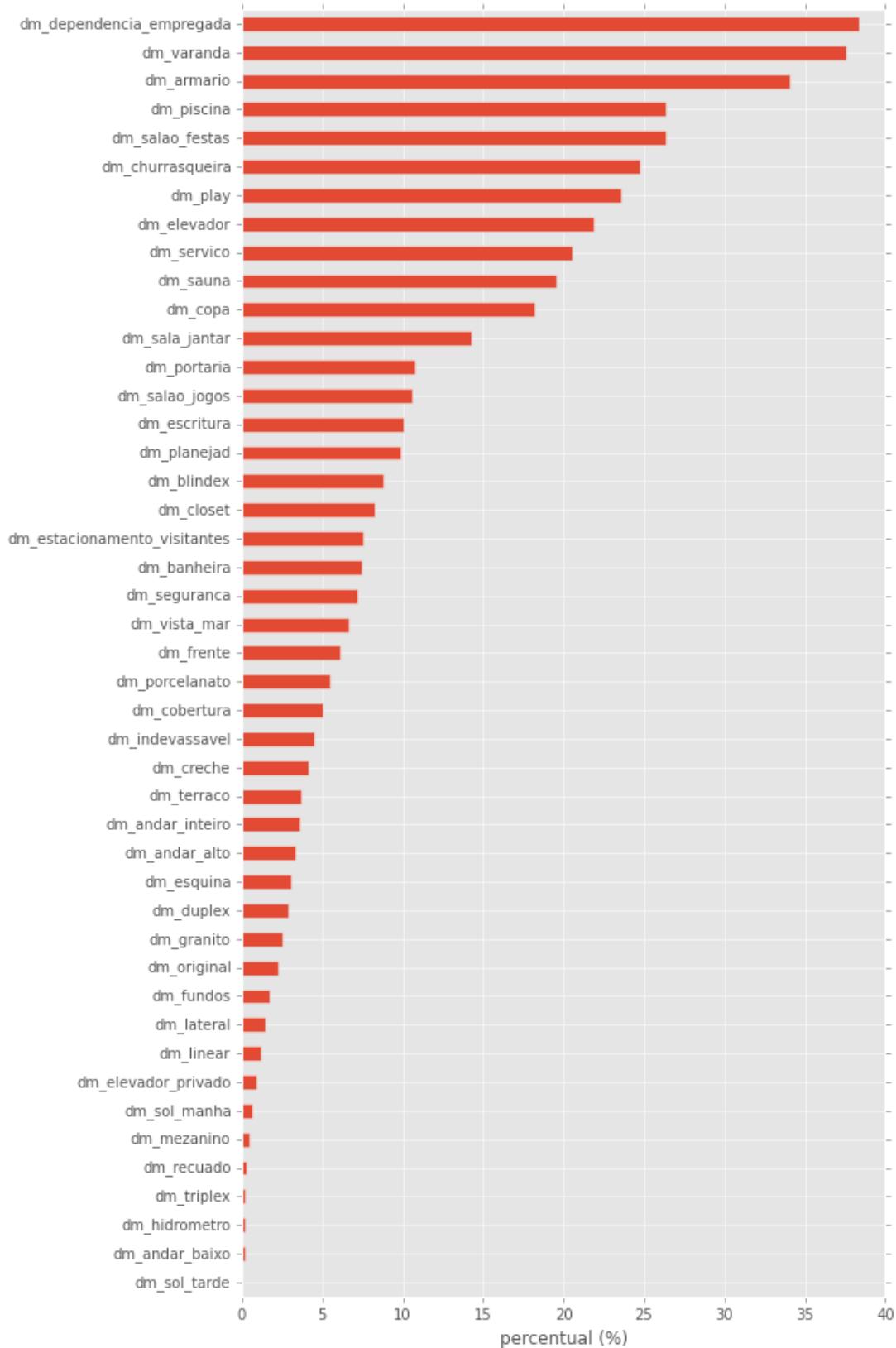
/Users/sergio/anaconda3/envs/py27/lib/python2.7/site-packages/pandas/core/strings.py:184: UserWarning: ' groups, use str.extract.", UserWarning)

In [15]: # Visualizar distribuição das variáveis.

```

s = dfc.sum()/len(dfc)*100
s.sort(ascending=True)
s.plot(kind='barh', figsize=(8,16));
xlabel('percentual (%)');

```



```
In [16]: savefig('../')
<matplotlib.figure.Figure at 0x10c2e8a50>
```

1.2 Variáveis numéricas.

Nesta seção, construiremos o restante das variáveis *dummies* numéricas.

Primerio, iremos tokenizar novamente, desta vez usando somente as pontuações junto com limites de palavras para delimitar os tokens. Com isso, os números voltam a fazer parte do conjunto de tokens.

```
In [17]: # Coleção de textos para verificar ocorrência vizinha de palavras.
coll = nltk.TextCollection(df.carac.apply(lambda x: [w for w in nltk.wordpunct_tokenize(x)]))
```

1.2.1 Variável unidades_andar

Variável que indica o número de unidades por andar.

```
In [18]: # Ver palavras vizinhas à "unidade".
coll.concordance('unidade', lines=10)
```

Displaying 10 of 2325 matches:

```
s 250 m2 area util4 andares 1 vagas1 unidade por andar excelente apartamento na
/ dts . 8 andares 152 m2 area util1 unidade por andar 152 m2 area total2 elevad
/ dts . 5 andares 159 m2 area util1 unidade por andar 1 vaga excelente localiza
23 . 377 3 quartos / dts . 1 suite1 unidade por andar 154 m2 area util154 m2 ar
2 quartos / dts . 82 m2 area total1 unidade por andar 82 m2 area utilconstruido
4 . 949 3 quartos / dts . 1 suite1 unidade por andar 172 m2 area util172 m2 ar
do sol : tarde - posicionamento da unidade : lateral - ref .: bt10113 venda [?
$ 6 . 739 1 quarto / dt . 1 suite1 unidade por andar 46 m2 area util146 m2 area
10 . 250 4 quartos / dts . 1 suite1 unidade por andar 200 m2 area util213 m2 ar
7 . 915 3 quartos / dts . 1 suite1 unidade por andar 82 m2 area util107 m2 are
```

```
In [19]: coll.concordance('unidades', lines=10)
```

Displaying 10 of 22427 matches:

```
139 m2 area util139 m2 area total2 unidades por andar 2 vagasconstruido em 198
5 . 098 3 quartos / dts . 1 suite4 unidades por andar 102 m2 area util102 m2 a
/ dt . 5 andares 26 m2 area util7 unidades por andar 26 m2 area total excelen
s 97 m2 area util9 andares 1 vagas8 unidades por andar Otima cobertura duplex ,
/ dts . 4 andares 65 m2 area util2 unidades por andar 70 m2 area total excelen
97 m2 area util12 andares 1 vagas4 unidades por andar apartamento proximo a pr
315 m2 area util3 andares 3 vagas2 unidades por andar jardim oceanico ! beliss
6 . 357 3 quartos / dts . 1 suite4 unidades por andar 140 m2 area util140 m2 a
m2 area util12 andares 1966 vagas6 unidades por andar ( flamengo ) proximo mus
. 692 4 quartos / dts . 2 suites8 unidades por andar 208 m2 area util208 m2 a
```

```
In [20]: # Capturar a quantidade de unidades por andar.
```

```
re_unid_andar = re.compile(r'(\d{1,3}) unidades por andar')
def obtem_unidades_andar(s):
    r = None
    m = re_unid_andar.search(s)
```

```

if m:
    r = int(m.groups(0)[0])
return r
unidades_andar = df.carac.apply(obtem_unidades_andar)

In [21]: # Verificar dados descritivos de "unidades por andar".
unidades_andar.describe()

Out[21]: count    21505.000000
          mean      5.729226
          std       9.194468
          min      2.000000
          25%     4.000000
          50%     4.000000
          75%     8.000000
          max     868.000000
          Name: carac, dtype: float64

In [22]: # Ver distribuição de valores.
unidades_andar.unique()

Out[22]: array([ nan,    2.,    4.,    7.,    8.,    6.,    5.,   14.,   10.,
       13.,    3.,   15.,   12.,   16.,   11.,    9.,   20.,   18.,
       25.,   40.,   30.,   23.,   32.,   17.,   22.,   24.,   21.,
       35.,   36.,   45.,  868.,   42.,   47.,   38.,   26.,   19.,
       28.,   27.,  213.,  532.,   29.,   401.,   34.,   31.,   50.,   564.])

In [23]: # Eliminar outliers.
unidades_andar = unidades_andar[unidades_andar < 100]

In [24]: # Verificar em detalhes uma ocorrência alta.
df.loc[unidades_andar[unidades_andar == 47].index].carac.values

Out[24]: array(['venda [?]\r\n'                                r$ 1.190.000 \n\n\n\n\n\n\ncondomi'])

In [25]: dfc['unidades_andar'] = unidades_andar

```

1.2.2 Variável andar

Representa em qual andar está o imóvel.

DESIMOS DESSA VARIÁVEL POR NÃO COSEGUIRMOS DETERMINAR UMA ESTRATÉGIA DE CAPTURA.

```
In [26]: coll.concordance('andar')
```

Displaying 25 of 34043 matches:

aramento na escritura . apartamento de andar alto , vista livre e indevassado de
util139 m2 area total2 unidades por andar 2 vagasconstruido em 19802 elevadore
uartos / dts . 1 suite4 unidades por andar 102 m2 area util102 m2 area total2 e
ndares 26 m2 area util7 unidades por andar 26 m2 area total excelente apartamen
a util9 andares 1 vaga8 unidades por andar Otima cobertura duplex , 1a locaCAo
banheiro social , copa / cozinha , 2 andar piscina , churrasqueira , banheiro s
ea util4 andares 1 vaga1 unidade por andar excelente apartamento na prudente de
ias completas . Otimo predio , 1 por andar com 4 andares , vaga na escritura ar
ndares 65 m2 area util2 unidades por andar 70 m2 area total excelente apartamen
ndares 152 m2 area util1 unidade por andar 152 m2 area total2 elevadores para c
r \$ 3 . 103 frente para o mar sauna andar inteiro armario de cozinha armario e

util12 andares 1 vaga4 unidades por andar apartamento proximo a praia , otimo
 nro proximo a praia , otimo predio , andar alto , reformado , cheio de armarios
 util3 andares 3 vagas2 unidades por andar jardim oceanico ! belissima cobertur
 uartos / dts . 1 suite4 unidades por andar 140 m2 area util140 m2 area total2 e
 112 andares 1966 vagas6 unidades por andar (flamengo) proximo museu republica
 ndares 159 m2 area util1 unidade por andar 1 vaga excelente localizacao , em ru
 de apenas cinco pavimentos , um por andar . apartamento claro e arejado , comp
 artos / dts . 2 suites8 unidades por andar 208 m2 area util208 m2 area total2 e
 a util7 andares 1 vaga6 unidades por andar 2 quartos , sala , cozinha e banheiro
 vico integrada a cozinha americana . andar alto . sol da manha . bela vista tot
 uartos / dts . 1 suite2 unidades por andar 94 m2 area util94 m2 area total2 ele
 util10 andares 1 vaga8 unidades por andar incrivel ! predio com portaria 24 hs
 4 hs . apartamento claro , arejado , andar alto , sol da manha , vista livre ,
 ts . 100 m2 area total2 unidades por andar 100 m2 area utilconstruido em 1970 1

1.2.3 Variável andares

Informa quantos andares possui o prédio do imóvel.

```
In [27]: coll.concordance('andares')
```

Displaying 25 of 113759 matches:

s vagas : 1 vaga area util : 150 m2 andares : 3 andares construido : 1970 preco
 vaga area util : 150 m2 andares : 3 andares construido : 1970 preco de venda :
 : 2 quartos / dts area util : 81 m2 andares : 12 andares construido : 1971 preco
 dts area util : 81 m2 andares : 12 andares construido : 1971 preco de venda :
 s vagas : 1 vaga area util : 280 m2 andares : 11 andares preco de venda : r \$ 2
 aga area util : 280 m2 andares : 11 andares preco de venda : r \$ 2 . 650 . 000
 25 . 899 3 quartos / dts . 1 suite5 andares 139 m2 area util139 m2 area total2
 ts vagas : 1 vaga area util : 48 m2 andares : 8 andares preco de venda : r \$ 23
 vaga area util : 48 m2 andares : 8 andares preco de venda : r \$ 230 . 000 cond
 2 m2 area total2 elevadores 1 vaga5 andares bom predio com play e salao de fest
 m2 : r \$ 8 . 846 1 quarto / dt . 5 andares 26 m2 area util7 unidades por andar
 s : 1 quarto / dt area util : 26 m2 andares : 5 andares preco de venda : r \$ 23
 / dt area util : 26 m2 andares : 5 andares preco de venda : r \$ 230 . 000 cond
 ts vagas : 1 vaga area util : 70 m2 andares : 4 andares preco de venda : r \$ 26
 vaga area util : 70 m2 andares : 4 andares preco de venda : r \$ 260 . 000 cond
 total2 elevadores 97 m2 area util9 andares 1 vaga8 unidades por andar Otima co
 ts vagas : 1 vaga area util : 97 m2 andares : 9 andares preco de venda : r \$ 38
 vaga area util : 97 m2 andares : 9 andares preco de venda : r \$ 380 . 000 cond
 ts vagas : 1 vaga area util : 62 m2 andares : 5 andares construido : 1999 preco
 vaga area util : 62 m2 andares : 5 andares construido : 1999 preco de venda :
 suite2 elevadores 250 m2 area util4 andares 1 vaga1 unidade por andar excelente
 . Otimo predio , 1 por andar com 4 andares , vaga na escritura ar condicionado
 te vagas : 1 vaga area util : 74 m2 andares : 5 andares preco de venda : r \$ 89
 vaga area util : 74 m2 andares : 5 andares preco de venda : r \$ 895 . 000 cond
 e vagas : 2 vagas area util : 77 m2 andares : 3 andares construido : 2013 preco

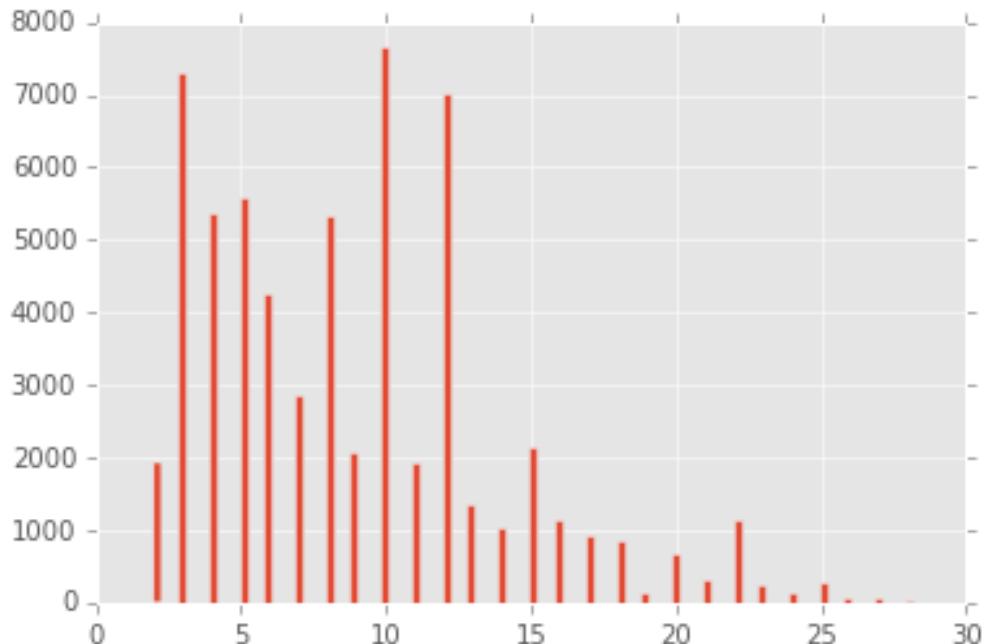
```
In [28]: re_andares = re.compile(r'(\d{1,3}) andares')
def obtem_andares(s):
    r = None
    m = re_andares.search(s)
    if m:
        r = int(m.groups(0)[0])
    return r
andares = df.carac.apply(obtem_andares)
```

```
In [29]: andares.unique()
```

```
Out[29]: array([ 3., nan, 12., 11., 5., 8., 4., 9., 6.,
 15., 10., 7., 2., 16., 22., 25., 17., 28.,
 20., 13., 441., 14., 21., 23., 18., 26., 24.,
 19., 107., 27., 35., 29., 30., 100., 121., 38.,
 42., 51., 32., 112., 140., 60., 34., 41., 70.,
 202., 53., 101., 50., 80., 36.])
```

```
In [30]: andares[andares < 30].hist(bins=100)
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x10c2e86d0>
```



```
In [31]: # Verificar em detalhes uma ocorrência alta.
```

```
df.loc[andares == 140].index[carac.values]
```

```
Out[31]: array(['campo de futebol\nchurrasqueira\nestacionamento visitantes\nnpiscina\nplayground\nquadra',
```

```
In [32]: dfc['andares'] = andares
```

1.2.4 Variável ano

Informa o ano em que o prédio do imóvel foi construído.

```
In [33]: coll.concordance('construcao')
```

Displaying 25 of 416 matches:

- familiares , e 8 projecoes para construcao de residencias multi - familiares
iancas e espaco p / academia (em construcao). imovel quitado , livre de onus
l 1 vaga excelente apartamento em construcao , com previsao de entrega para de
o , novo (com menos de 4 anos de construcao), todo montado com armarios plan

condominio barra sunday , ano de construcao 2011 , composto de varanda com vira e aprovada pela prefeitura para construcao de aproximadamente 70m2 . 2 vagas hoje linear , toda preparada para construcao na laje . sala em 02 ambientes coisa , novo (com menos de 2 anos de construcao), frontal , claro e arejado , proximo centro de terreno , excelente construcao , muito potencial . living com 10 quartos , novo (com menos de 3 anos de construcao), pronto para entrar e morar , completo , tipo 1alocacao com 2 anos de construcao . sala , 2 quartos , cozinha planejada da barra da tijuca , metro em construcao . 1o piso : varandao , area exterior , praia , mar e campo de golf em construcao para as olimpiadas . ar condicionado . temos outros no mesmo nivel em construcao . saimoveis salao de festas armario de luxo , lazer completo , em construcao com conclusao para 2015 . apartamento , novo (com menos de 2 anos de construcao), vazio , pronto para entrar e m 56m), novo (menos de 2 anos de construcao), composto de : varanda gourmet dores vintage way - sao conrado - construcao joao fortes . entrega em dez / 14 edificio com portaria 24 horas . construcao recente . melhor trecho do bairro em condominio . somente a vista ! construcao particular ! esquina salao de festao automatico , possibilidade de construcao de 2o andar . documentacao perfeita do mesmo proprietario desde sua construcao . se voce esta procurando um apartamento tijuca , com apenas tres anos de construcao . linda vista para lagoa e ampla rantes - excelente apartamento em construcao . composto de 3 qtos , 1 suite , apartamento (menos de 2 anos de construcao), em privilegiada localizacao ,

```
In [34]: coll.concordance('construido')
```

Displaying 25 of 19385 matches:

```

util : 150 m2 andares : 3 andares construido : 1970 preco de venda : r $ 1 . 6
util : 81 m2 andares : 12 andares construido : 1971 preco de venda : r $ 880 .
util : 62 m2 andares : 5 andares construido : 1999 preco de venda : r $ 280 .
util : 77 m2 andares : 3 andares construido : 2013 preco de venda : r $ 560 .
util : 140 m2 andares : 6 andares construido : 2013 preco de venda : r $ 699 .
util : 69 m2 andares : 10 andares construido : 2014 preco de venda : r $ 508 .
m2 : r $ 15 . 152 1 quarto / dt . construido em 19532 elevadores 33 m2 area ut
util : 70 m2 andares : 3 andares construido : 1980 preco de venda : r $ 410 .
util : 81 m2 andares : 6 andares construido : 1970 preco de venda : r $ 565 .
util : 129 m2 andares : 6 andares construido : 1980 preco de venda : r $ 660 .
vagas : 1 vaga area util : 114 m2 construido : 2000 preco de venda : r $ 1 . 2
util : 140 m2 andares : 9 andares construido : 1990 preco de venda : r $ 3 . 5
2 : r $ 3 . 836 3 quartos / dts . construido em 1975 73 m2 area util4 andares
til : 127 m2 andares : 22 andares construido : 2009 preco de venda : r $ 1 . 4
util : 165 m2 andares : 3 andares construido : 1995 preco de venda : r $ 2 . 0
til : 113 m2 andares : 17 andares construido : 2006 preco de venda : r $ 1 . 2
til : 217 m2 andares : 28 andares construido : 1980 preco de venda : r $ 1 . 6
util : 60 m2 andares : 6 andares construido : 1970 preco de venda : r $ 370 .
til : 183 m2 andares : 12 andares construido : 1952 preco de venda : r $ 1 . 6
util : 88 m2 andares : 6 andares construido : 2010 preco de venda : r $ 650 .
util : 82 m2 andares : 3 andares construido : 1960 preco de venda : r $ 265 .
util : 90 m2 andares : 8 andares construido : 1977 preco de venda : r $ 1 . 7
vagas : 1 vaga area util : 84 m2 construido : 2003 preco de venda : r $ 915 .
util : 86 m2 andares : 15 andares construido : 2014 preco de venda : r $ 660 .
util : 80 m2 andares : 22 andares construido : 1998 preco de venda : r $ 800 .

```

```
In [35]: re_ano = re.compile(r'construido\s*(?::|em)\s*(\d{2,4})')
```

```
m = re_ano.search('construido : 2938')
```

```

def obtem_ano_construcao(s):
    r = None
    m = re_ano.search(s)
    if m:
        r = int(m.groups(0)[0])
    return r

ano = df.carac.apply(obtem_ano_construcao)

In [36]: # Verificar em detalhes uma ocorrência alta.
df.loc[ano[ano == 343].index].carac.values

Out[36]: array(['venda [?]\r\n', r$ 1.360.000 \n\n\n\n\n\n\n\n\ncondomi'])

In [37]: # Remove inexistentes.
ano.dropna(inplace=True)

# Encontra quantidade de caracteres.
ano = ano.astype('int')
ano = ano.apply(str)
qtd = ano.str.len()
qtd.value_counts().index.values

Out[37]: array([4, 2, 3])

In [38]: # Separa em grupo de 3 e 4 caracteres.
ano2d = ano[ano.str.len() == 2].astype('int')
ano3d = ano[ano.str.len() == 3].astype('int')
ano4d = ano[ano.str.len() == 4].astype('int')

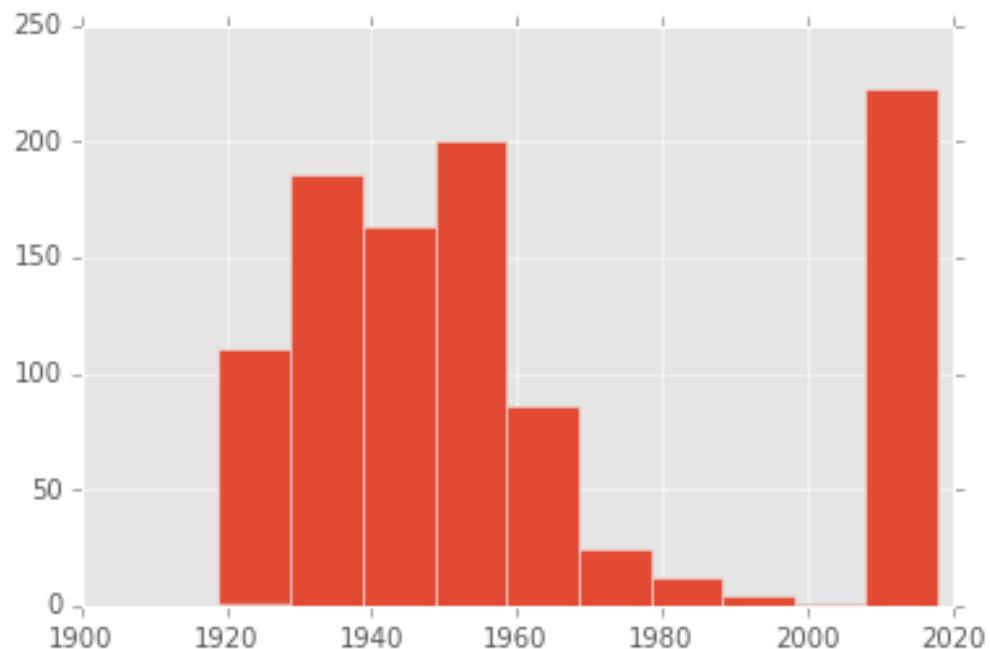
# Verificar como está o grupo de 3 caracteres.
#boxplot(ano3d)
hist(ano2d);


```

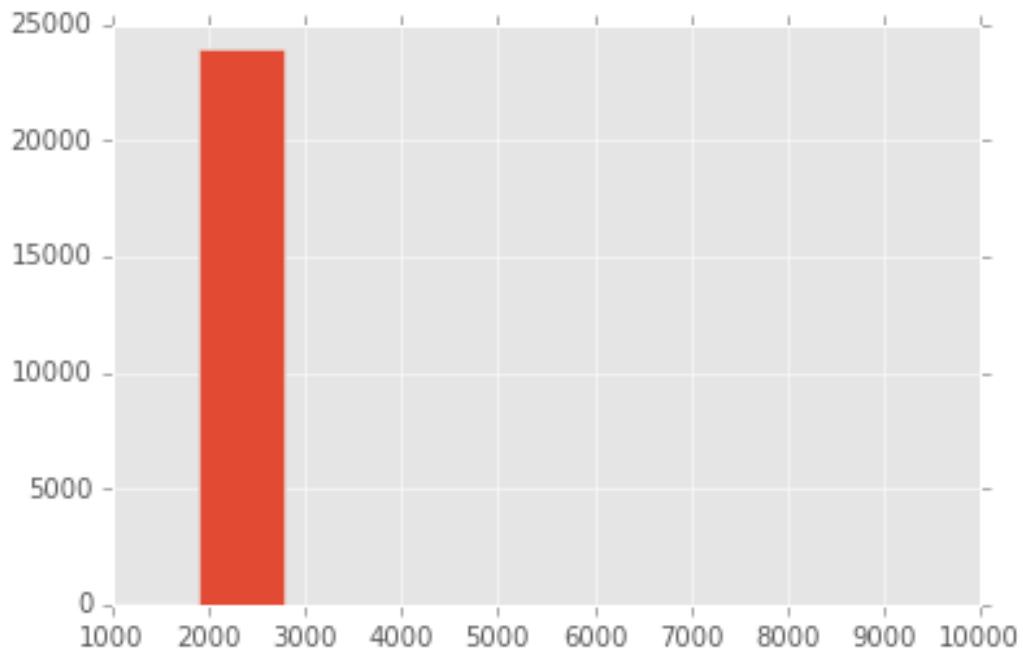
Number of Characters	Frequency
10	175
15	68
20	118
25	87
30	110
35	60
40	20
45	10
50	5
55	2
60	1

```
In [39]: # Determinamos que o grupo de 3 caracteres consiste na junção do ano em 2 caracteres com o número de milhares
# Faremos então o ano ser composto apenas dos dois primeiros caracteres, e juntá-los com os de milhares
x = ano3d.astype('str').str[:2].astype('int')
ano2d = z(pd.concat([x,ano2d], axis=0))
```

```
In [40]: # Definimos que os anos entre 19 e 99 percentem ao século 1900, e de 00 a 18, ao século 2000.
grupo1 = (ano2d >= 19) & (ano2d <=99)
grupo2 = ~grupo1
ano2d.loc[grupo1] = ano2d[grupo1] + 1900
ano2d.loc[grupo2] = ano2d[grupo2] + 2000
hist(ano2d);
```

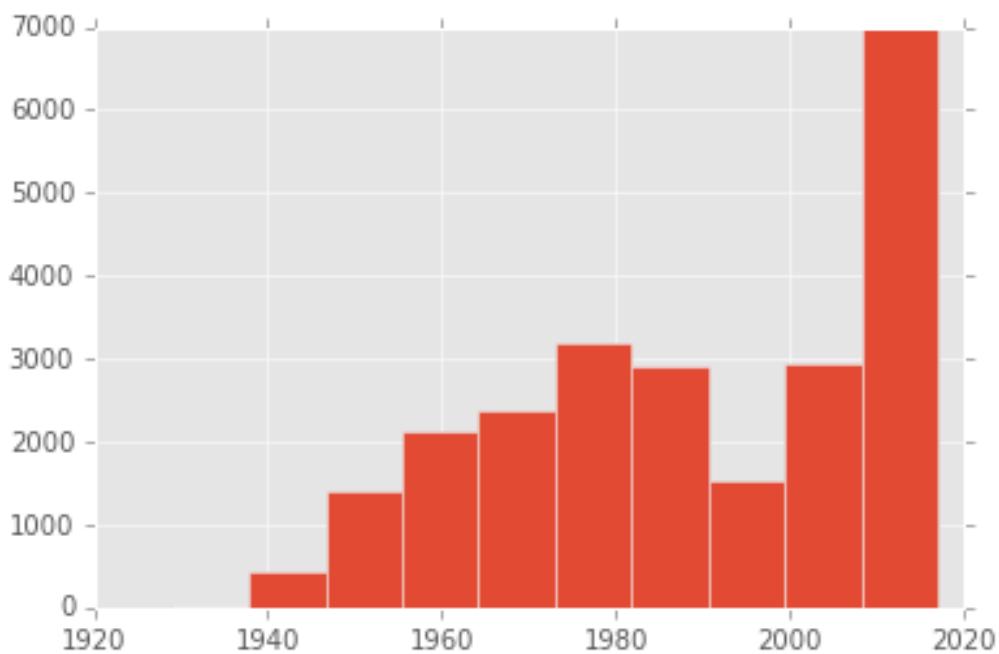


```
In [41]: # Verificar como está o grupo com 4 dígitos.
hist(ano4d[~ano4d.isnull()]);
```



```
In [42]: # Tratar anos inferiores a 1920 e superiores a 2017 como None.  
ano4d.loc[ano4d > 2017] = None  
ano4d.loc[ano4d < 1920] = None
```

```
In [43]: # Verificar como está o grupo com 4 dígitos.  
hist(ano4d[~ano4d.isnull()]);
```

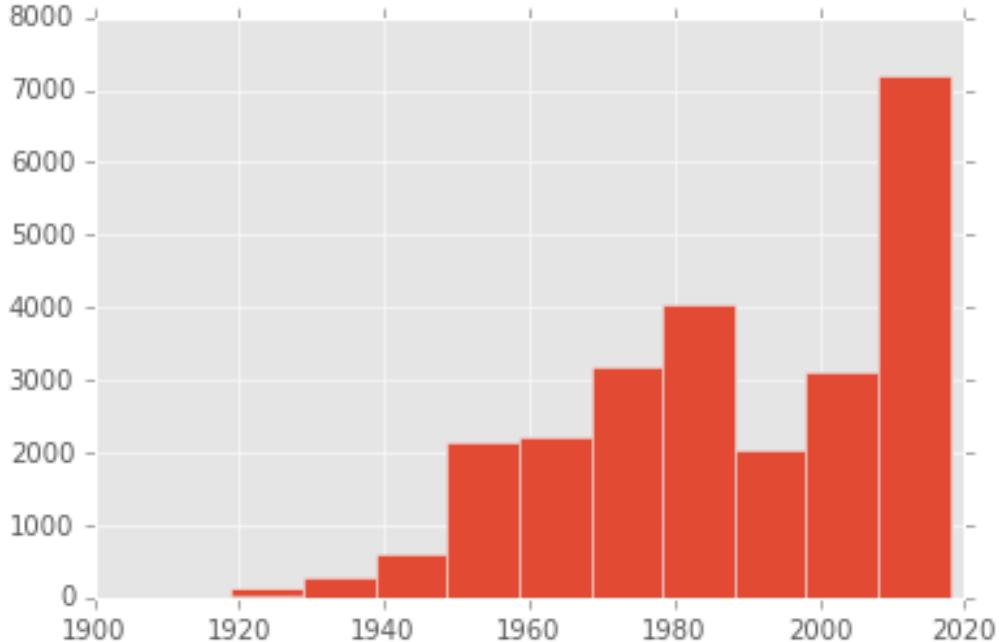


```
In [44]: # Comprovar que os grupos são disjuntos.
(ano2d & ano4d).sum()

Out[44]: 0

In [45]: # Reatribui variável ao DataFrame.
ano.loc[ano2d.index] = ano2d
ano.loc[ano4d.index] = ano4d
dfc['ano'] = ano[~ano.isnull()].astype('int')
```

```
# Verifica como ficou.
hist(dfc.ano[~dfc.ano.isnull()]);
```



1.3 Salvar as características no banco de dados.

```
In [46]: # Converter todas variáveis para inteiro.
for col in dfc.columns:
    dfc[col] = dfc[~dfc[col].isnull()][col].astype('int')

In [47]: from sqlalchemy import create_engine, MetaData
import sqlalchemy

NOME_TABELA = '_var_estrutural'

engine = create_engine(r'postgresql://sergio:1234@localhost/zap')
```

```
meta = sqlalchemy.MetaData(engine, schema='public')
meta.reflect(engine, schema='public')
pssql = z.pd.io.sql.SQLDatabase(engine, meta=meta)

pssql.to_sql(dfc, NOME_TABELA, if_exists='replace', index=True)
z.exec_sql('ALTER TABLE ' + NOME_TABELA +
    ' ADD CONSTRAINT pk_var_estrutural PRIMARY KEY(id);' )

/Users/sergio/anaconda3/envs/py27/lib/python2.7/site-packages/sqlalchemy/dialects/postgresql/base.py:20
name, format_type, default, notnull, domains, enums, schema)
```

Apêndice E

Listagem do IPython Notebook

Variaveis_socioambientais.ipynb

Código de construção das variáveis socioambientais do modelo hedônico.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fgv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

Variaveis_socioambientais

May 16, 2015

1 Variáveis sócioambientais

Este Notebook constrói as variáveis sócioeconômicas selecionadas para o modelo de regressão.

1.1 Variável ‘Média de Anos de Estudo por Bairro’

```
In [1]: import zap_util as z
         from xlrd import open_workbook

In [2]: b = z.get('select gid,nome_s as nome from bairro', id_=None)
         wb = open_workbook('../armazen_dados_rj/488.xls', encoding_override=True)
         ws = wb.sheet_by_index(0)
         nomes = ws.col_values(0,1)
         anos = ws.col_values(1,1)
         df = z.pd.DataFrame()
         df['nome'] = [str.strip(z.remove_acento(n)) for n in nomes]
         df['se_anos_estudo'] = anos

         print 'Tabela bairro tem {} itens e arquivo tem {} itens.'.format(len(b), len(df))

Tabela bairro tem 161 itens e arquivo tem 158 itens.

In [3]: # Bairros no banco de dados mas não no arquivo.
         set(b.nome.tolist()) - set(df.nome.tolist())

Out[3]: {'Freguesia _Ilha_',
         'Freguesia _Jacarepagua_',
         'Gericino',
         'Lapa',
         'Osvaldo Cruz',
         'Vasco da Gama'}

In [4]: # Bairros no arquivo mas não no banco de dados.
         set(df.nome.tolist()) - set(b.nome.tolist())

Out[4]: {'Freguesia (Ilha)', 'Freguesia (Jacarepagua)', 'Osvaldo Cruz'}

In [5]: # Ajustar nomes do arquivo para serem igual ao do banco de dados.
         df.loc[df.nome == 'Freguesia (Ilha)', 'nome'] = 'Freguesia _Ilha_'
         df.loc[df.nome == 'Freguesia (Jacarepagua)', 'nome'] = 'Freguesia _Jacarepagua_'
         df.loc[df.nome == 'Osvaldo Cruz', 'nome'] = 'Osvaldo Cruz'

In [6]: # Verificar novamente bairros no banco de dados mas não no arquivo.
         # Esses bairros realmente não estão no arquivo.
         set(b.nome.tolist()) - set(df.nome.tolist())
```

```

Out[6]: {'Gericino', 'Lapa', 'Vasco da Gama'}

In [7]: b = z.pd.merge(b,df,on=['nome'], how='left')

In [8]: # Tratamento dos valores ausentes.

def media(coluna, bairros):
    soma = 0
    for v in bairros:
        soma += b[b.nome==v][coluna].values[0]
    return soma/len(bairros)

# Gericino
b.ix[b.nome=='Gericino','se_anos_estudo'] = \
    media('se_anos_estudo',[ 'Bangue', 'Padre Miguel', 'Realengo'])

# Lapa
b.ix[b.nome=='Lapa','se_anos_estudo'] = \
    media('se_anos_estudo',[ 'Centro', 'Santa Teresa', 'Gloria'])

# Vasco da Gama
b.ix[b.nome=='Vasco da Gama', 'se_anos_estudo'] = \
    media('se_anos_estudo', [ 'Sao Cristovao', 'Caju', 'Benfica'])

```

2 Variável ‘Percentual Alfabetização’

```

In [9]: wb = open_workbook('../armazen_dados_rj/3132.xls')
ws = wb.sheet_by_index(0)
nomes = ws.col_values(0,1)
perc_alfabetizacao = ws.col_values(1,1)
df = z.pd.DataFrame()
df['nome'] = [str.strip(z.remove_acento(n)) for n in nomes]
df['se_perc_alfabetizacao'] = perc_alfabetizacao

print 'Tabela bairro tem {} itens e arquivo tem {} itens.'.format(len(b), len(df))

```

Tabela bairro tem 161 itens e arquivo tem 160 itens.

```

In [10]: # Bairros no banco de dados mas não no arquivo.
set(b.nome.tolist()) - set(df.nome.tolist())

Out[10]: {'Lapa', 'Vila Kosmos'}

In [11]: # Bairros no arquivo mas não no banco de dados.
set(df.nome.tolist()) - set(b.nome.tolist())

Out[11]: {'Vila Cosmos'}

In [12]: # Ajustar nomes do arquivo para serem igual ao do banco de dados.
df.loc[df.nome == 'Vila Cosmos', 'nome'] = 'Vila Kosmos'

In [13]: # Verificar novamente bairros no banco de dados mas não no arquivo.
# Esses bairros realmente não estão no arquivo.
set(b.nome.tolist()) - set(df.nome.tolist())

```

```

Out[13]: {'Lapa'}

In [14]: b = z.pd.merge(b,df,on=['nome'], how='left')

In [15]: # Tratamento dos valores ausentes.

    # Lapa
    b.ix[b.nome=='Lapa','se_perc_alfabetizacao'] = \
        media('se_perc_alfabetizacao',['Centro', 'Santa Teresa', 'Gloria'])

```

3 Variável ‘Rendimento Nominal Médio’

```

In [16]: wb = open_workbook('../armazen_dados_rj/3151.xls')
    ws = wb.sheet_by_index(0)
    nomes = ws.col_values(0,1)
    rend = ws.col_values(1,1)
    df = z.pd.DataFrame()
    df['nome'] = [str.strip(z.remove_acento(n)) for n in nomes]
    df['se_renda'] = rend

    print 'Tabela bairro tem {} itens e arquivo tem {} itens.'.format(len(b), len(df))

Tabela bairro tem 161 itens e arquivo tem 160 itens.

In [17]: # Bairros no banco de dados mas não no arquivo.
    set(b.nome.tolist()) - set(df.nome.tolist())

Out[17]: {'Lapa', 'Sao Cristovao'}

In [18]: # Bairros no arquivo mas não no banco de dados.
    set(df.nome.tolist()) - set(b.nome.tolist())

Out[18]: {'Imperial de Sao Cristovao'}

In [19]: # Ajustar nomes do arquivo para serem igual ao do banco de dados.
    df.loc[df.nome == 'Imperial de Sao Cristovao', 'nome'] = 'Sao Cristovao'

In [20]: # Verificar novamente bairros no banco de dados mas não no arquivo.
    # Esses bairros realmente não estão no arquivo.
    set(b.nome.tolist()) - set(df.nome.tolist())

Out[20]: {'Lapa'}

In [21]: b = z.pd.merge(b,df,on=['nome'], how='left')

In [22]: # Tratamento dos valores ausentes.

    # Lapa
    b.ix[b.nome=='Lapa','se_renda'] = \
        media('se_renda',['Centro', 'Santa Teresa', 'Gloria'])

```

4 Variável ‘Saneamento’.

```

In [23]: wb = open_workbook('../armazen_dados_rj/3169.xls')
    ws = wb.sheet_by_index(0)
    nomes = ws.col_values(0,1)

```

```

sanea = ws.col_values(1,1)
df = z.pd.DataFrame()
df['nome'] = [str.strip(z.remove_acento(n)) for n in nomes]
df['se_saneamento'] = sanea

print 'Tabela bairro tem {} itens e arquivo tem {} itens.'.format(len(b), len(df))

Tabela bairro tem 161 itens e arquivo tem 161 itens.

In [24]: # Bairros no banco de dados mas não no arquivo.
          set(b.nome.tolist()) - set(df.nome.tolist())

Out[24]: set()

In [25]: # Bairros no arquivo mas não no banco de dados.
          set(df.nome.tolist()) - set(b.nome.tolist())

Out[25]: set()

In [26]: # Arquivo não tem dados válidos para Lapa.
          df.loc[df.nome == 'Lapa', 'se_saneamento'] = None
          df['se_saneamento'] = df['se_saneamento'].astype(float)

In [27]: b = z.pd.merge(b,df,on=['nome'], how='left')

In [28]: # Tratamento dos valores ausentes.

          # Lapa
          b.ix[b.nome=='Lapa', 'se_saneamento'] = \
              media('se_saneamento',[ 'Centro', 'Santa Teresa', 'Gloria'])

```

5 Variável IDH

```

In [29]: wb = open_workbook('..../armazen_dados_rj/1172.xls')
          ws = wb.sheet_by_index(0)
          nomes = ws.col_values(0,1)
          idh = ws.col_values(1,1)
          df = z.pd.DataFrame()
          df['nome'] = [str.strip(z.remove_acento(n)) for n in nomes]
          df['se_idh'] = idh

          print 'Tabela bairro tem {} itens e arquivo tem {} itens.'.format(len(b), len(df))

Tabela bairro tem 161 itens e arquivo tem 159 itens.

In [30]: # Bairros no banco de dados mas não no arquivo.
          set(b.nome.tolist()) - set(df.nome.tolist())

Out[30]: {'Gericino', 'Lapa'}

In [31]: # Bairros no arquivo mas não no banco de dados.
          set(df.nome.tolist()) - set(b.nome.tolist())

Out[31]: set()

```

```
In [32]: b = z.pd.merge(b,df, on=['nome'], how='left')

In [33]: # Tratamento de valores ausentes.

    # Gericino
    b.ix[b.nome=='Gericino', 'se_idh'] = media('se_idh', ['Bangu', 'Padre Miguel', 'Realengo'])

    # Lapa
    b.ix[b.nome=='Lapa', 'se_idh'] = media('se_idh', ['Centro', 'Santa Teresa', 'Gloria'])

In [34]: # Verificar se restou algum valor nulo.
    b.isnull().sum()

Out[34]: gid          0
         nome         0
         se_anos_estudo 0
         se_perc_alfabetizacao 0
         se_renda        0
         se_saneamento   0
         se_idh          0
         dtype: int64
```

6 Índices de Criminalidade

Os índices de criminalidade foram obtidos do Instituto de Segurança Pública do RJ pelo notebook [Capturar Info Bairros](#).

```
In [35]: # AISPs da cidade do Rio de Janeiro.
aisp_idx = [2,3,4,5,6,9,14,16,17,18,19,22,23,27,31,40,41]

from xlrd import open_workbook, cellname

# Repositório dos arquivos baixados.
DIR = './../../../instituto_seguranca_publica/xls/'

# Construir nomes dos arquivos.
x = ['2014{:0>2}'.format(i) for i in range(1,11)]
args = ['2013{:0>2}'.format(i) for i in range(10,13)]+ x

# Helper para indexar coluna no Excel por sua letra.
def colindex(colname):
    cols_ = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
    return cols_.index(colname)

# Índices de criminalidade por AISP.
indices = {}

# Carregar os índices a partir dos arquivos.
for arg in args[:]:
    wb = open_workbook(DIR+arg+'.xls')

    for i in aisp_idx[:]:
```

```

crimes = indices.setdefault(i, {})
try:
    ws = wb.sheet_by_name(u'Aisp {}'.format(i))
except:
    ws = wb.sheet_by_name(u'Aisp{}'.format(i))
col = colindex('S')
crimes['se.crm_violento'] = crimes.setdefault('se.crm_violento',0) + \
    sum(ws.col_values(col, 6, 11))
crimes['se.crm_roubo'] = crimes.setdefault('se.crm_roubo',0) + \
    sum(ws.col_values(col, 22, 33))

del ws
del wb

df_aisp = z.pd.DataFrame(indices).T
df_aisp['aisp'] = df_aisp.index

In [36]: # De <-> para entre AISP e Bairro.
nome = './../instituto_seguranca_publica/aisp_bairro.csv'
aisp_bairro = z.pd.DataFrame.from_csv(nome, index_col=None)

# Remover acentos.
aisp_bairro['nome'] = aisp_bairro.nome.apply(z.remove_acento)

In [37]: # Listar bairros no banco de dados mas não de->para.
# As correções necessárias foram feitas.
set(b.nome.tolist()) - set(aisp_bairro.nome.tolist())

Out[37]: set()

In [38]: # Listar bairros no banco de dados mas não de->para.
# As correções necessárias foram feitas.
set(aisp_bairro.nome.tolist()) - set(b.nome.tolist())

Out[38]: set()

In [39]: # Bairros em um único AISP.
g = aisp_bairro.groupby('nome')
s = g.size()
df_bairro = b[['gid','nome']]
nomes_unicoaisp = s[s == 1].index.tolist()
b1 = z.pd.merge(
    df_bairro[df_bairro.nome.isin(nomes_unicoaisp)],
    aisp_bairro, on=['nome'], how='left')
b1aisp = z.pd.merge(b1, df_aisp, on=['aisp'], how='left')
del b1aisp['aisp'],b1

In [40]: # Calcular média dos indicadores para bairros em mais de um AISP.

data = []
for i in s[s>1].index.tolist():
    aispids = g.get_group(i).aisp.unique().tolist()
    data.append({'nome':i,
        'gid':b[(b.nome==i)].index.tolist()[0],
        'se.crm_roubo':df_aisp.loc[aispids].mean().se.crm_roubo,
        'se.crm_violento':df_aisp.loc[aispids].mean().se.crm_violento
    })

```

```
})
```

```
b2_aisp = z.pd.DataFrame(data)
```

```
In [41]: # Juntar tudo para salvar no banco.  
df = z.pd.concat([b1_aisp, b2_aisp])  
df.index = df.gid  
del df['gid']  
df
```

```
Out[41]:
```

gid	nome	se_crm_roubo	se_crm_violento
1	Paqueta	4871.753940	1730.203982
4	Galeao	506.370607	528.446144
8	Campo Grande	548.152777	750.171811
15	Jardim Guanabara	506.370607	528.446144
25	Pitangueiras	506.370607	528.446144
20	Bangu	1422.348026	630.102144
33	Parque Anchieta	2138.462156	538.945453
28	Penha	1247.621093	539.180020
30	Ribeira	506.370607	528.446144
41	Mare	1005.882404	548.783181
42	Vila Militar	1422.348026	630.102144
46	Realengo	1422.348026	630.102144
43	Cidade Universitaria	506.370607	528.446144
50	Paciencia	374.149360	724.667841
53	Vaz Lobo	2121.794262	724.358959
63	Santa Cruz	374.149360	724.667841
67	Campo dos Afonsos	1422.348026	630.102144
70	Engenheiro Leal	2121.794262	724.358959
79	Campinho	2121.794262	724.358959
90	Inhoaiba	548.152777	750.171811
106	Taquara	646.448508	764.408423
116	Santa Teresa	4871.753940	1730.203982
124	Flamengo	1180.675882	423.449787
127	Vargem Grande	1167.358492	644.473336
128	Alto da Boa Vista	1124.097257	588.673608
126	Guaratiba	374.149360	724.667841
134	Sepetiba	374.149360	724.667841
140	Jardim Botanico	777.602554	567.540861
132	Urca	1180.675882	423.449787
136	Jacarepagua	646.448508	764.408423
..
114	Gloria	1180.675882	423.449787
115	Rio Comprido	1740.699949	715.936741
117	Estacio	1740.699949	715.936741
119	Catumbi	1740.699949	715.936741
120	Grajau	1124.097257	588.673608
121	Pechincha	646.448508	764.408423
122	Andarai	1124.097257	588.673608
123	Catete	1180.675882	423.449787
125	Laranjeiras	1180.675882	423.449787
129	Cosme Velho	1180.675882	423.449787

```

133      Cidade de Deus    646.448508    764.408423
130          Curicica    646.448508    764.408423
139      Gardenia Azul    646.448508    764.408423
131      Botafogo    1180.675882    423.449787
135          Anil    646.448508    764.408423
137      Camorim    1167.358492    644.473336
138      Humaita    1180.675882    423.449787
141      Vargem Pequena    1167.358492    644.473336
144          Lagoa    777.602554    567.540861
151          Rocinha    777.602554    567.540861
148          Leblon    777.602554    567.540861
152      Pedra de Guaratiba    374.149360    724.667841
161          Lapa    4871.753940   1730.203982
159          Deodoro    1422.348026    630.102144
160          Gericino    1422.348026    630.102144
3          Bancarios    506.370607    528.446144
143          Leme    809.532657    805.917270
20          Centro    3306.226945   3306.226945
71          Colegio    2130.128209   2130.128209
23          Tijuca    1432.398603   1432.398603

```

[161 rows x 3 columns]

7 Salvar variáveis sócio economicas no banco de dados.

```
In [42]: # Salvar variáveis de bairros no banco de dados
b = z.pd.merge(b,df,on=['nome'], how='left')
b.describe()
```

```
Out[42]:
      gid  se_anos_estudo  se_perc_alfabetizacao  se_renda \
count  161.000000      161.000000      161.000000  161.000000
mean   81.000000       8.352174      97.311573  1931.945528
std    46.620811      2.154657      1.923924  1459.179621
min    1.000000      3.880000      85.330000  571.010000
25%   41.000000      6.950000      96.630000  1085.850000
50%   81.000000      7.800000      97.680000  1395.590000
75%  121.000000      9.810000      98.510000  2103.470000
max   161.000000     13.890000     100.000000  8286.460000

      se_saneamento  se_idh  se_crm_roubo  se_crm_violento
count  161.000000  161.000000  161.000000  161.000000
mean   0.912236   0.840027  1455.035931  686.739609
std    0.143975   0.065726  887.606510  337.135620
min    0.000000   0.711408  374.149360  423.449787
25%   0.905568   0.798364  777.602554  539.180020
50%   0.963885   0.838543  1422.348026  577.233233
75%   0.991115   0.877972  1740.699949  724.358959
max   0.999774   0.970435  4871.753940  3306.226945
```

```
In [43]: # Salva no banco de dados.
z.d.salva_dataframe(b, '_var_socioecon')
```

Apêndice F

Listagem do IPython Notebook

Tra-

tar_variaveis_estruturais.ipynb

Código de construção das variáveis socioambientais do modelo hedônico.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fgv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

Tratar_variaveis_estruturais

June 4, 2015

1 Table of Contents

- 1. Tratamento das variáveis estruturais
- 2. Variáveis Longitude e Latitude
- 3. Verificar e remover outliers.
 - 3.1 Variável ‘quartos’
 - 3.2 Característica ‘área’
 - 3.3 Característica ‘garagem’
 - 3.4 Característica ‘suites’
 - 3.5 Característica ‘m2’
 - 3.6 Variável ‘preço’
- 4. Salvar ids dos imóveis resultantes do filtro de outlier.

2 1. Tratamento das variáveis estruturais

Vamos verificar como estão os dados. Primeiro, a view vw_imovel_rio seleciona apenas os imóveis geograficamente na cidade do Rio de Janeiro, logo já elimina os imóveis que não possuem latitude e longitude.

In [1]: %pylab inline

```
import zap_util as z
from scipy.stats import zscore

z.set_style()

Populating the interactive namespace from numpy and matplotlib
```

In [2]: # Selecionar do banco somente imóveis da cidade do
df = z.get('select id,area,preco_ajustado as preco,m2_ajustado as m2,' +
 'garagem,quartos,suites,condominio,lng,lat ' +
 'from vw_imovel_rio inner join preco_ajustado p using (id)')

df.describe()

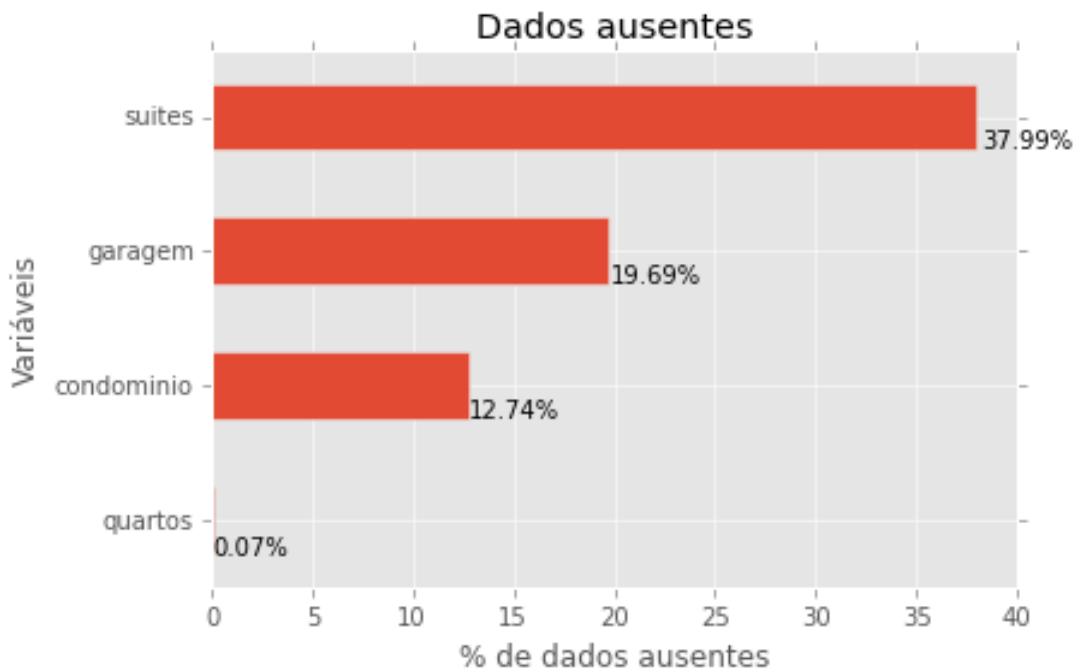
Out[2]:

	area	preco	m2	garagem	quartos	\
count	59412.000000	59412.000000	59412.000000	47713.000000	59369.000000	
mean	123.551404	1389.054776	10.288691	2.446796	2.632889	
std	105.676316	3269.934734	6.489658	41.535622	1.447283	
min	15.000000	45.000000	0.383259	1.000000	1.000000	

25%	70.000000	495.000000	6.041503	1.000000	2.000000
50%	92.000000	840.318168	8.606557	1.000000	3.000000
75%	144.000000	1579.719066	12.500000	2.000000	3.000000
max	12930.000000	653123.640000	320.619343	2011.000000	234.000000
	suites	condominio	lng	lat	
count	36839.000000	51841.000000	59412.000000	59412.000000	
mean	1.435354	22.429195	-43.284459	-22.952593	
std	1.145904	4392.359266	0.097964	0.045369	
min	1.000000	0.001000	-43.721668	-23.066667	
25%	1.000000	0.410000	-43.357212	-22.985393	
50%	1.000000	0.650000	-43.248444	-22.956712	
75%	2.000000	1.000000	-43.197077	-22.923000	
max	150.000000	1000000.000000	-43.157761	-22.786389	

Temos mais de 59000 registros. Vejamos quais informações numéricas estão faltando.

```
In [3]: miss = df.isnull().sum()
miss = miss[miss > 0] # Somente os que não possuem dados.
miss = miss / len(df) * 100 # Percentual
miss.sort()
ax = miss.plot(kind='barh', title='Dados ausentes');
xlabel('% de dados ausentes');
ylabel (u'Variáveis');
for p in ax.patches:
    ax.annotate(str(p.get_width().round(2))+'%', xy=(p.get_width() * 1.01, p.get_y() * 1.005),
savefig('..../texto/img/var_intrins_ausente.png')
```



Trataremos as características ausentes conforme seu tipo. Valores numéricos serão definidos como 0, e valores categóricos conforme o contexto.

In [4]: # Remover os imóveis que não tiveram detalhes obtidos. São doze no total.

```
# Remover imóveis sem quartos.  
df = df[~df.quartos.isnull()]  
  
df.condominio.fillna(value=0,inplace=True)  
df.garagem.fillna(value=0, inplace=True)  
df.suites.fillna(value=0, inplace=True)  
  
# Verificar quais variáveis ainda estão nulas.  
df.isnull().sum()
```

Out[4]:

area	0
preco	0
m2	0
garagem	0
quartos	0
suites	0
condominio	0
lng	0
lat	0

dtype: int64

3 2. Variáveis Longitude e Latitude

As variáveis longitude e latitude, que permitem a geolocalização do imóvel, não são estruturais e sim auxiliares para a determinação das variáveis de acessibilidade. Entretanto, decidimos tratá-las logo para encontrar imóveis com localizações repetidas.

In [5]:

```
antes = len(df)  
lat_antes = df.lat.copy()  
lng_antes = df.lng.copy()
```

In [6]: # Verificar distribuição das coordenadas repetidas.

```
g = df.groupby(['lng','lat'])  
g.size().describe()
```

Out[6]:

count	21830.000000
mean	2.719606
std	11.262730
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	1108.000000

dtype: float64

In [7]: # Iremos filtrar nosso dataset permitindo até 5 repetições (75%).

```
latlng = g.size()[g.size()<=5].index.tolist()  
index_sel = []  
for k in latlng:
```

```

        index_sel.extend(g.groups[k])
df = df.loc[index_sel]

In [8]: depois = len(df)

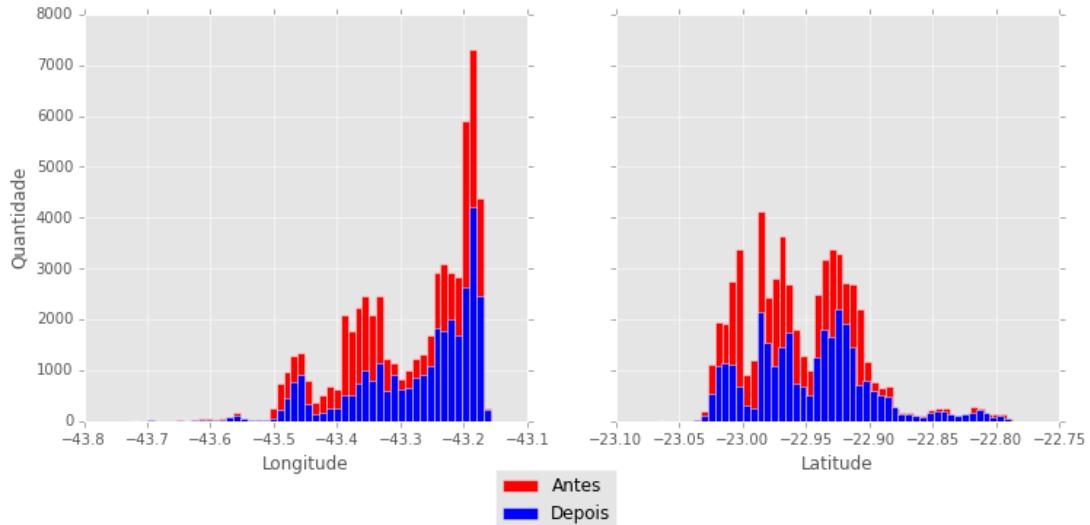
print "Antes: {}. Depois: {}. Excluídos:{}".format(antes,depois,antes-depois)

Antes: 59369. Depois: 31090. Excluídos:28279

In [9]: bins=50;
f,(a1,a2) = subplots(1,2,sharey=True)
f.set_size_inches(12,5)
a1.hist(lng_anter, color='r', bins=bins, label='Antes');
a1.hist(df.lng,color='b',bins=bins, label='Depois');
a2.hist(lat_anter,color='r',bins=bins,label='Antes');
a2.hist(df.lat,color='b',bins=bins, label='Depois');
a1.set_ylabel('Quantidade');
a1.set_xlabel('Longitude');
a2.set_xlabel('Latitude');
lgd= legend( loc = 'lower center', bbox_to_anchor = (0,-0.1,1,1),
bbox_transform = plt.gcf().transFigure );

f.savefig('../texto/img/var_latlng_anter_depois.png',
bbox_extra_artists=(lgd,), bbox_inches='tight')

```



4 3. Verificar e remover outliers.

Usaremos a quantidade em desvios padrão em conjunto com histogramas e boxplot para identificar e remover os outliers que definimos como aqueles acima de 3 desvios padrões.

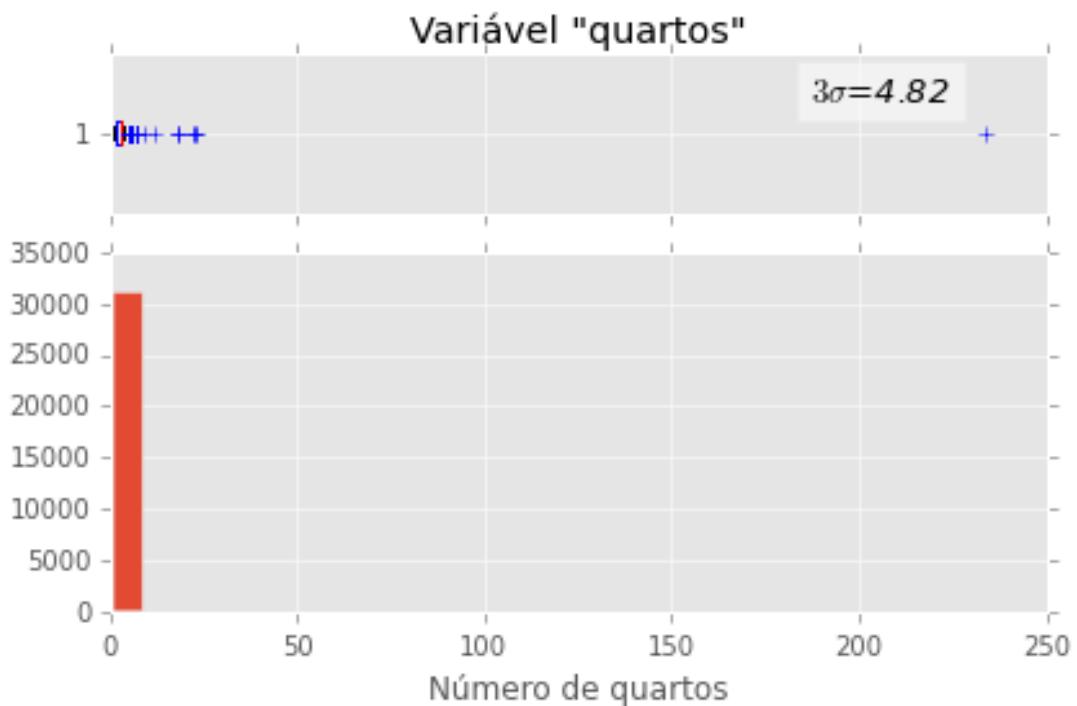
4.1 3.1 Variável ‘quartos’

Representa a quantidade de quartos no imóvel.

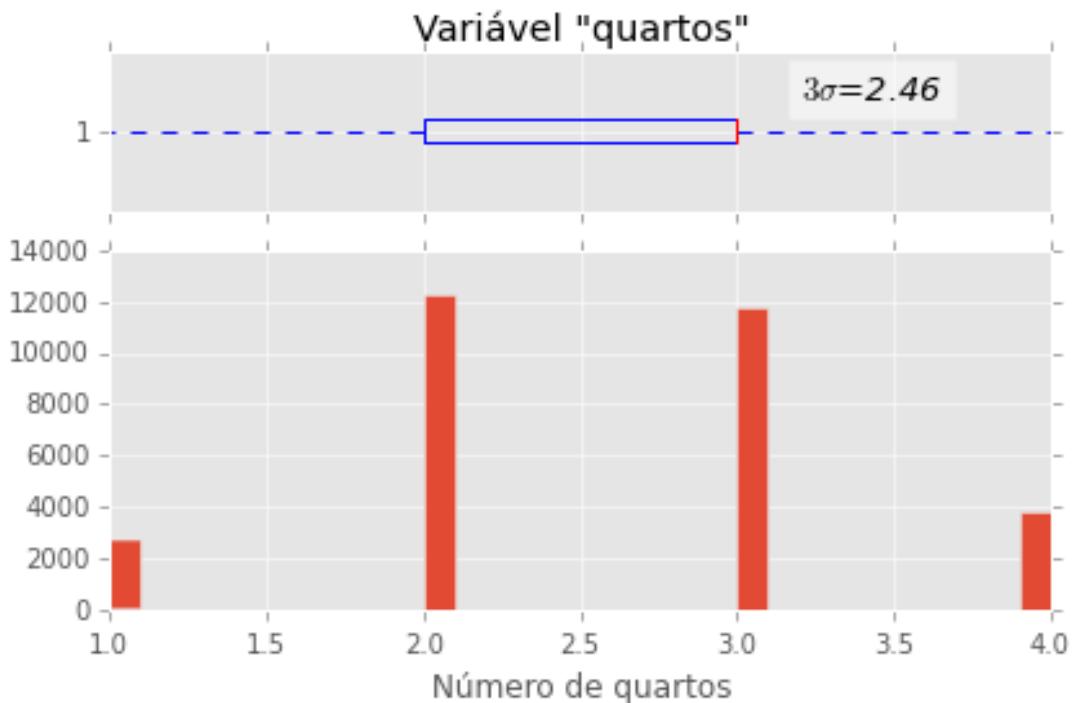
```
In [10]: print 'Total de registros: {}'.format(len(df))
antes = len(df)

Total de registros: 31090

In [11]: reload(z)
z.plot_boxhist(df.quartos, u'Variável "quartos"', u'Número de quartos')
savefig('../../../texto/img/var_quartos_boxhist_antes.png')
```



```
In [12]: # Determinamos arbitrariamente o limite de quartos de interesse no estudo.
limite = 4
df2 = df[df.quartos <= limite]
z.plot_boxhist(df2.quartos, u'Variável "quartos"', u'Número de quartos')
savefig('../../../texto/img/var_quartos_boxhist_depois.png')
```



```
In [13]: depois = len(df2)
      print "Depois:{} . Excluídos:{}" .format(depois,antes-depois)
```

Depois:30547. Excluídos:543

```
In [14]: df = df2
```

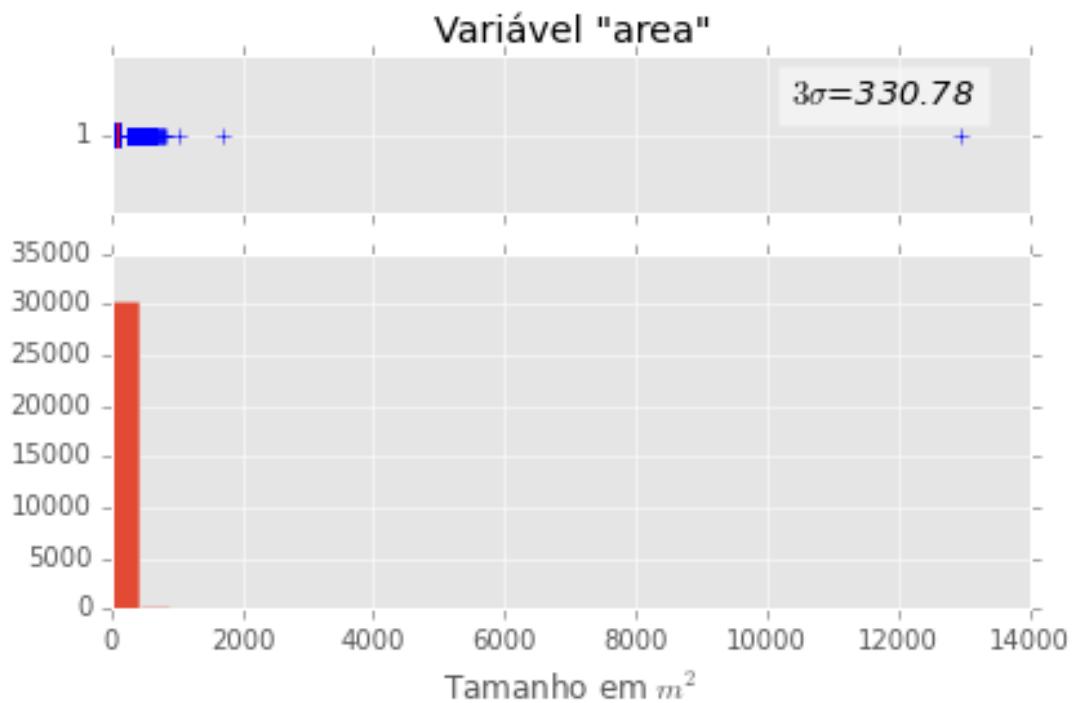
4.2 3.2 Característica ‘area’

Representa a área do imóvel em metros quadrados, m^2 .

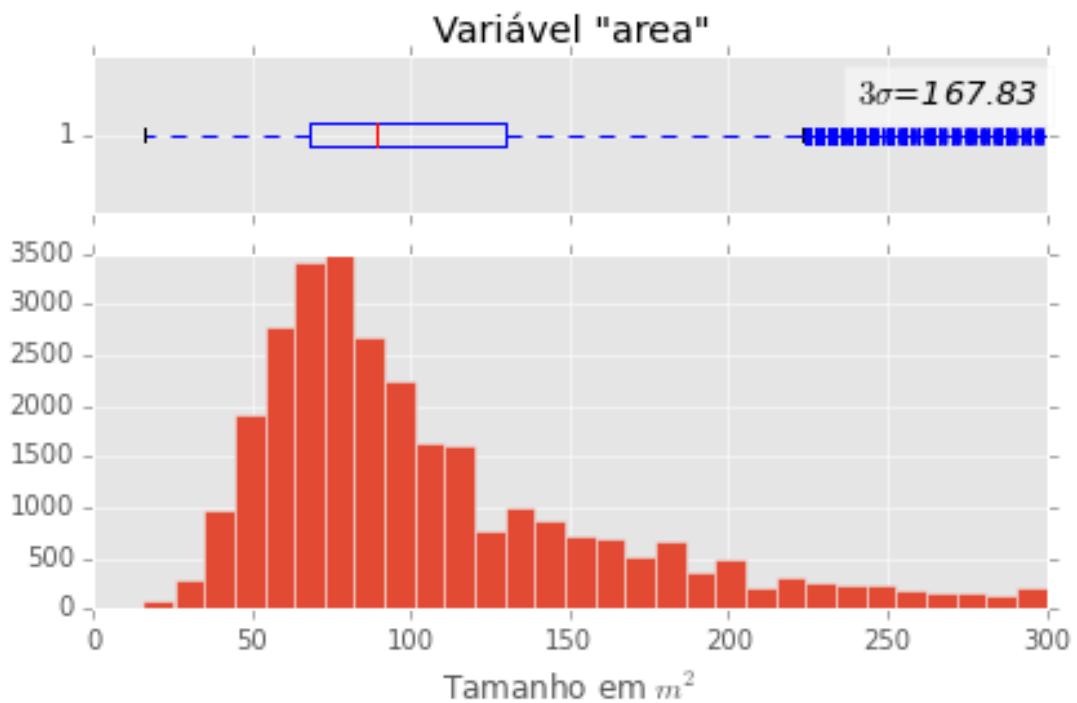
Com base na reportagem <http://g1.globo.com/economia/noticia/2014/09/rio-continua-com-o-metro-quadrado-mais-caro-do-pais-aponta-fipezap.html> do jornal O Globo de 4 de setembro de 2013, mostra que o valor médio do metro quadrado mais caro da cidade é de R\$ 23.613, limitamos os valores de $\$ m^2 \$$ ao máximod de R\$ 30.000,00.

```
In [15]: antes = len(df)
```

```
In [16]: z.plot_boxhist(df.area, u'Variável "area"', 'Tamanho em $m^2$')
      savefig('..../texto/img/var_area_boxhist_antes.png')
```



```
In [17]: # Remover imóveis acima de 300 m2.  
limite = 300  
df2 = df[df.area <= limite]  
z.plot_boxhist(df2.area, u'Variável "area"', 'Tamanho em $m^2$')  
savefig('../../../texto/img/var_area_boxhist_depois.png')
```



```
In [18]: depois = len(df2)

        print "Antes: {}. Depois: {}. Excluídos:{}".format(antes,depois,antes-depois)

Antes: 30547. Depois: 29327. Excluídos:1220
```

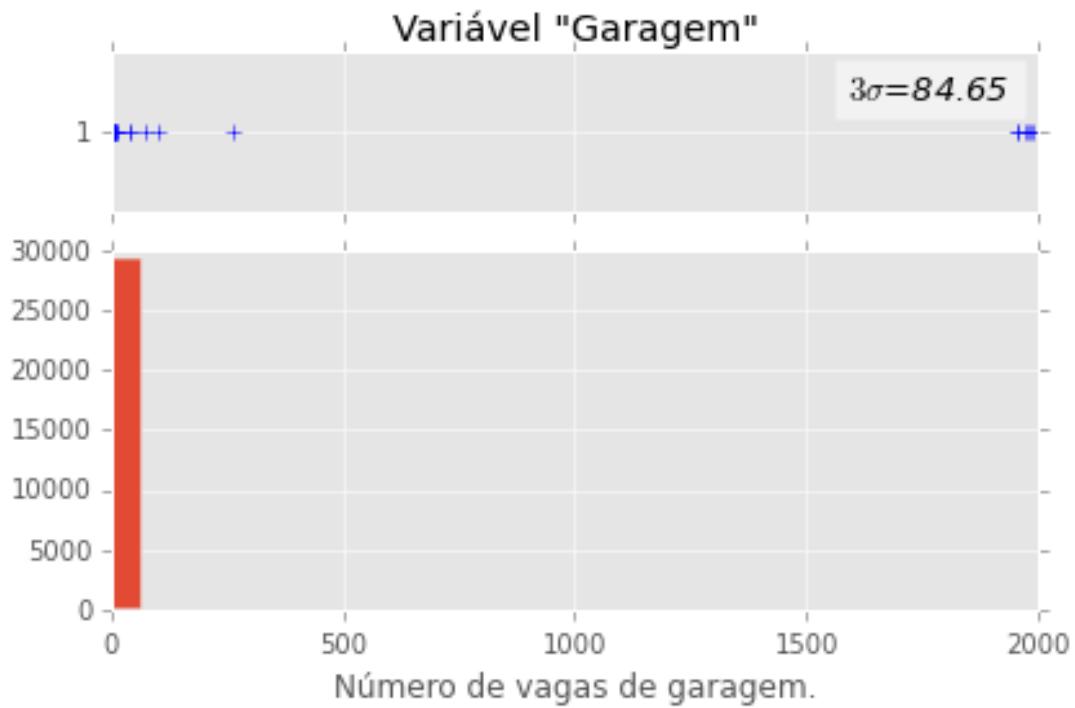
```
In [19]: df = df2
```

4.3 3.3 Característica ‘garagem’

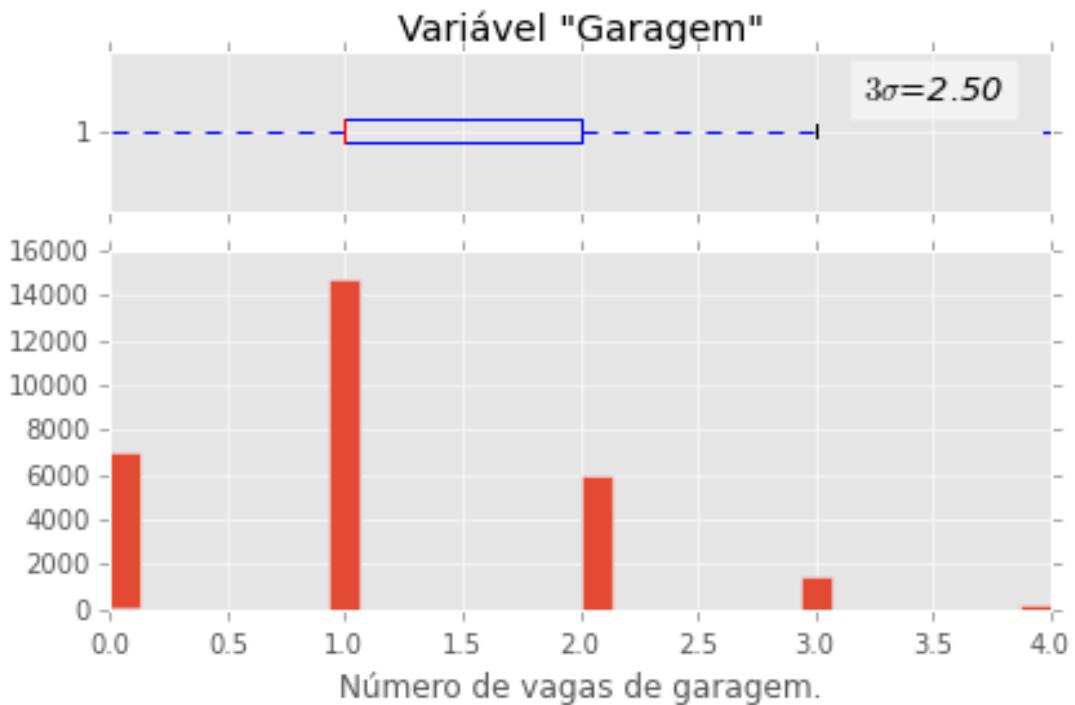
Representa a quantidade de vagas de garagem pertencentes ao imóvel.

```
In [20]: antes = len(df)
```

```
In [21]: z.plot_boxhist(df.garagem, u'Variável "Garagem"', u'Número de vagas de garagem.')
        savefig('..../texto/img/var_garagem_boxhist_antes.png')
```



```
In [22]: limite = 4
df2 = df[df.garagem<=limite]
z.plot_boxhist(df2.garagem, u'Variável "Garagem"', u'Número de vagas de garagem.')
savefig('../../../texto/img/var_garagem_boxhist_depois.png')
```



```
In [23]: depois = len(df2)

print "Antes: {}. Depois: {}. Excluídos:{}".format(antes,depois,antes-depois)

Antes: 29327. Depois: 29296. Excluídos:31
```

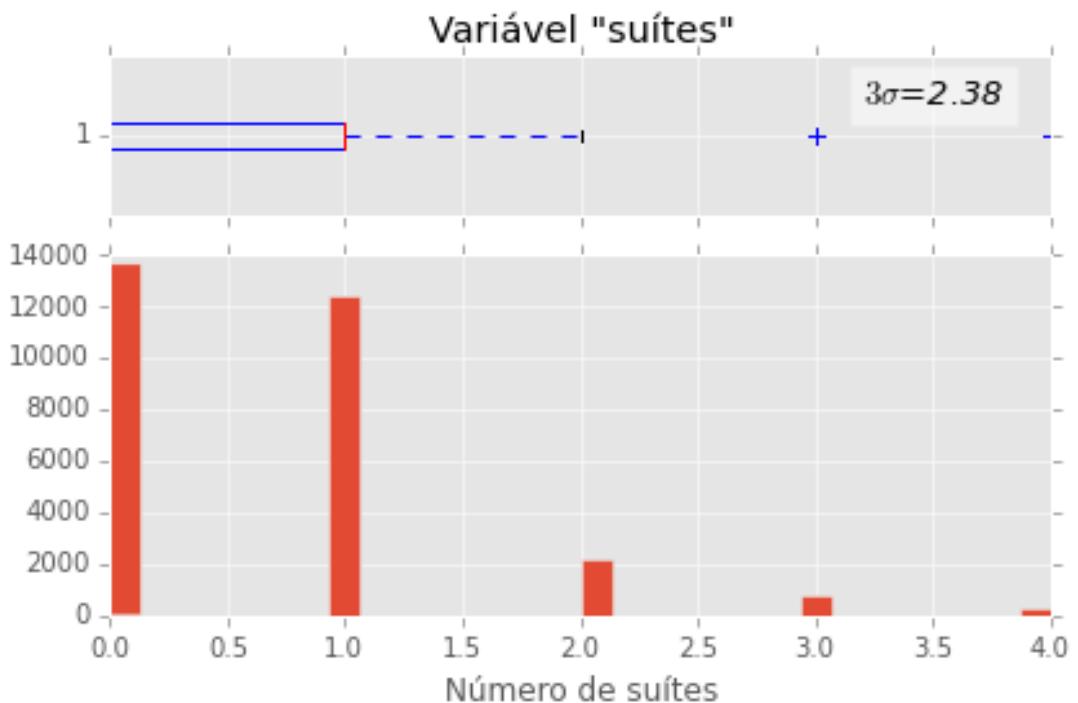
```
In [24]: df = df2
```

4.4 3.4 Característica ‘suites’

Representa a quantidade de suítes, quartos com banheiro próprio.
 Verificamos não existirem outliers.

```
In [25]: antes = len(df)

In [26]: z.plot_boxhist(df.suites, u'Variável "suites"', u'Número de suítes')
         savefig('..../texto/img/var_suites_boxhist_antes.png')
```



```
In [27]: depois = len(df2)

print "Antes: {}. Depois: {}. Excluídos:{}".format(antes,depois,antes-depois)

Antes: 29296. Depois: 29296. Excluídos:0
```

4.5 3.5 Característica ‘m2’

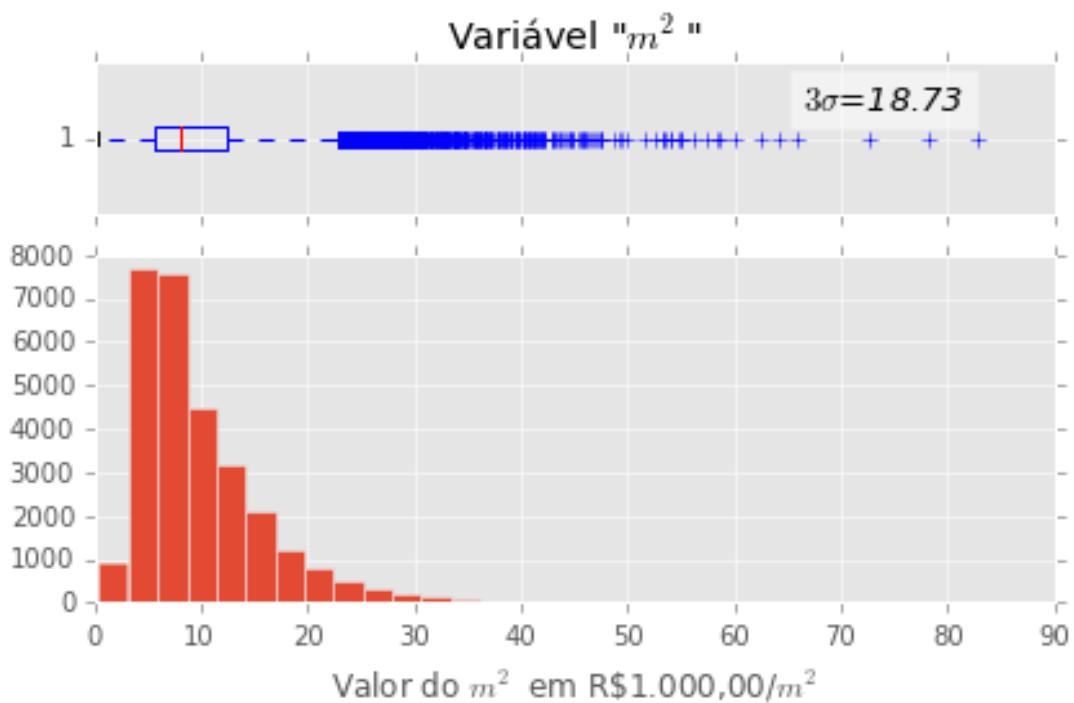
Representa o valor do metro quadrado.

Reportagem do Globo em setembro de 2014 mostra metro quadrado médio por bairro mais caro é inferior a 25000.

<http://g1.globo.com/economia/noticia/2014/09/rio-continua-com-o-metro-quadrado-mais-caro-do-pais-aponta-fipezap.html>

```
In [28]: antes = len(df)
```

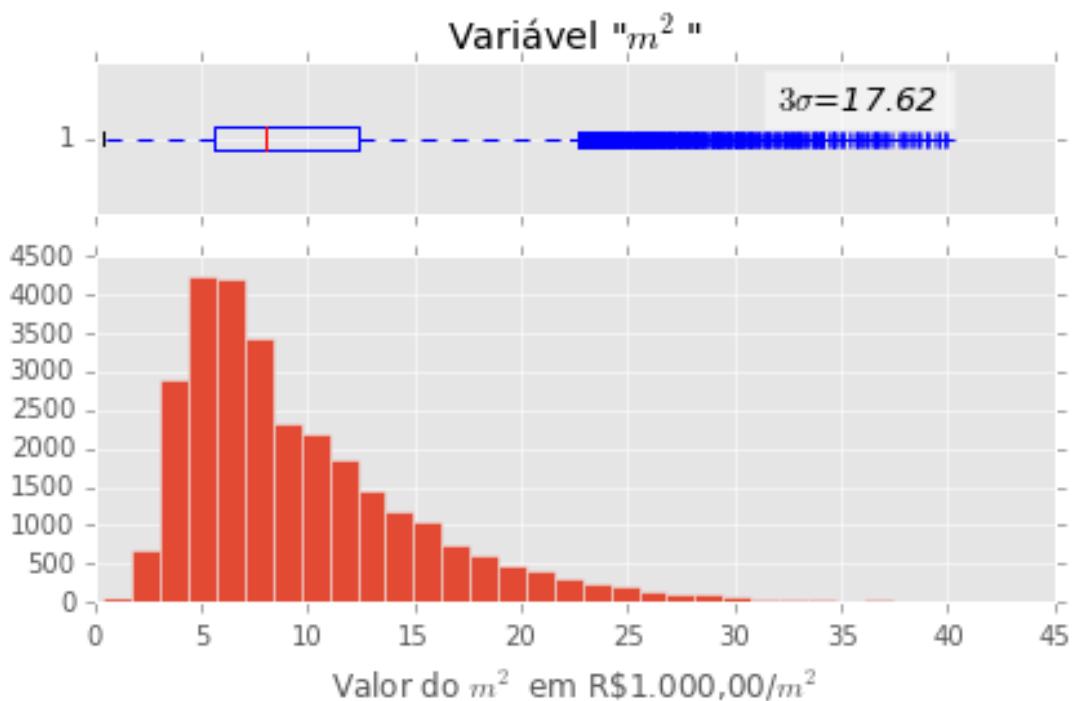
```
In [29]: z.plot_boxhist(df.m2, u'Variável "$m^2$" , 'Valor do $m^2$ em R\$1.000,00/$m^2$')
savefig('..../texto/img/var_m2_boxhist_antes.png')
```



```
In [30]: # Com base na reportagem, definimos o limite de R$ 40 mil/m2.
limite = 40

df2 = df[df.m2 <= limite]

z.plot_boxhist(df2.m2, u'Variável "$m^2$" , 'Valor do $m^2$ em R\$1.000,00/$m^2')
savefig('..../texto/img/var_m2_boxhist_depois.png')
```



```
In [31]: depois = len(df2)

        print "Antes: {}. Depois: {}. Excluídos:{}".format(antes,depois,antes-depois)

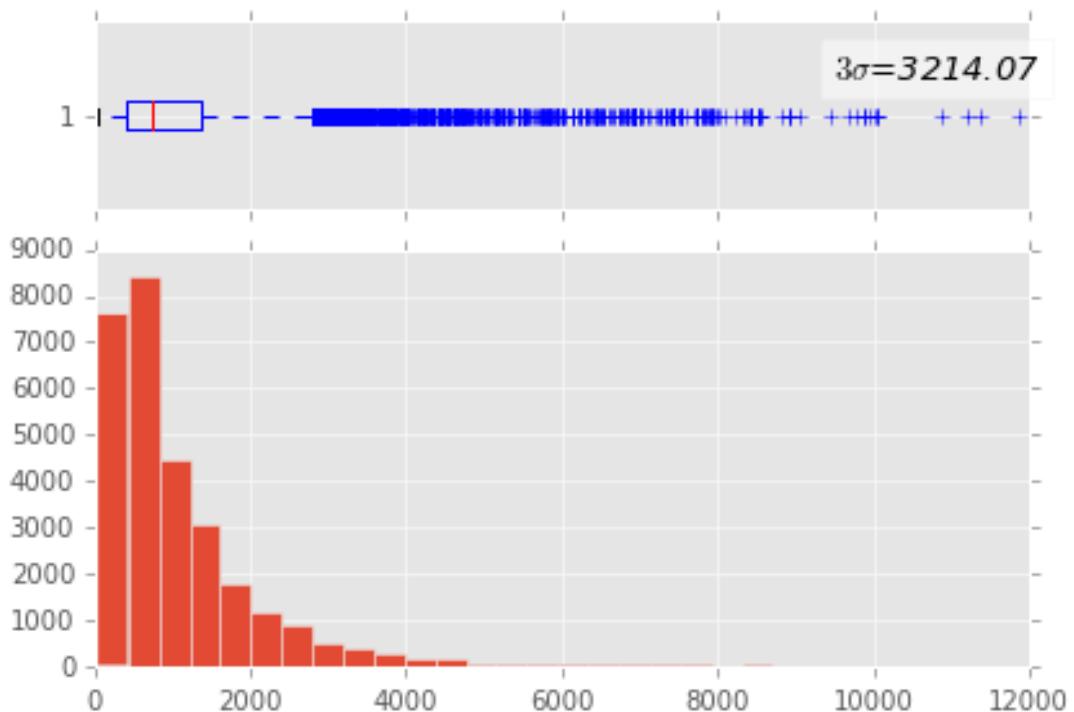
Antes: 29296. Depois: 29204. Excluídos:92
```

```
In [32]: df = df2
```

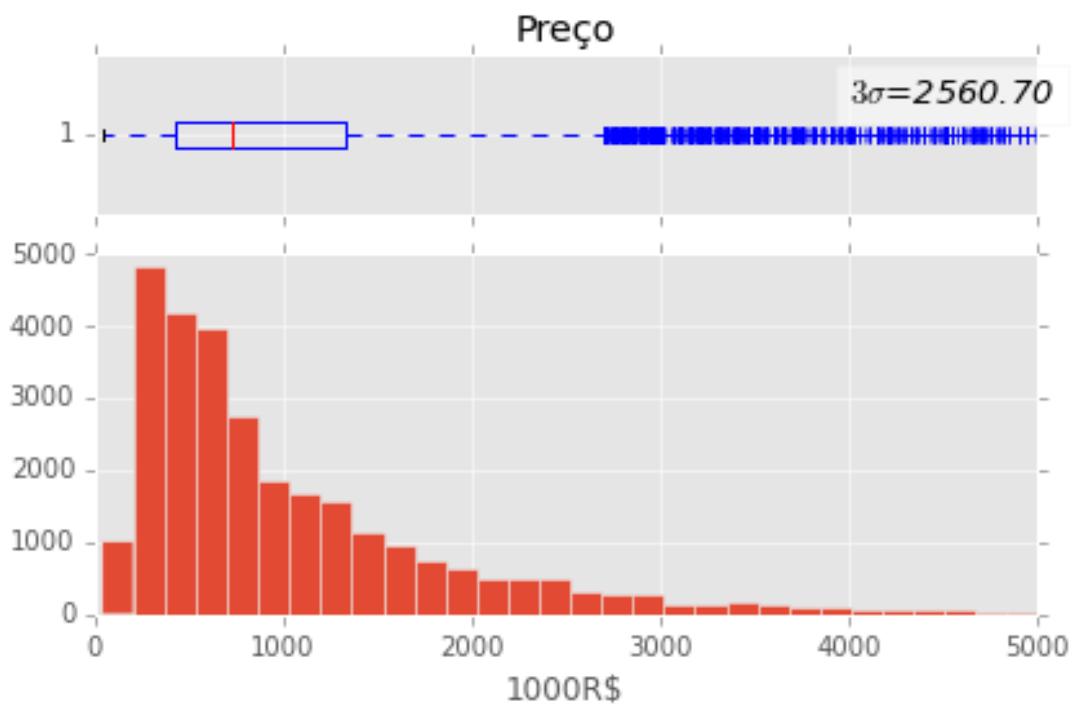
4.6 3.6 Variável ‘preço’

```
In [33]: antes = len(df)

In [34]: z.plot_boxhist(df.preco)
        savefig('..../texto/img/var_preco_boxhist_antes.png')
```



```
In [35]: df2 = df[df.precio <= 5000]
z.plot_boxhist(df2.precio, u'Preço', u'1000R\$');
savefig('..../texto/img/var_precio_boxhist_depois.png')
```



```
In [36]: depois = len(df2)

print "Antes: {}. Depois: {}. Excluídos:{}".format(antes,depois,antes-depois)

Antes: 29204. Depois: 28819. Excluídos:385

In [37]: df = df2
```

5 4. Salvar ids dos imóveis resultantes do filtro de outlier.

Presumimos que o conjunto de dados esteja livre de outliers. Desta forma iremos salvar os ids dos imóveis selecionados para utilização futura.

```
In [38]: # Salvar ids selecionados no banco de dados, usando copy_from
# para inserir uma grande quantidade de dados.
# A instrução condicional 'if' serve apenas para não executar
# esse procedimento por acidente.
if False:
    z.exec_sql('delete from imovel_sel ')
    f = open('./imovel_sel.txt', 'w')
    df.to_csv(f,index=True,columns=[],sep="\t")
    f.close()
    f = open('./imovel_sel.txt', 'r')
    con = z.d.conecta_db()
    con.cursor().copy_from(f, 'imovel_sel',columns=['id'])
    con.commit()
    f.close()
    z.exec_sql('refresh materialized view vw_imovel')
```

Apêndice G

Verificação das variáveis de acessibilidade

As variáveis de acessibilidade de distâncias a pontos de interesse foram construídas de forma determinística. Os imóveis e pontos de interesse foram verificados dentro dos limites da cidade do Rio de Janeiro. As visualizações a seguir demonstram a ausência de *outliers*.

Distância ao centro da cidade do Rio de Janeiro

A distribuição da distância euclidiana ao centro da cidade do Rio de Janeiro, definido na página 48, pode ser observada na figura a seguir:

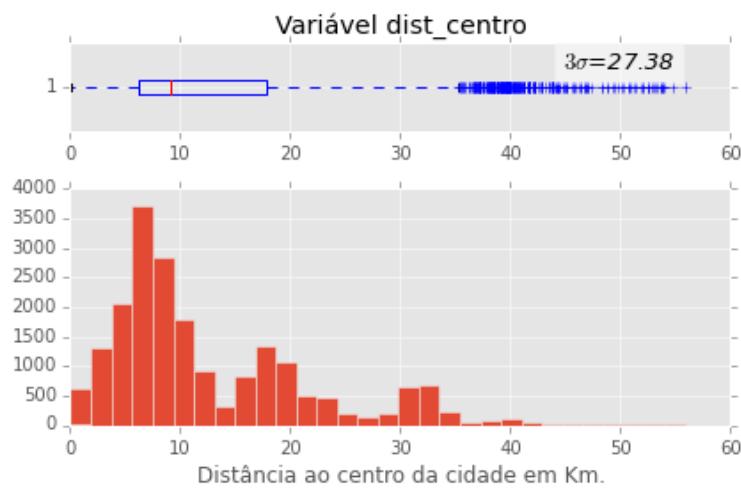


Figura G.1: Boxplot e histograma da variável *dist_centro*

Distância à Delegacia de Polícia Civil

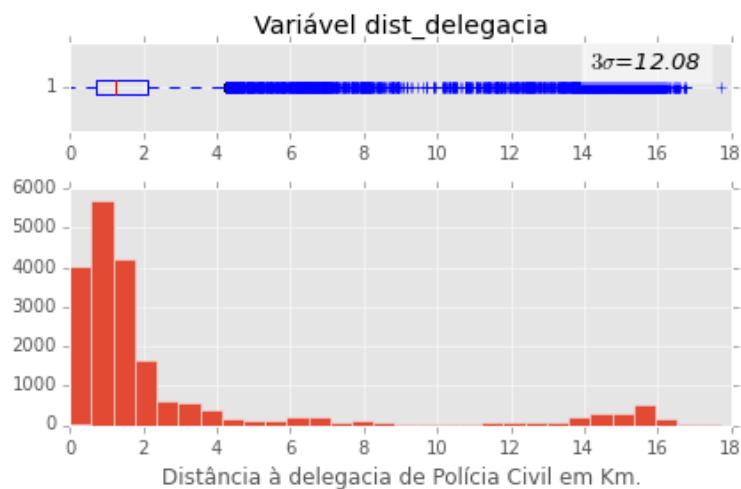
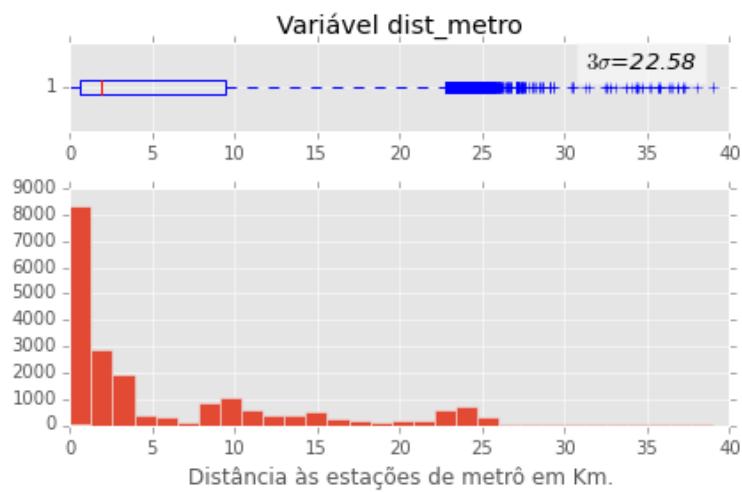
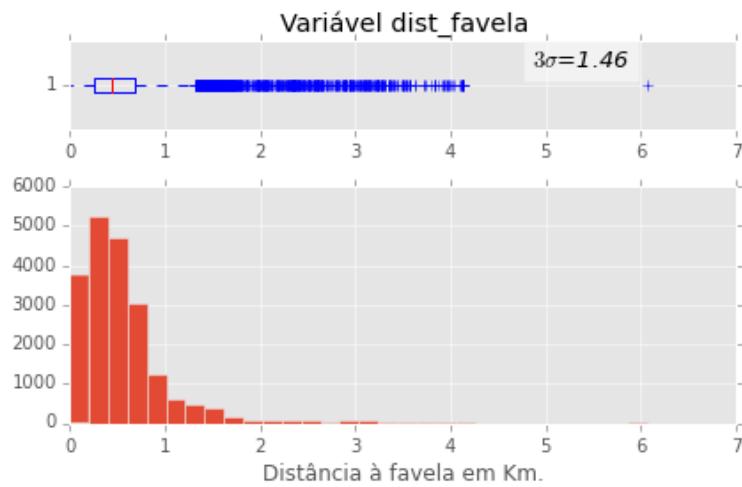


Figura G.2: Boxplot e histograma da variável *dist_delegacia*

Distância à estação de metrôFigura G.3: Boxplot e histograma da variável *dist_metro***Distância à favela**Figura G.4: Boxplot e histograma da variável *dist_favela*

Distância à lagoa

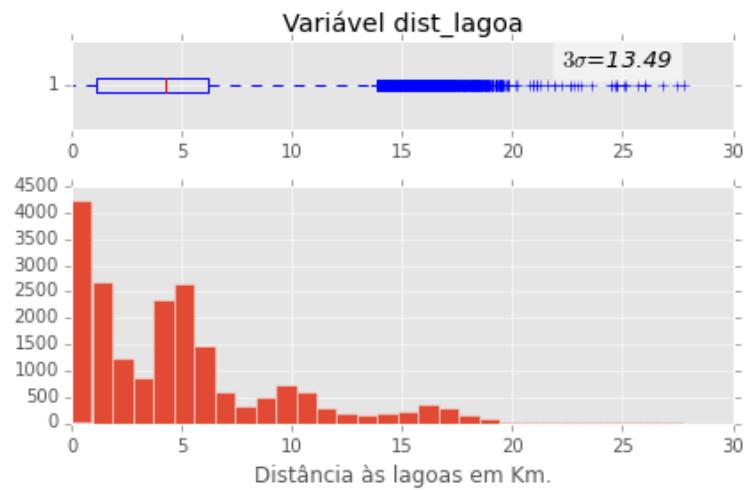


Figura G.5: Boxplot e histograma da variável *dist_lagoa*

Distância à praia

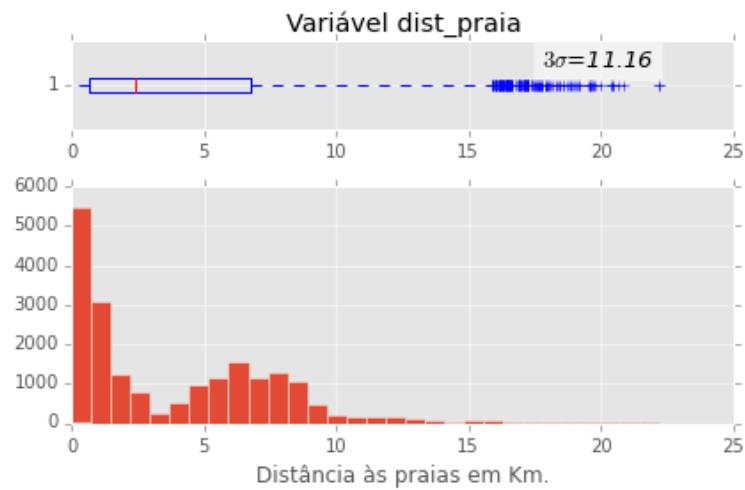
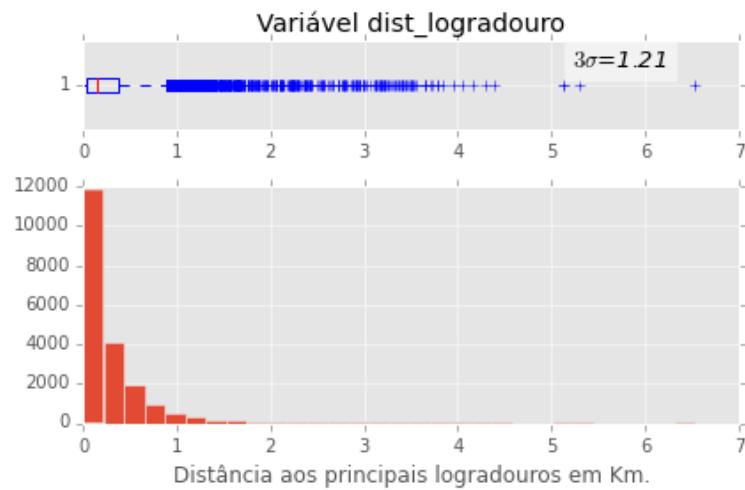
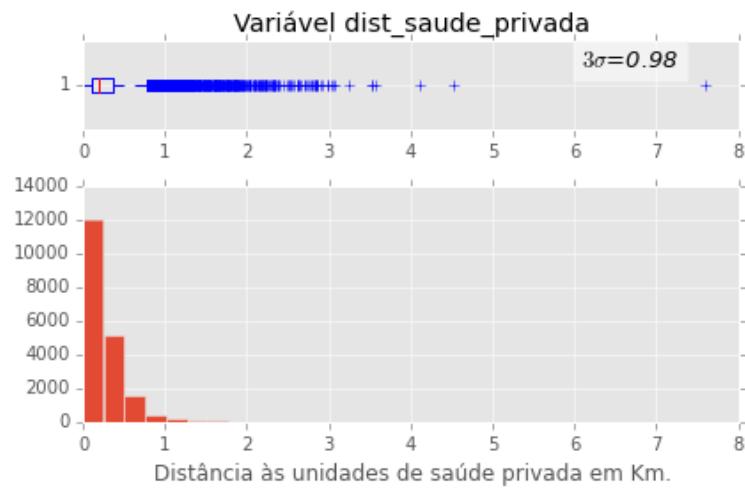


Figura G.6: Boxplot e histograma da variável *dist_praia*

Distância à principal logradouroFigura G.7: Boxplot e histograma da variável *dist_logradouro***Distância à unidade de saúde privada**Figura G.8: Boxplot e histograma da variável *dist_saude_privada*

Distância à unidade de saúde pública

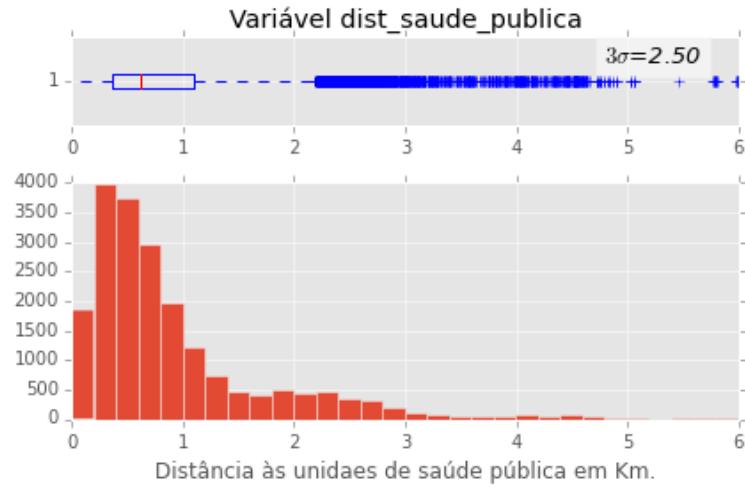


Figura G.9: Boxplot e histograma da variável *dist_saude_publica*

Distância à unidade do Corpo de Bombeiros

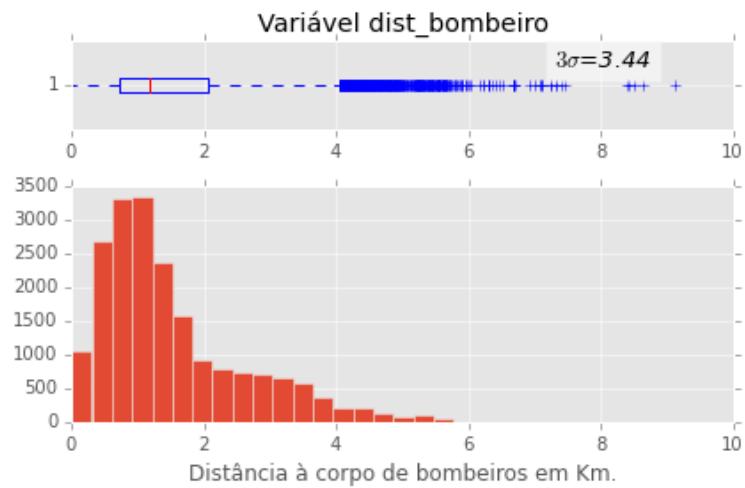


Figura G.10: Boxplot e histograma da variável *dist_bombeiro*

Apêndice H

Verificação das variáveis socioambientais

Média de anos de estudo

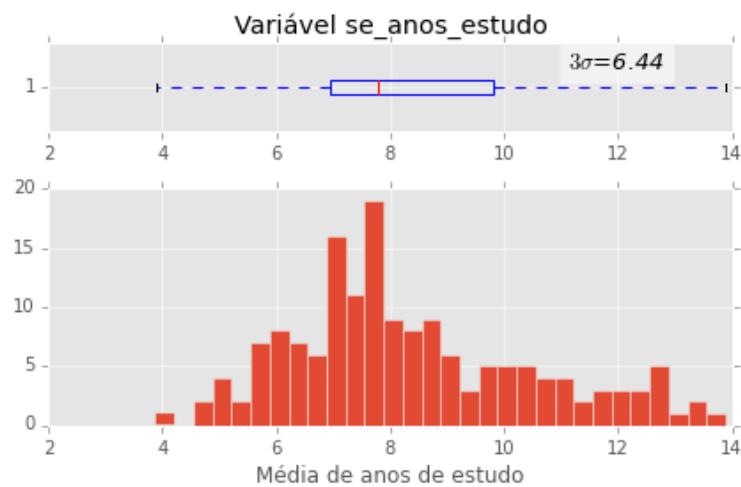


Figura H.1: Boxplot e histograma da variável *se_anos_estudo*

Percentual de alfabetização

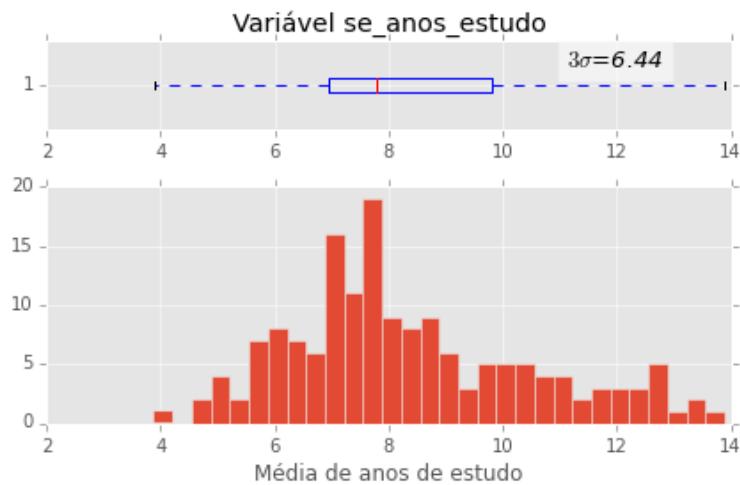


Figura H.2: Boxplot e histograma da variável *se_anos_estudo*

Percentual de alfabetização

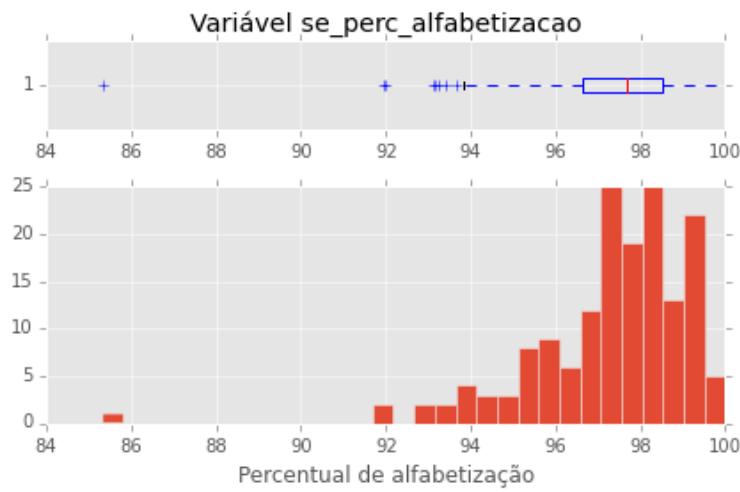
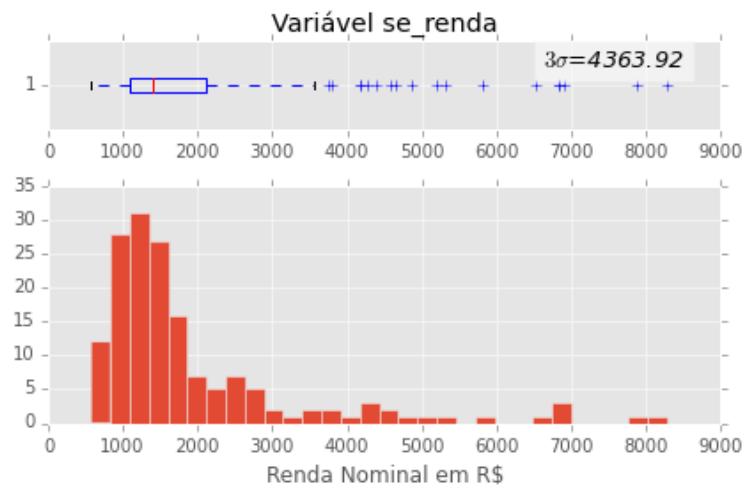
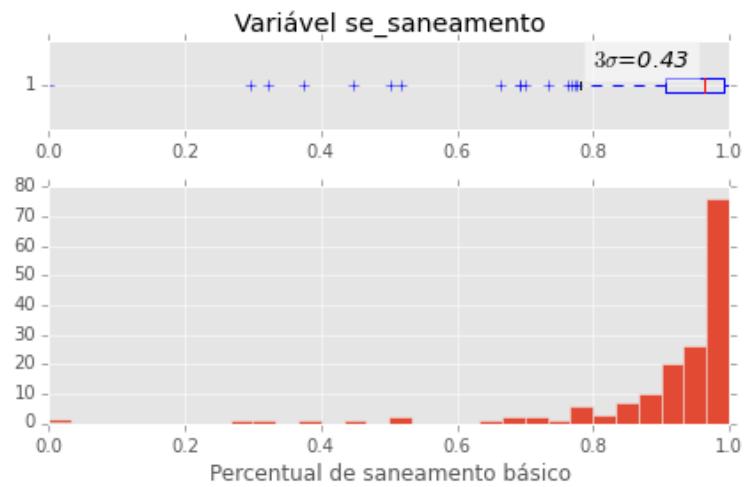


Figura H.3: Boxplot e histograma da variável *se_perc_alfabetizacao*

Renda nominalFigura H.4: Boxplot e histograma da variável *se_renda***Saneamento**Figura H.5: Boxplot e histograma da variável *se_saneamento*

Percentual de alfabetização

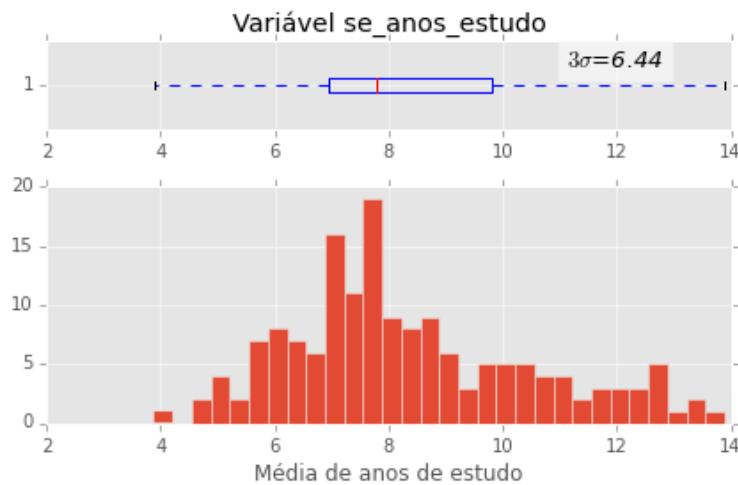


Figura H.6: Boxplot e histograma da variável *se_anos_estudo*

Índice de Desenvolvimento Humano

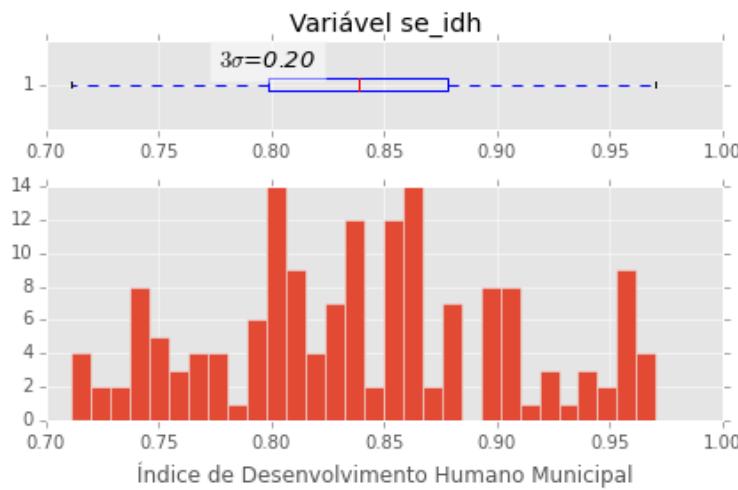


Figura H.7: Boxplot e histograma da variável *se_idh*

Crimes de roubo

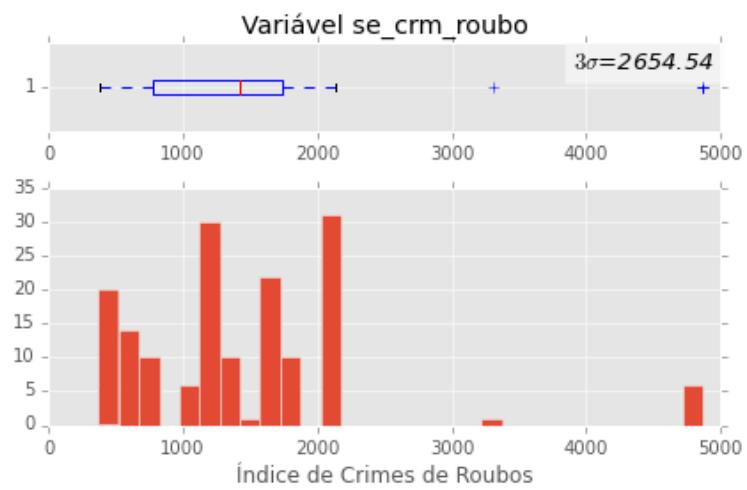


Figura H.8: Boxplot e histograma da variável *se_crm_roubo*

Crimes violentos

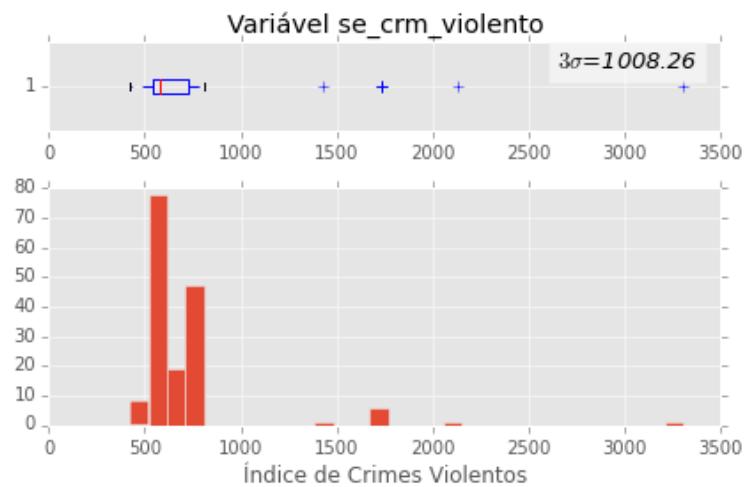


Figura H.9: Boxplot e histograma da variável *se_crm_violento*

Apêndice I

Listagem do IPython Notebook

Reg_Linear_-

_Modelo_Cidade.ipynb

Código para a criação de diferentes modelos de regressão linear para toda a cidade do Rio de Janeiro.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fgv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

Reg_Linear_-_Modelo_Cidade

June 15, 2015

1 Table of Contents

- 1. Inicialização
 - 1.1 Carregar dataset
 - 1.2 Tratar dataset.
- 2. Não usar variáveis de bairro: m1
- 3. Usar variáveis de bairro: m2
- 4. Remover bairros com poucas observações $m_{bairro} < n_{var}$: m3
 - 4.1 Identificar bairros com poucas observações
 - 4.2 Remover bairros com poucas observações
- 5. Remover variáveis correlacionadas
 - 5.1 Verificar correlação absoluta com preço
 - 5.2 Identificar variáveis correlacionadas
 - 5.3 Remover variáveis correlacionadas de menor correlação com preço: m4
 - 5.4 Remover variáveis correlacionadas por quantidade de ocorrência: m5
- 6. Lag espacial sem bairros: m6
- 7. Lag espacial com bairros: m7
- 8. Lag espacial sem variáveis correlacionadas
 - 8.1 Remover variáveis correlacionadas de menor correlação com preço: m8
 - 8.2 Remover variáveis correlacionadas por quantidade: m9
- 9. Seleção manual: m10
 - 9.1 Identificar variáveis correlacionadas
 - * 9.1.1 Acessibilidade
 - * 9.1.2 Distância
 - * 9.1.3 Variáveis dummies
 - 9.2 Executar modelo
- 10. Avaliar evolução do modelo escolhido, m10, adicionando uma variável por vez
- 11. Seleção do melhor modelo
 - 11.1 Comparar estatísticas
 - 11.2 Análise gráfica
- 12. Salvar modelo da regressão linear

2 1. Inicialização

```
In [4]: %pylab inline
import zap_util as z
import re
import statsmodels.api as sm
import statsmodels.formula.api as smf
from collections import OrderedDict
from sklearn import cross_validation as cv
import pysal

K_FOLDS = 4
CORR_THRESHOLD = 0.8
z.set_style()
# Lista de modelos
score = []
rcParams['figure.figsize'] = (9,6)
# Variáveis a serem tratadas como categóricas.
vars_catgr = ['suites','quartos','garagem']
```

Populating the interactive namespace from numpy and matplotlib

```
In [5]: def plot_score(scr = None, size= (6,4)):
    """
    Plota gráfico comparativo do RMSE dos modelos.
    """
    if scr == None:
        scr = score
    x_name = []
    y_test = []
    y_train = []
    for name, train, test,r2 in scr:
        x_name.append(name)
        y_train.append(train)
        y_test.append(test)
    figsize(size[0],size[1])
    scatter(range(len(x_name)), y_train,color='b', label='Treino');
    scatter(range(len(x_name)), y_test,color='r', label='Teste');
    xticks(range(len(x_name)), x_name, rotation=90);
    legend();
    xlim(-1,len(x_name)+1);

def save_result(name,results):
    """
    Helper para registro do RMSE.
    """
    for i in range(len(score)):
        if name ==score[i][0]:
            score[i] = [name,results[0],results[1],results[2]]
            return i
    score.append([name,results[0],results[1],results[2]])
```

```

def plot_rmser2(scr = None, size= (6,4)):
    """
    Plota um scatter plot com o RMSE no eixo X e R2 no eixo Y.
    """

    if scr == None:
        scr = score
    x_name = []
    x_train = []
    x_test = []
    x_r2 = []
    for name, train, test,r2 in scr:
        x_name.append(name)
        x_train.append(train)
        x_test.append(test)
        x_r2.append(1-r2)
    f,a=subplots()
    f.set_size_inches(size)
    a.scatter(x_test, x_r2,color='b');
    xlabel(r'$RMSE(Test)$',fontsize=14);
    ylabel(r'$1-R^2$',fontsize=14);

    for i in range(len(score)):
        label, x, y = x_name[i], x_test[i], x_r2[i]
        sinal = -1
        if i%2:
            sinal = 1

        plt.annotate(
            label,
            xy = (x, y), xytext = (20, 20*sinal),
            textcoords = 'offset points',
            ha = 'right',
            va = 'bottom',
            bbox = dict(boxstyle = 'round,pad=0.5',
                        fc = 'white',
                        alpha = 0.5),
            arrowprops = dict(arrowstyle = '->',
                             connectionstyle = 'arc3,rad=0',
                             color='black')
        )

def to_html(lm_result,model):
    DIR = './modelos'
    z.res_coef(lm_result).\
        to_html('{}_{}.html'.format(DIR,model))
    z.res_coef_bairros(lm_result).\
        to_html('{}_{}_{}.html'.format(DIR,model,'bairros'))
    z.res_coef_intr(lm_result).\

```

```
        to_html('{}_{}.html'.format(DIR,model,'vars'))
        lm_result.save('./modelos/{}.bin'.format(model))
```

2.1 1.1 Carregar dataset

```
In [6]: # Obter dataset do Banco de Dados.
df = z.get_imoveis_dataframe(False)

df.shape
```

```
Out[6]: (28111, 75)
```

2.2 1.2 Tratar dataset.

```
In [7]: # Verificação de nulos.
has_null = df.isnull().sum().sum() != 0
print 'Nulos no dataset: {}'.format(has_null)
```

```
Nulos no dataset: False
```

```
In [8]: # Verificação de constantes.
std_ = df.std()
has_const = std_[std_<0.001].sum()
print 'Variaveis constantes: {}'.format(has_const)
```

```
Variaveis constantes: 0
```

```
In [9]: print u'Qtd de variáveis: {}'.format(df.shape[1])
```

```
Qtd de variáveis: 75
```

3 2. Não usar variáveis de bairro: m1

```
In [10]: reload(z)
df_m1 = z.prep_statsmodels(df, False )
del df_m1['bairro_g']
df_m1.shape
```

```
Out[10]: (28111, 75)
```

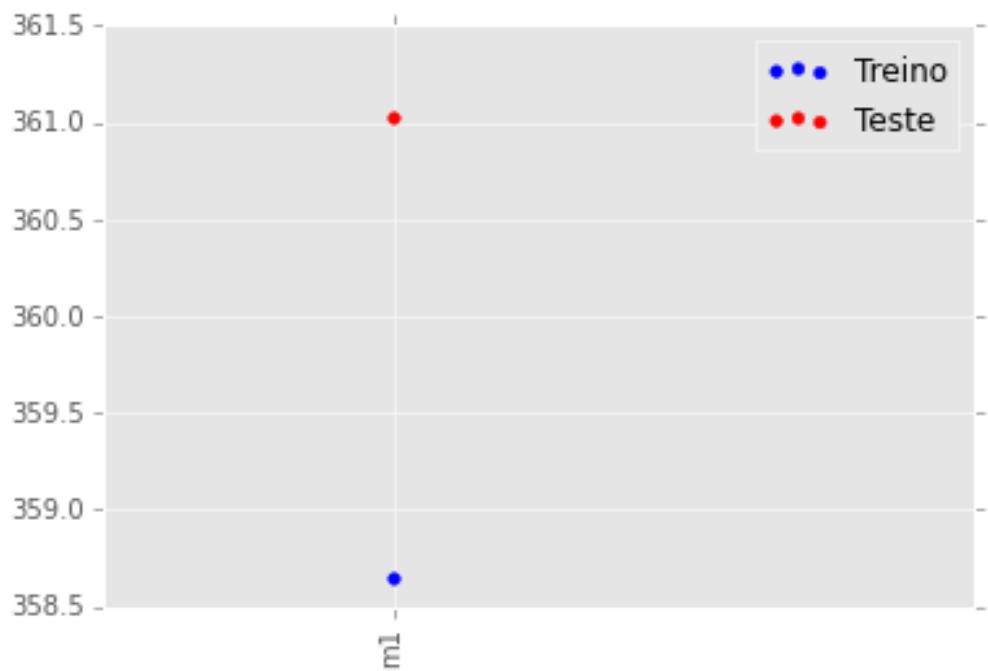
```
In [11]: %time
```

```
save_result('m1', z.run_model( df_m1,K_FOLDS))
```

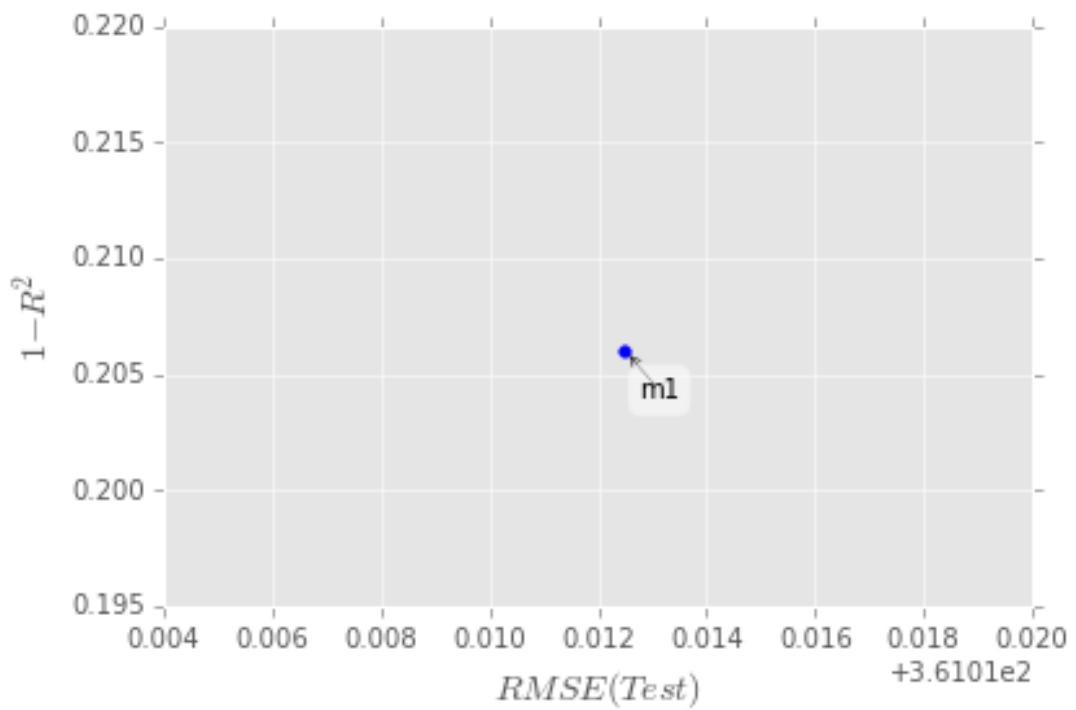
```
CPU times: user 3.36 s, sys: 430 ms, total: 3.79 s
Wall time: 3.12 s
```

```
In [12]: lm_,_,_ = z.ols(df_m1,avoid_plow=True, remove_plow_by_step=True)
to_html(lm,'m1')
del df_m1
```

```
In [13]: plot_score()
```



In [14]: `plot_rmser2()`



4 3. Usar variáveis de bairro: m2

```
In [15]: df_m2 = z.prep_statsmodels(df,True,True)
df_m2.shape

Out[15]: (28111, 229)

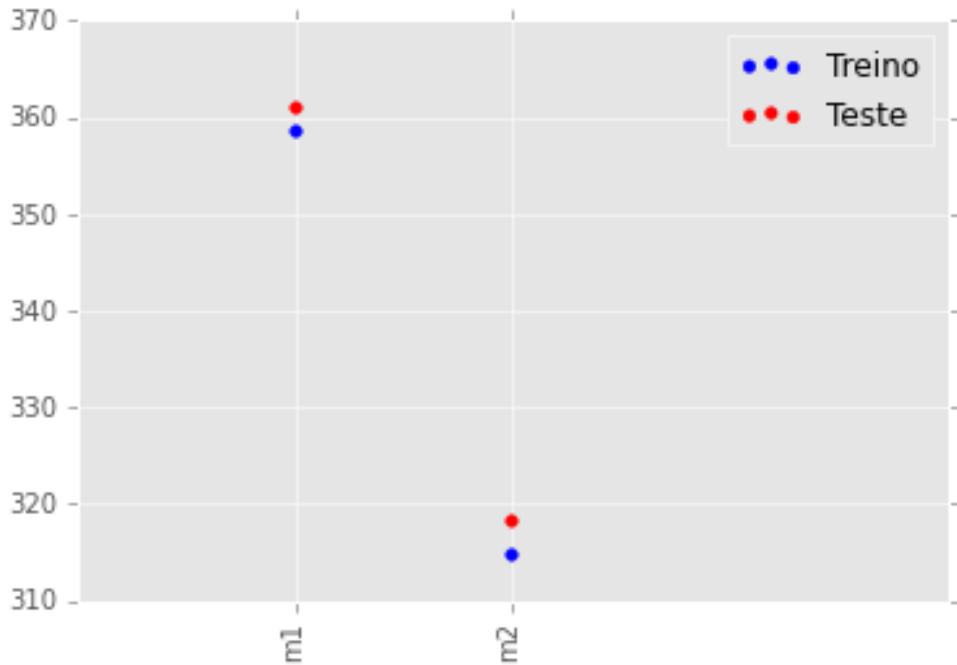
In [16]: %time

save_result('m2', z.run_model(df_m2,K_FOLDS))

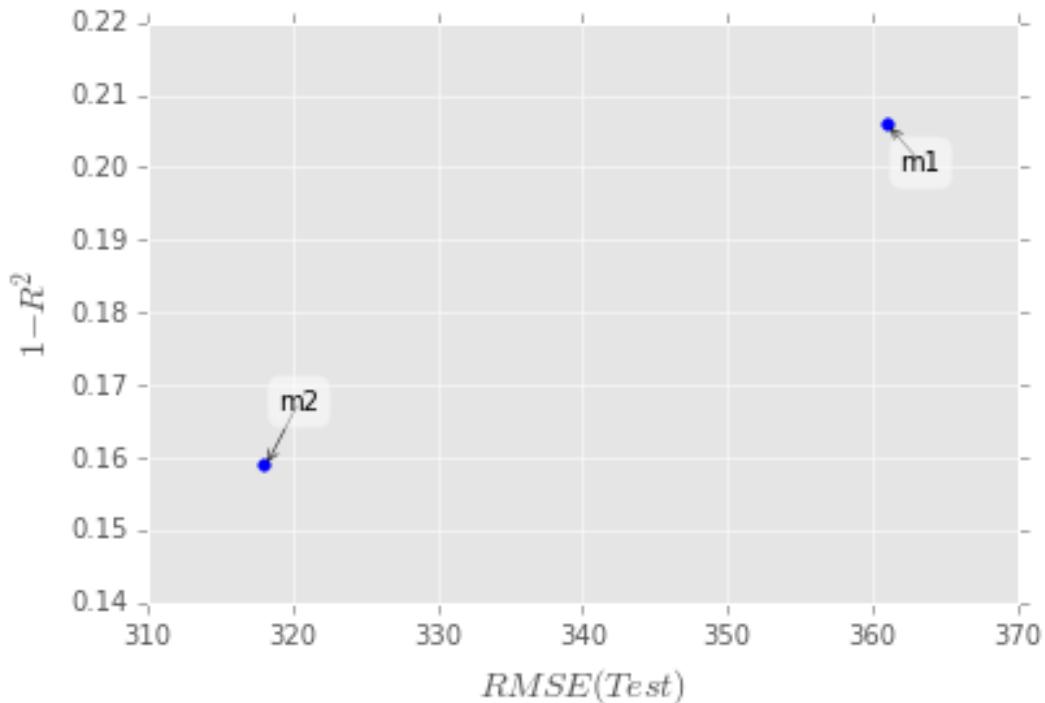
CPU times: user 28.9 s, sys: 2.88 s, total: 31.8 s
Wall time: 24 s
```

```
In [17]: lm_,-,_ = z.ols(df_m2,avoid_plow=True, remove_plow_by_step=True)
to_html(lm,'m2')
del df_m2
```

```
In [18]: plot_score()
```



```
In [19]: plot_rmser2()
```



5 4. Remover bairros com poucas observações $m_{bairro} < n_{var}$: m3

5.1 4.1 Identificar bairros com poucas observações

```
In [20]: bairro = df.bairro_g.copy()
        bairro = bairro.apply(lambda x : x.replace(' ', '_').\
            replace('(', '').replace(')', ''))
```

```
In [21]: print 'Quantidade total de variáveis: {}'.format( df.shape[1] )
```

Quantidade total de variáveis: 75

```
In [22]: print 'Qtd total de bairros: {}'.format(len(bairro.unique()))
```

Qtd total de bairros: 155

```
In [23]: print 'Qtd de observações: {}'.format(len(df))
```

Qtd de observações: 28111

```
In [24]: # Bairros com menos observações que qtd de variáveis.
        x = bairro.value_counts()
        bairro_less = x[x<df.shape[1]]
```

```
In [25]: print 'Qtd de bairros com poucas observações: {}'.format(len(bairro_less))
```

Qtd de bairros com poucas observações: 103

```
In [26]: # Bairros com mais observações que qtd de variáveis.
bairro_more = x[x>df.shape[1]]
```

```
In [27]: print 'Qtd de bairros com mais observações: {}'.format(len(bairro_more))
```

```
Qtd de bairros com mais observações: 51
```

5.2 4.2 Remover bairros com poucas observações

```
In [28]: df_m3 = z.prep_statsmodels(df)
```

```
In [29]: print 'Antes da remoção, m = {}, n = {}'.format(df_m3.shape[0], df_m3.shape[1])
```

```
Antes da remoção, m = 28111, n = 229
```

```
In [30]: df_m3 = df_m3[~df.bairro_g.isin(bairro_less.index.tolist())].copy()
for i in bairro_less.index.tolist():
    del df_m3[i]
```

```
In [31]: print 'Depois da remoção, m = {}, n = {}'.format(df_m3.shape[0], df_m3.shape[1])
```

```
Depois da remoção, m = 26704, n = 126
```

```
In [32]: # Verificar variáveis constantes.
const = df_m3.columns[df_m3.std() < 0.001]
print 'Qtd de variáveis constantes: {}'.format(len(const))
```

```
Qtd de variáveis constantes: 0
```

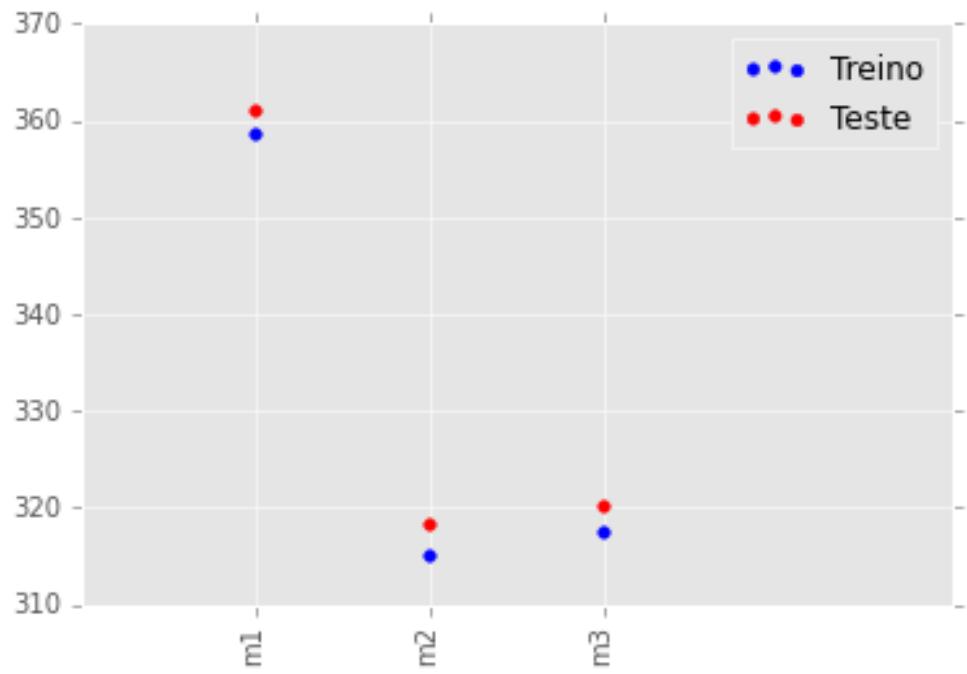
```
In [33]: %%time
```

```
save_result('m3', z.run_model(df_m3,K_FOLDS))
```

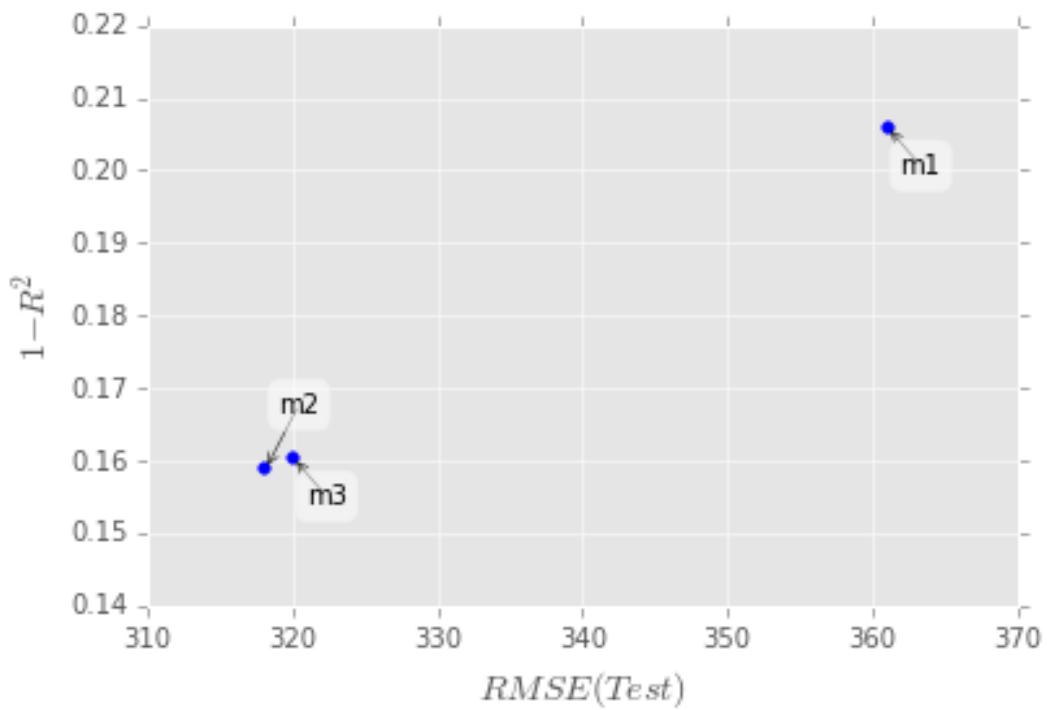
```
CPU times: user 7.68 s, sys: 789 ms, total: 8.46 s
Wall time: 6.63 s
```

```
In [34]: lm,_,_ = z.ols(df_m3,avoid_plow=True, remove_plow_by_step=True)
to_html(lm,'m3')
del df_m3
```

```
In [35]: plot_score()
```



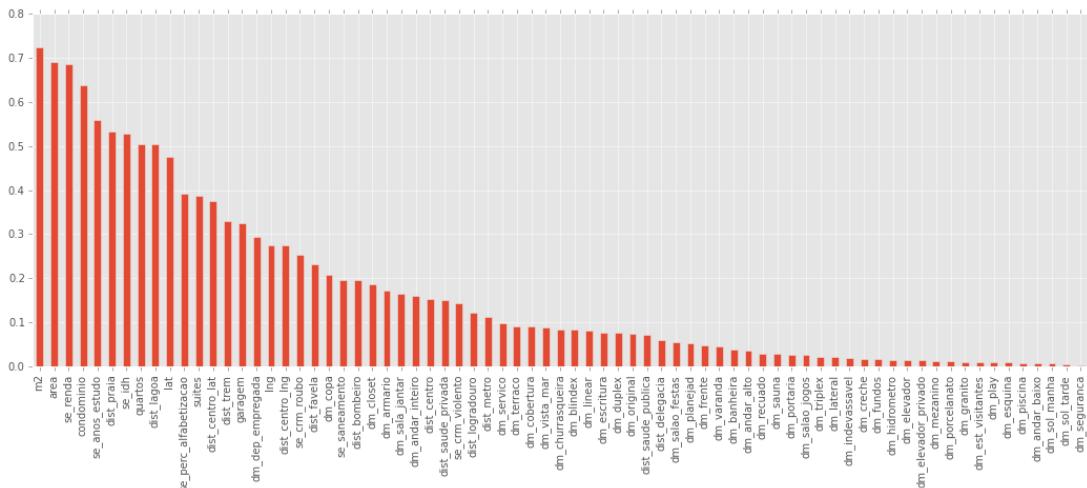
In [36]: `plot_rmser2()`



6 5. Remover variáveis correlacionadas

6.1 5.1 Verificar correlação absoluta com preço

```
In [37]: dcorr = []
    for var in df.columns:
        if var not in ['bairro_g', 'preco']:
            dcorr[var] = corrcoef(df['preco'], df[var])[0][1]
dcorrabs = z.pd.Series(dcorr).abs()
dcorrabs.sort(ascending=False);
dcorrabs.plot(kind='bar', figsize=(18,6));
```



6.2 5.2 Identificar variáveis correlacionadas

```
In [38]: # Identificação inicial das variáveis correlacionadas.
z.print_autocorr(df, [])
```

Coluna	Autocorrelacionada com
se_anos_estudo	: ['se_renda(0.86)', 'se_idh(0.94)', 'se_perc_alfabetizacao(0.85)']
dist_delegacia	: ['dist_metro(0.85)']
dist_centro_lng	: ['lng(-1.00)', 'dist_centro(0.97)', 'dist_metro(0.89)']
se_saneamento	: ['dist_centro_lng(-0.81)', 'lng(0.81)', 'dist_metro(-0.81)']
lng	: ['dist_centro(-0.97)', 'dist_metro(-0.89)']
dist_centro	: ['dist_metro(0.94)']
se_idh	: ['se_perc_alfabetizacao(0.83)']
dist_trem	: ['dist_centro_lat(0.90)']
dist_lagoinha	: ['lat(0.91)']

6.3 5.3 Remover variáveis correlacionadas de menor correlação com preço: m4

```
In [39]: # Ordenar variáveis por correlação com preço.
```

```
# Dicionário de variáveis correlacionadas.
vars_ac = z.vars_corr(df.corr(), CORR_THRESHOLD)
```

```

# Transformar os valores do dict em uma lista só.
vars_coef = reduce(lambda x,y : x+y, vars_ac.values())

# Extrair somente o nome.
vars_ = [i[:i.find('(')] for i in vars_coef]

# Unir com demais variáveis keys do dict.
vars_ = list(set(vars_).union(set(vars_ac.keys())))

# Recuperar correlação com preço.
vars_ac = dcorrabs[vars_]

# Ordernar por correlação com preço.
vars_ac.sort()

# Excluir variáveis temporárias.
del vars_, vars_coef

# Exibir lista final.
vars_ac

Out[39]: dist_delegacia      0.059389
          dist.metro        0.112188
          dist.centro       0.152582
          se_saneamento     0.196676
          dist_centro_lng   0.275108
          lng               0.276008
          dist.trem         0.330050
          dist.centro_lat   0.375193
          se_perc_alfabetizacao 0.392501
          lat               0.475684
          dist.lagoa         0.503342
          se_idh             0.527971
          se_anos_estudo    0.559080
          se_renda           0.685542
          dtype: float64

In [40]: # Refinamento da correlação.
          vars_auto_corr1 = ['dist_delegacia', 'dist.metro', 'dist.centro', 'se_saneamento',
                               'dist_centro_lng', 'dist.trem', 'se_perc_alfabetizacao', 'se_anos_estudo']

          z.print_autocorr(df, vars_auto_corr1, CORR_THRESHOLD)

          Coluna           | Autocorrelacionada com
          dist.lagoa       : ['lat(0.91)']

In [41]: print 'Quantidade de variáveis correlacionadas: {}'.format(len(vars_auto_corr1))

Quantidade de variáveis correlacionadas: 8

In [42]: print 'Quantidade total de variáveis: {}'.format(len(df.columns))

Quantidade total de variáveis: 75

In [43]: df_m4 = z.prep_statsmodels(df)

```

```
In [44]: print 'Antes da remoção, m = {}, n = {}'.format(df_m4.shape[0], df_m4.shape[1])
Antes da remoção, m = 28111, n = 229

In [45]: for i in vars_auto_corr1:
    try:
        del df_m4[i]
    except:
        print 'Erro ao tentar excluir {}.'.format(i)

Erro ao tentar excluir dist_centro_lng.
Erro ao tentar excluir se_anos_estudo.

In [46]: # Verificar variáveis constantes.
const = df_m4.columns[df_m4.std() == 0]
print 'Qtd de variáveis constantes: {}'.format(len(const))

Qtd de variáveis constantes: 0

In [47]: print 'Depois da remoção, m = {}, n = {}'.format(df_m4.shape[0], df_m4.shape[1])
Depois da remoção, m = 28111, n = 223

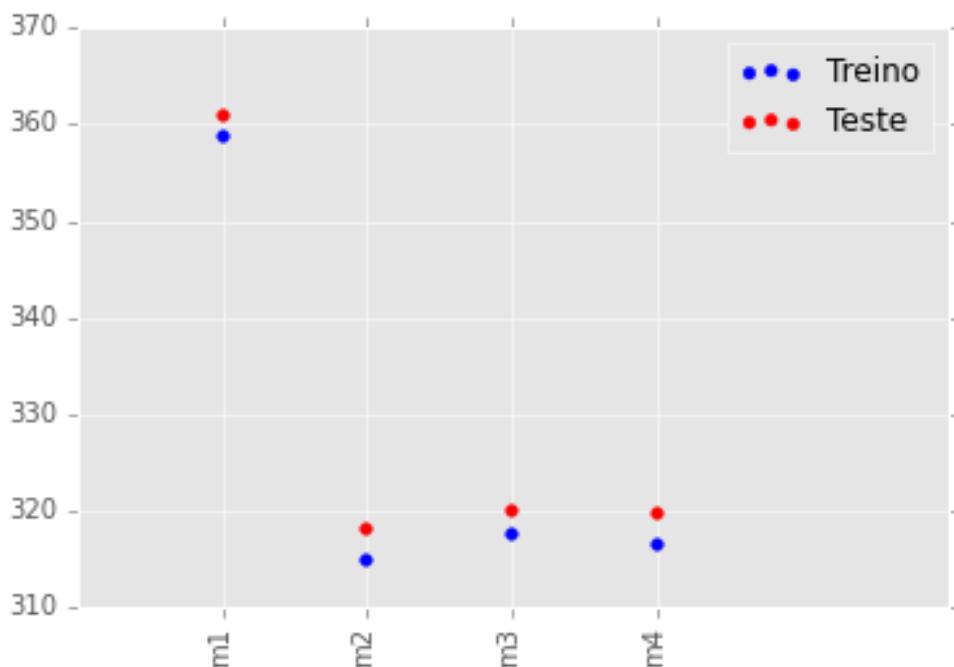
In [48]: %time

save_result('m4', z.run_model(df_m4,K_FOLDS))

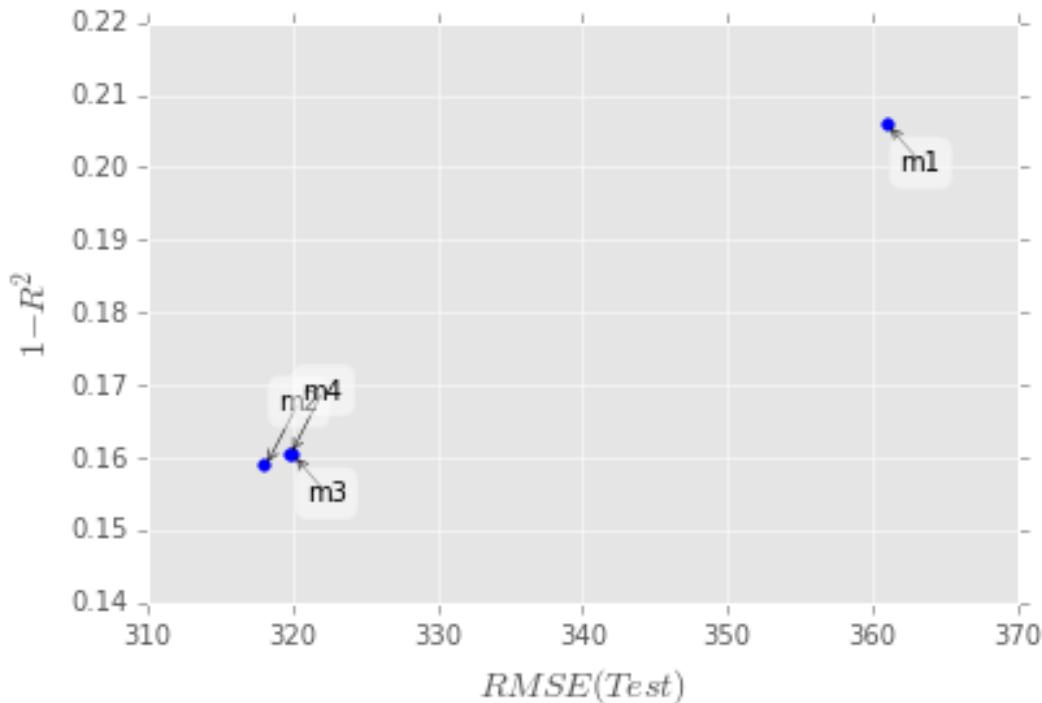
CPU times: user 26.3 s, sys: 2.66 s, total: 28.9 s
Wall time: 23.2 s

In [49]: lm_,_,_ = z.ols(df_m4,avoid_plow=True, remove_plow_by_step=True)
to_html(lm,'m4')
del df_m4

In [50]: plot_score()
```



```
In [51]: plot_rmser2()
```



6.4 5.4 Remover variáveis correlacionadas por quantidade de ocorrência: m5

```
In [52]: # Ordenar variáveis por ocorrência.  
vars_ac = z.vars_corr(df.corr(), CORR_THRESHOLD)  
vars_coef = reduce(lambda x,y : x+y, vars_ac.values())  
vars_ = [i[:i.find('(?')]] for i in vars_coef] + vars_ac.keys()  
vars_ = z.pd.Series(vars_)  
vars_.value_counts()
```

```
Out[52]: dist_metro      5  
dist_centro_lng        4  
dist_centro            2  
se_idh                 2  
se_saneamento          2  
se_perc_alfabetizacao 2  
dist_lagoa              1  
dist_trem              1  
lng                     1  
lat                     1  
se_renda               1  
dist_centro_lat         1  
dist_delegacia          1
```

```

se_anos_estudo           1
dtype: int64

In [53]: # Refinamento da correlação.
vars_auto_corr2 = ['dist_metro', 'dist_centro_lng',
                   'se_idh', 'se_perc_alfabetizacao', 'dist_trem', 'se_anos_estudo']
vars_auto_corr2.sort()
z.print_autocorr(df, vars_auto_corr2, CORR_THRESHOLD)

Coluna          | Autocorrelacionada com
se_saneamento   : ['lng(0.81)']
lng              : ['dist_centro(-0.97)']
dist_lagoa       : ['lat(0.91)']

In [54]: print 'Quantidade de variáveis correlacionadas: {}'.format(len(vars_auto_corr2))

Quantidade de variáveis correlacionadas: 6

In [55]: print 'Quantidade total de variáveis: {}'.format(len(df.columns))

Quantidade total de variáveis: 75

In [56]: df_m5 = z.prep_statsmodels(df)

In [57]: print 'Antes da remoção, m = {}, n = {}'.format(df_m5.shape[0], df_m5.shape[1])

Antes da remoção, m = 28111, n = 229

In [58]: for i in vars_auto_corr2:
          try:
              del df_m5[i]
          except:
              print 'Erro ao tentar excluir variável {}.'.format(i)

Erro ao tentar excluir variável dist_centro_lng.
Erro ao tentar excluir variável se_anos_estudo.

In [59]: # Verificar variáveis constantes.
const = df_m5.columns[df_m5.std() == 0]
print 'Qtd de variáveis constantes: {}'.format(len(const))

Qtd de variáveis constantes: 0

In [60]: print 'Depois da remoção, m = {}, n = {}'.format(df_m5.shape[0], df_m5.shape[1])

Depois da remoção, m = 28111, n = 225

In [61]: %%time

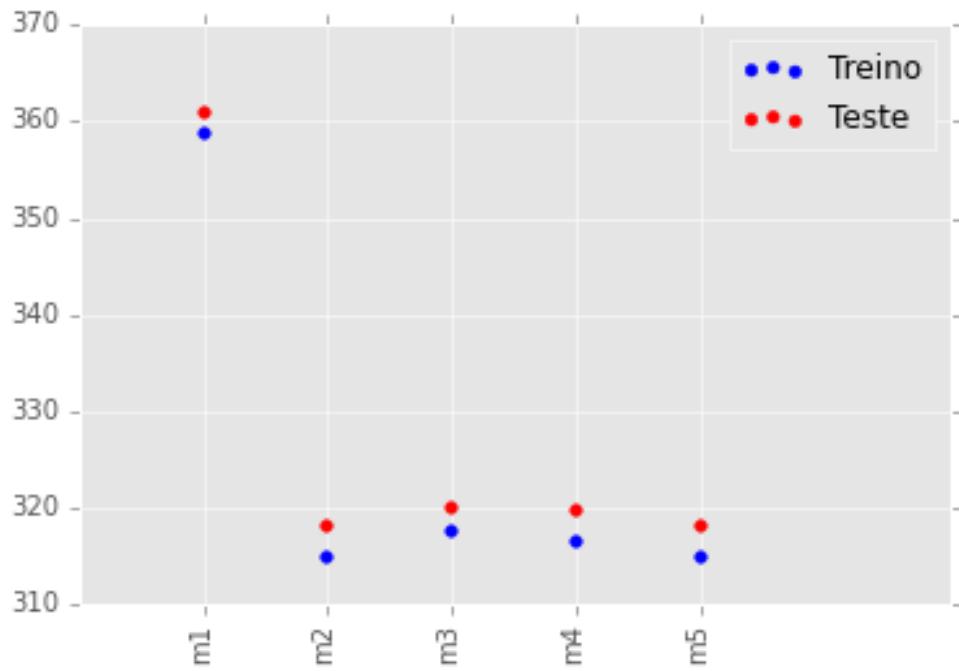
save_result('m5', z.run_model(df_m5, K_FOLDS))

CPU times: user 23.7 s, sys: 2.32 s, total: 26 s
Wall time: 19.6 s

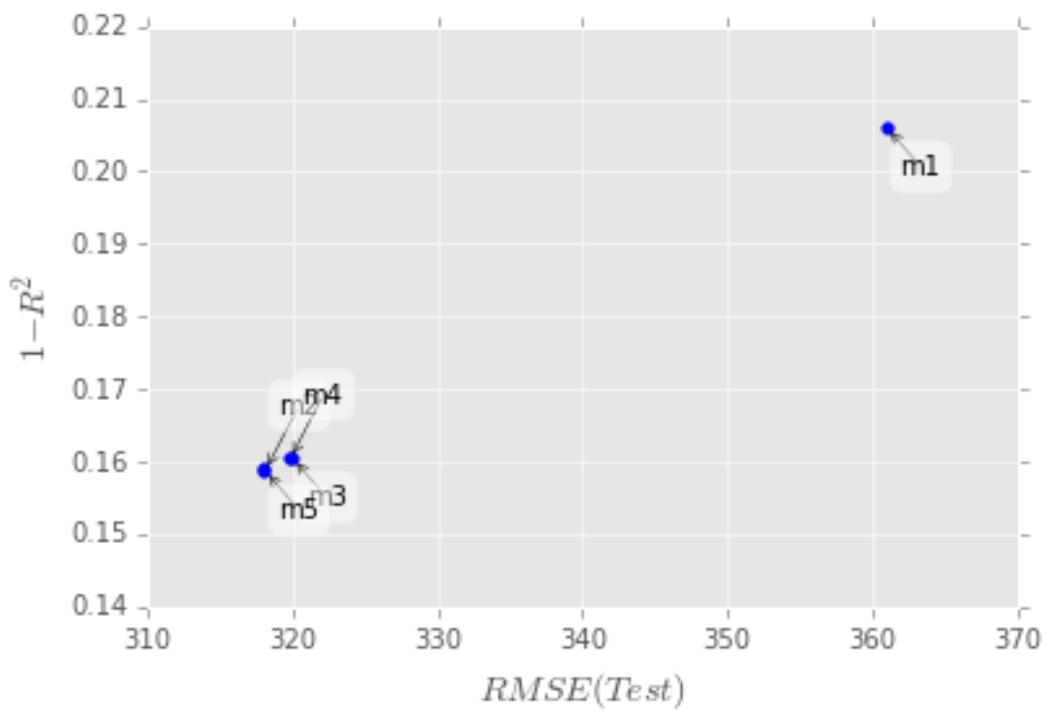
In [62]: lm_-, - = z.ols(df_m5, avoid_plow=True, remove_plow_by_step=True)
          to_html(lm, 'm5')
          del df_m5

```

```
In [63]: plot_score()
```



```
In [64]: plot_rmser2()
```



7 6. Lag espacial sem bairros: m6

```
In [65]: w = z.getW()

In [66]: df_m6 = z.prep_statsmodels(df, False)
         del df_m6['bairro_g']
         df_m6['preco_lag'] = pysal.lag_spatial(w, df_m6.precio)

In [67]: df_m6.shape

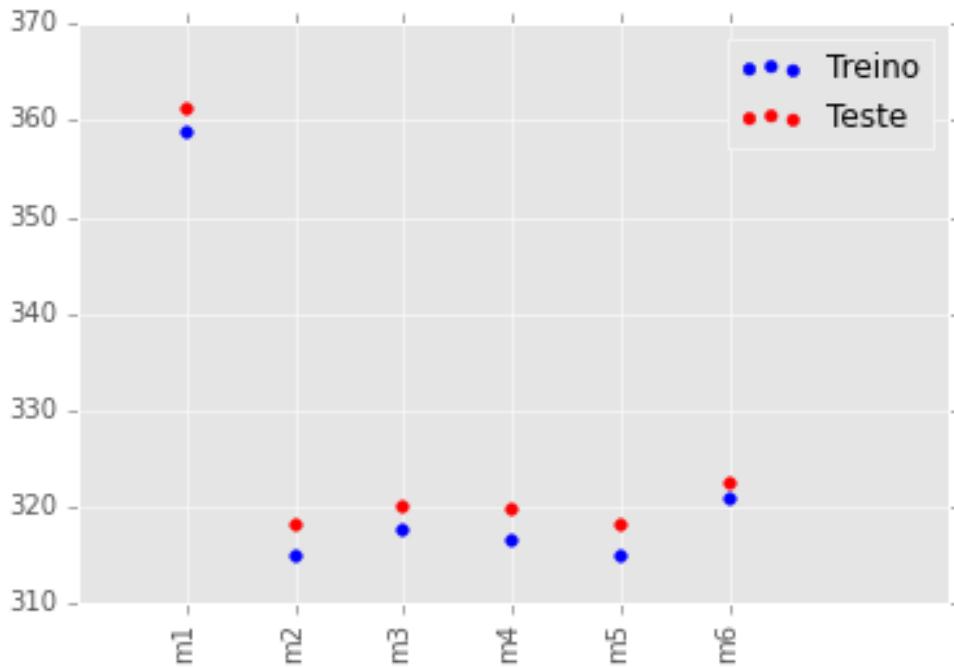
Out[67]: (28111, 76)

In [68]: %%time
         save_result('m6', z.run_model(df_m6,K_FOLDS))

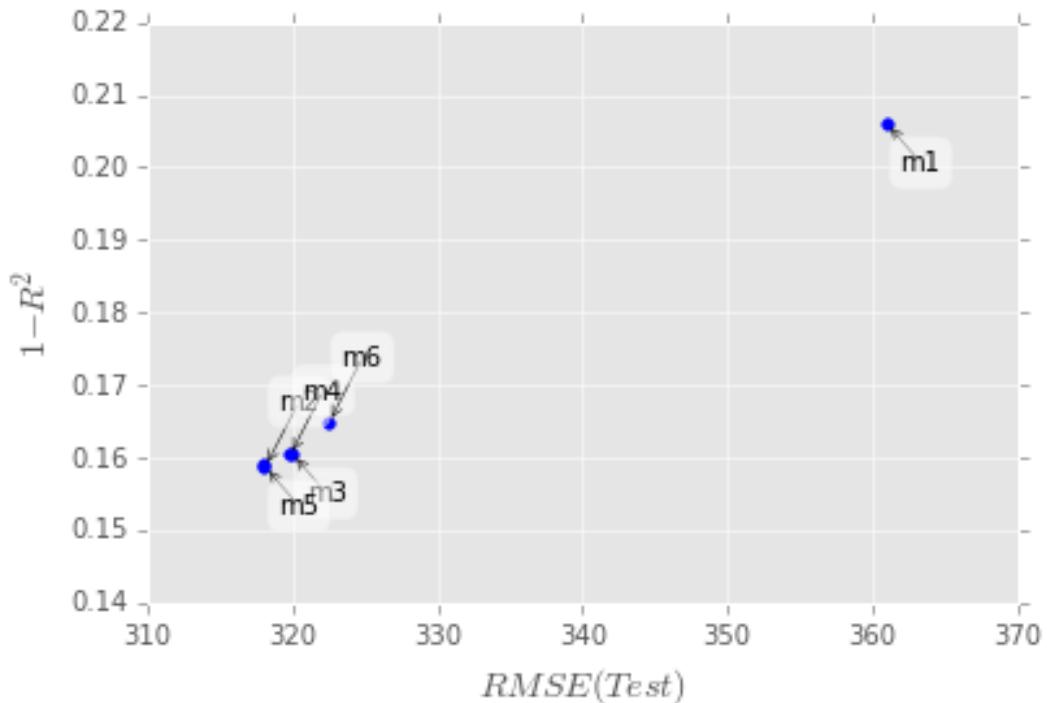
CPU times: user 4.07 s, sys: 595 ms, total: 4.67 s
Wall time: 3.89 s

In [69]: lm_,-,_ = z.ols(df_m6,avoid_plow=True, remove_plow_by_step=True)
         to_html(lm,'m6')
         del df_m6

In [70]: plot_score()
```



```
In [71]: plot_rmser2()
```



8 7. Lag espacial com bairros: m7

```
In [72]: df_m7 = z.prep_statsmodels(df)
df_m7.shape
```

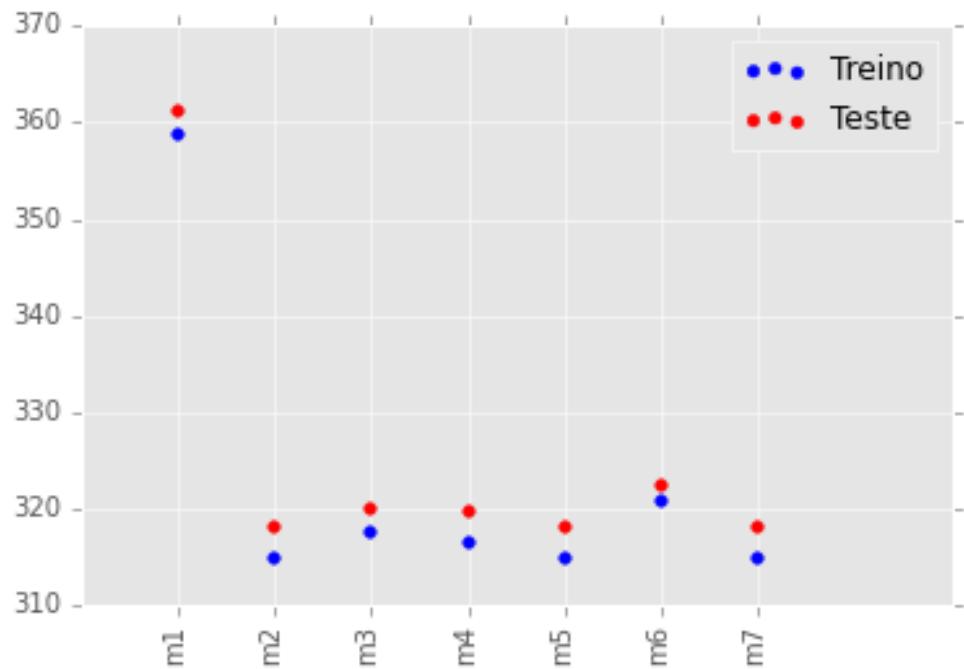
```
Out[72]: (28111, 229)
```

```
In [73]: %time
save_result('m7', z.run_model(df_m7, K_FOLDS))
```

CPU times: user 33.5 s, sys: 3.9 s, total: 37.4 s
Wall time: 31.2 s

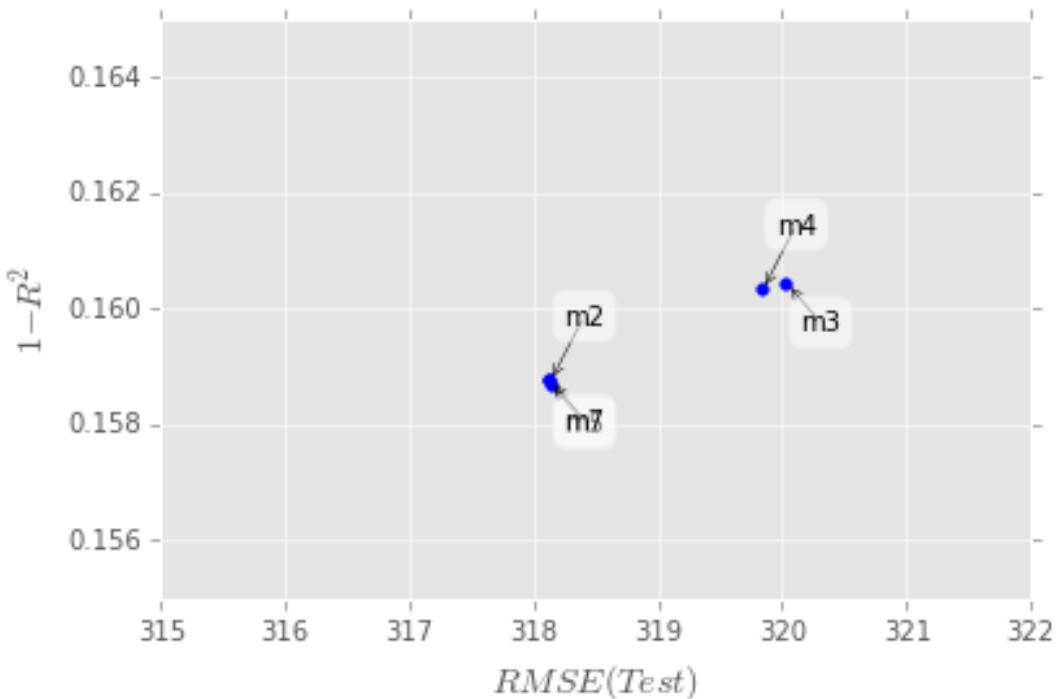
```
In [74]: lm_-, - = z.ols(df_m7, avoid_plow=True, remove_plow_by_step=True)
to_html(lm, 'm7')
del df_m7
```

```
In [75]: plot_score()
```



```
In [76]: plot_rmser2()  
       xlim(315,322)  
       ylim(0.155,0.165)
```

```
Out[76]: (0.155, 0.165)
```



9 8. Lag espacial sem variáveis correlacionadas

9.1 8.1 Remover variáveis correlacionadas de menor correlação com preço: m8

```
In [77]: df_m8 = z.prep_statsmodels(df)

In [78]: df_m8 = z.prep_statsmodels(df)
        for i in vars_auto_corr1:
            try:
                del df_m8[i]
            except:
                print 'Erro ao tentar excluir {}'.format(i)
df_m8.shape
```

Erro ao tentar excluir dist_centro_lng.
 Erro ao tentar excluir se_anos_estudo.

```
Out[78]: (28111, 223)
```

```
In [79]: # Verificar se tem constantes
std_ = df_m8.std()
std_[std_<0.001]
```

```
Out[79]: Series([], dtype: float64)
```

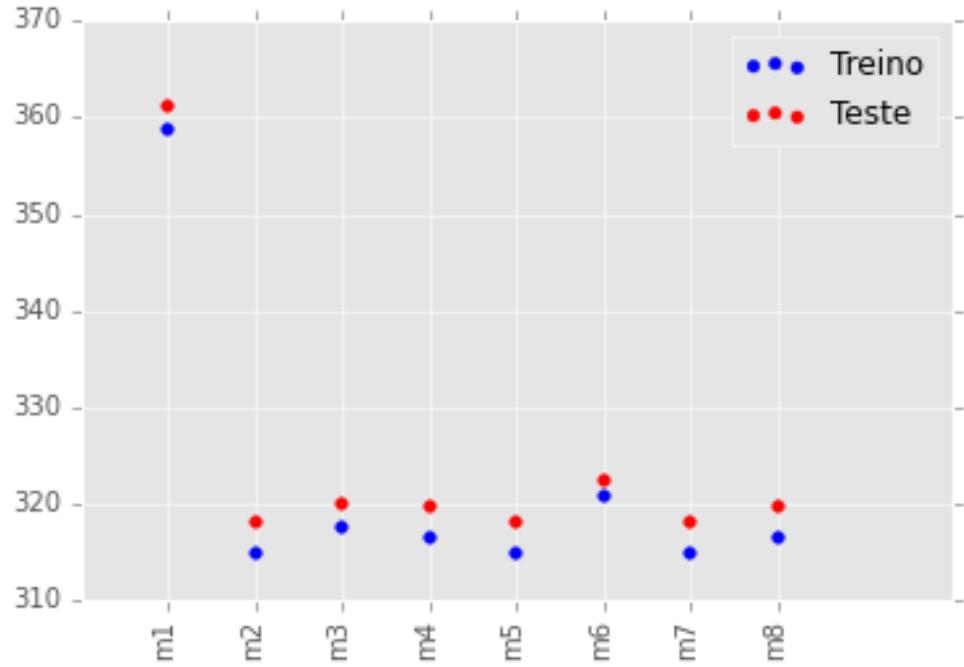
```
In [80]: %time
```

```
save_result('m8', z.run_model(df_m8,K_FOLDS))
```

```
CPU times: user 23.1 s, sys: 2.16 s, total: 25.3 s
Wall time: 18.8 s
```

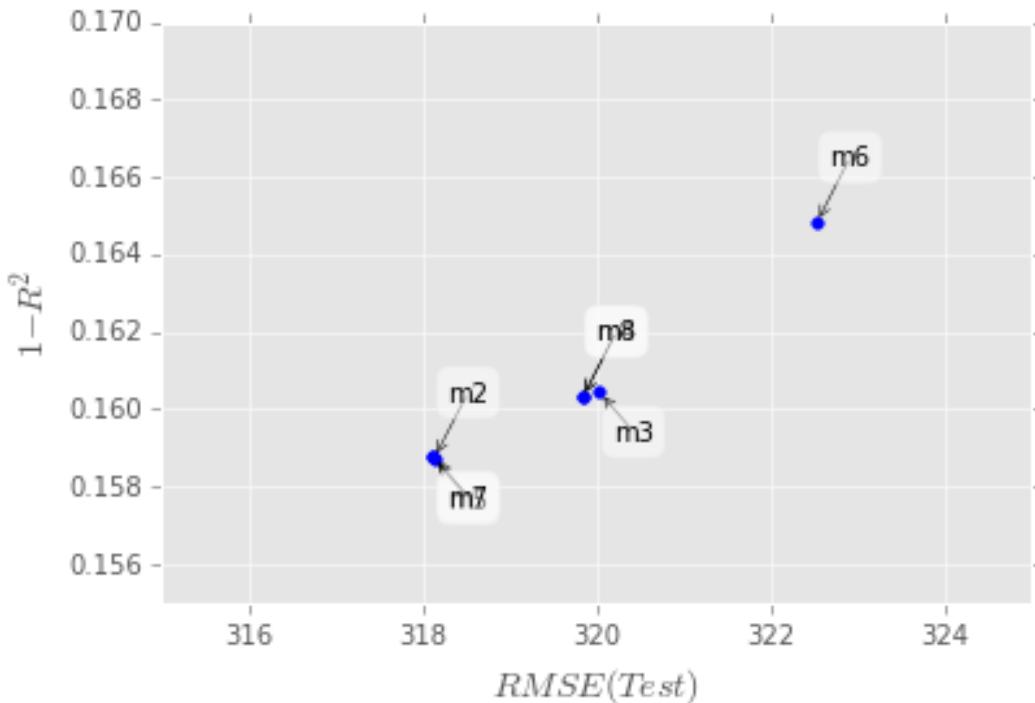
```
In [81]: lm_,-,_ = z.ols(df_m8,avoid_plow=True, remove_plow_by_step=True)
to_html(lm,'m8')
del df_m8
```

```
In [82]: plot_score()
```



```
In [83]: plot_rmser2()
ylim(0.155,0.17)
xlim(315,325)
```

```
Out[83]: (315, 325)
```



9.2 8.2 Remover variaveis correlacionadas por quantidade: m9

```
In [84]: df_m9 = z.prep_statsmodels(df)

In [85]: df_m9 = z.prep_statsmodels(df)
        for i in vars_auto_corri:
            try:
                del df_m9[i]
            except:
                print 'Erro ao tentar excluir {}'.format(i)
df_m9.shape
```

Erro ao tentar excluir dist_centro_lng.
 Erro ao tentar excluir se_anos_estudo.

```
Out[85]: (28111, 223)
```

```
In [86]: # Verificar se tem constantes
        std_ = df_m9.std()
        std_[std_<0.001]
```

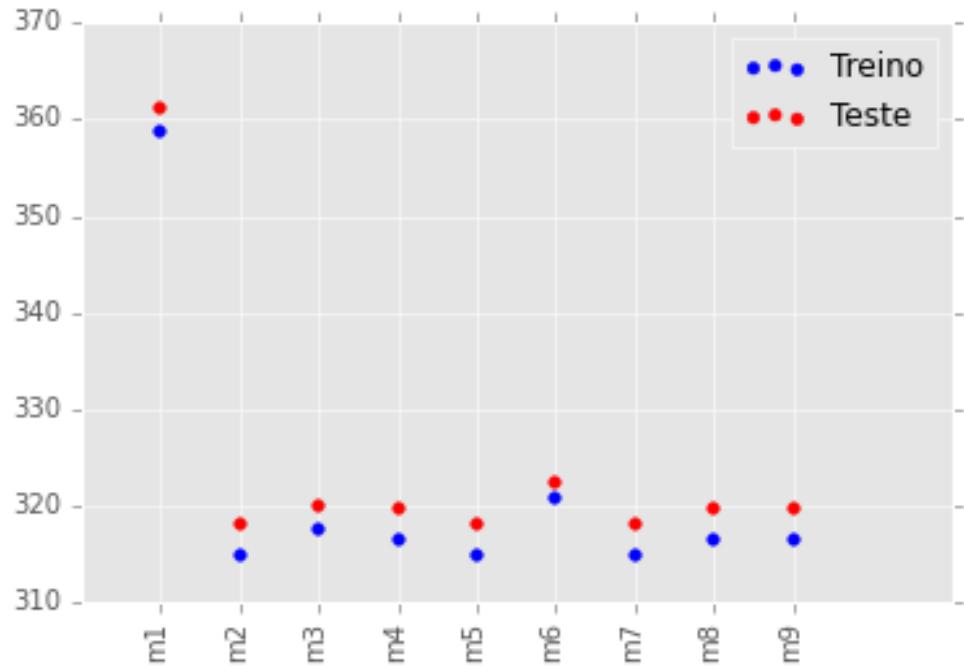
```
Out[86]: Series([], dtype: float64)
```

```
In [87]: %time
        save_result('m9', z.run_model(df_m9,K_FOLDS))
```

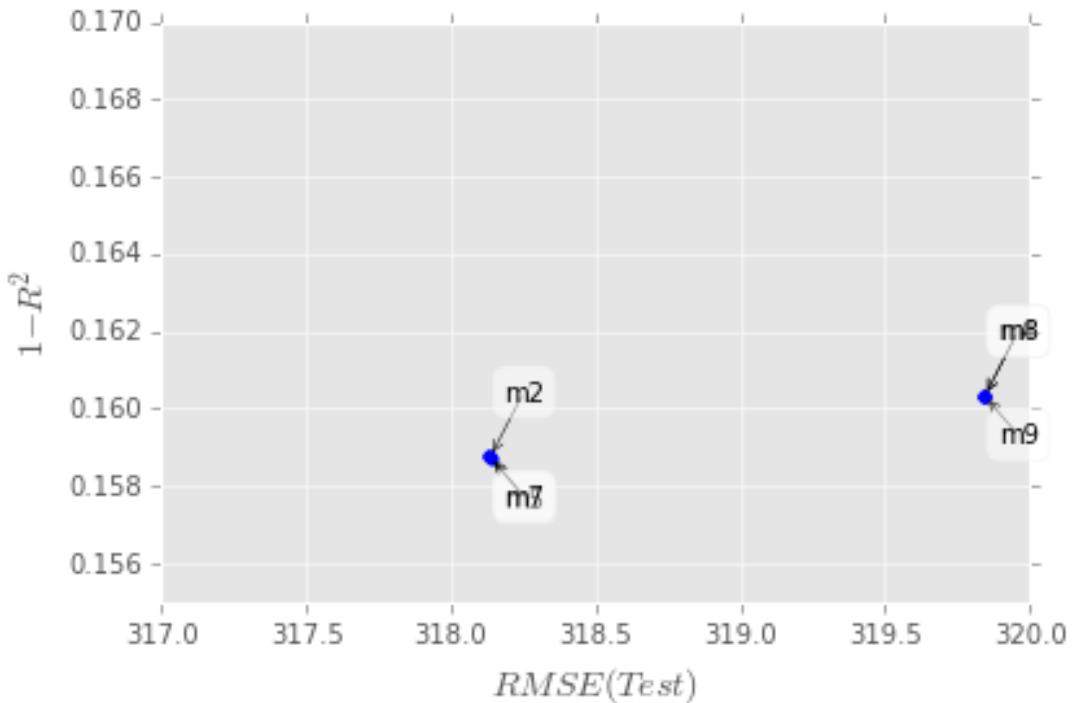
CPU times: user 26.2 s, sys: 2.88 s, total: 29.1 s
 Wall time: 22.9 s

```
In [88]: lm_,-,_ = z.ols(df_m9,avoid_plow=True, remove_plow_by_step=True)
to_html(lm,'m9')
del df_m9
```

```
In [89]: plot_score()
```



```
In [90]: plot_rmser2()
xlim(317,320);
ylim(0.155,0.17);
```



10 9. Seleção manual: m10

10.1 9.1 Identificar variáveis correlacionadas

10.1.1 9.1.1 Acessibilidade

```
In [91]: vars_se = [i for i in df.columns if i.find('se_') > -1]
df_se = df[vars_se]
corr_se = df_se.corr()
z.print_autocorr(corr_se)

Coluna          | Autocorrelacionada com
se_crm_violento : ['se_perc_alfabetizacao(-0.88)', 'se_renda(-0.88)', 'se_idh(-0.88)']
se_renda        : ['se_idh(0.95)', 'se_crm_roubo(-0.95)']
se_anos_estudo : ['se_crm_violento(-0.90)', 'se_perc_alfabetizacao(0.96)', 'se_renda(0.97)', 'se_idh(0.97)']
se_idh          : ['se_crm_roubo(-0.94)']
se_perc_alfabetizacao : ['se_renda(0.89)', 'se_idh(0.97)', 'se_crm_roubo(-0.86)']

In [92]: vars_ig_se = ['se_renda', 'se_perc_alfabetizacao', 'se_anos_estudo',
                  'se_crm_violento', 'se_crm_roubo', 'se_saneamento']
z.print_autocorr(corr_se, vars_ig_se)

Nao ha colunas correlacionadas entre si.

In [93]: del vars_se, df_se, corr_se
```

10.1.2 9.1.2 Distância

```
In [94]: vars_dist = [i for i in df.columns if i.find('dist_') > -1]
df_dist = df[vars_dist]
corr_dist = df_dist.corr()
z.print_acorr(corr_dist)

Coluna          | Autocorrelacionada com
dist_saude_publica : ['dist_centro_lat(0.80)', 'dist_trem(0.87)', 'dist_lagoa(-0.87)']
dist_bombeiro   : ['dist_centro_lng(0.81)', 'dist_centro(0.82)']
dist_delegacia  : ['dist_centro(0.91)', 'dist.metro(0.97)', 'dist_centro_lat(0.81)', 'dist_trem(0.89)']
dist_centro_lng : ['dist_centro(0.96)', 'dist.metro(0.87)']
dist_praia      : ['dist_centro_lat(-0.91)', 'dist_trem(-0.84)', 'dist_lagoa(0.84)']
dist_centro     : ['dist.metro(0.97)']
dist_metro      : ['dist_trem(0.84)']
dist_centro_lat : ['dist_trem(0.98)', 'dist_lagoa(-0.90)']
dist_trem       : ['dist_lagoa(-0.94)']

In [95]: vars_ig_dist = ['dist_centro_lat', 'dist_centro_lng', 'dist_saude_publica',
                     'dist_bombeiro', 'dist_delegacia', 'dist_trem', 'dist_lagoa', 'dist.metro']

z.print_acorr(corr_dist, vars_ig_dist)
```

Nao ha colunas correlacionadas entre si.

```
In [96]: del vars_dist, df_dist, corr_dist
```

10.1.3 9.1.3 Variáveis dummies

```
In [97]: vars_dm = [i for i in df.columns if i.find('dm_') > -1]

In [98]: df_dm = df[vars_dm]
z.print_acorr(df_dm.corr())

Coluna          | Autocorrelacionada com
dm_churrasqueira : ['dm_sauna(0.89)']
dm_salao_jogos   : ['dm_churrasqueira(0.81)', 'dm_est_visitantes(0.85)', 'dm_sauna(0.85)']
dm_banheira     : ['dm_creche(0.87)', 'dm_est_visitantes(0.83)']
dm_creche       : ['dm_est_visitantes(0.88)']
dm_duplex        : ['dm_cobertura(0.80)']
dm_piscina       : ['dm_salao_festas(0.85)', 'dm_salao_jogos(0.84)', 'dm_play(0.81)', 'dm_churrasqueira(0.81)']
dm_play          : ['dm_churrasqueira(0.81)', 'dm_sauna(0.82)']
dm_salao_festas : ['dm_play(0.95)', 'dm_churrasqueira(0.85)', 'dm_sauna(0.85)']

In [99]: vars_ig_dm = ['dm_churrasqueira', 'dm_duplex', 'dm_salao_jogos', 'dm_est_visitantes',
                  'dm_salao_festas', 'dm_banheira', 'dm_sauna', 'dm_play']

z.print_acorr(df_dm.corr(), vars_ig_dm)
```

Nao ha colunas correlacionadas entre si.

```
In [100]: del vars_dm, df_dm
```

10.2 9.2 Executar modelo

```
In [101]: df_m10 = z.prep_statsmodels(df, False)
del df_m10['bairro_g']
```

```
In [102]: vars_ig = vars_ig_se+vars_ig_dm+vars_ig_dist

In [134]: print ', '.join(['\\var{{{}}}'.format(i.replace('_', '\_')) for i in sort(vars_ig)])
\var{dist\_bombeiro}, \var{dist\_centro\_lat}, \var{dist\_centro\_lng}, \var{dist\_delegacia}, \var{dist\_}

In [103]: print ', '.join(['\\var{{{}}}'.format(i.replace('_', '\_')) for i in sort(vars_ig)])
\var{dist\_bombeiro}, \var{dist\_centro\_lat}, \var{dist\_centro\_lng}, \var{dist\_delegacia}, \var{dist\_}

In [104]: z.print_autocorr(df_m10, vars_ig)

Nao ha colunas correlacionadas entre si.

In [105]: for i in vars_ig:
    try:
        del df_m10[i]
    except Exception:
        print 'Variavel {} nao existe no modelo.'.format(i)
df_m10.shape

Variavel se_anos_estudo nao existe no modelo.
Variavel dist_centro_lat nao existe no modelo.
Variavel dist_centro_lng nao existe no modelo.

Out[105]: (28111, 56)
```

10.3 Tradicional: m10trad

```
In [106]: %time
    save_result('m10trad', z.run_model(df_m10, K_FOLDS))

CPU times: user 2.84 s, sys: 435 ms, total: 3.28 s
Wall time: 2.86 s

In [107]: lm,_,_ = z.ols(df_m10, avoid_plow=True, remove_plow_by_step=True)
          to_html(lm, 'm10trad')
```

10.3.1 Lag Espacial: m10lag

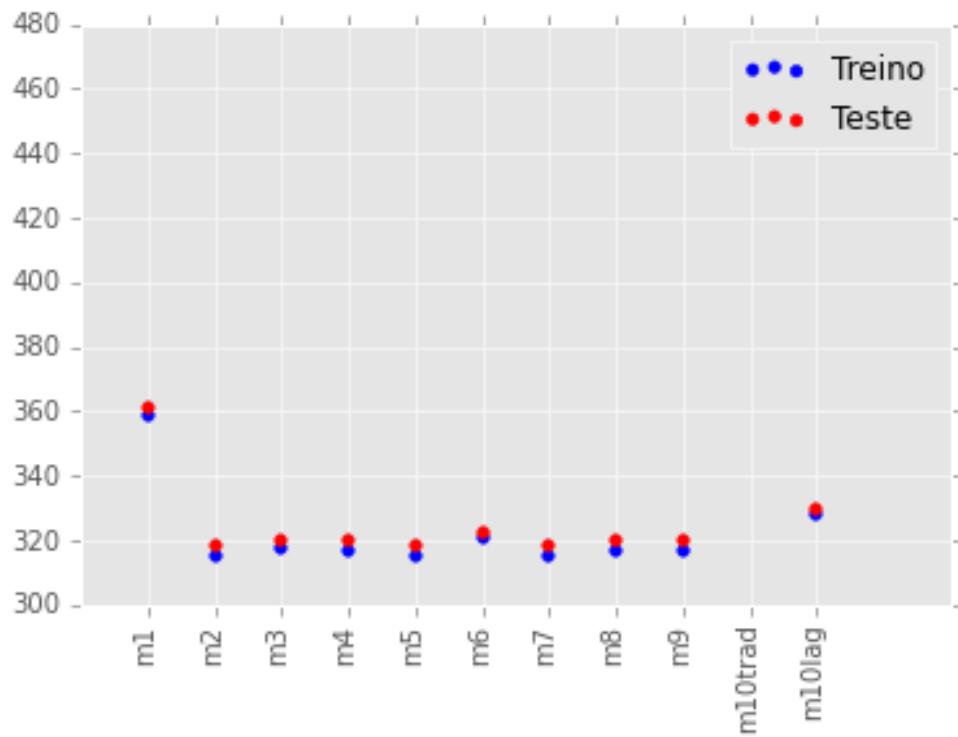
```
In [108]: w = z.getW()
        df_m10['preco_lag'] = pysal.lag_spatial(w,df_m10.preco)

In [109]: %time
        save_result('m10lag', z.run_model(df_m10, K_FOLDS))

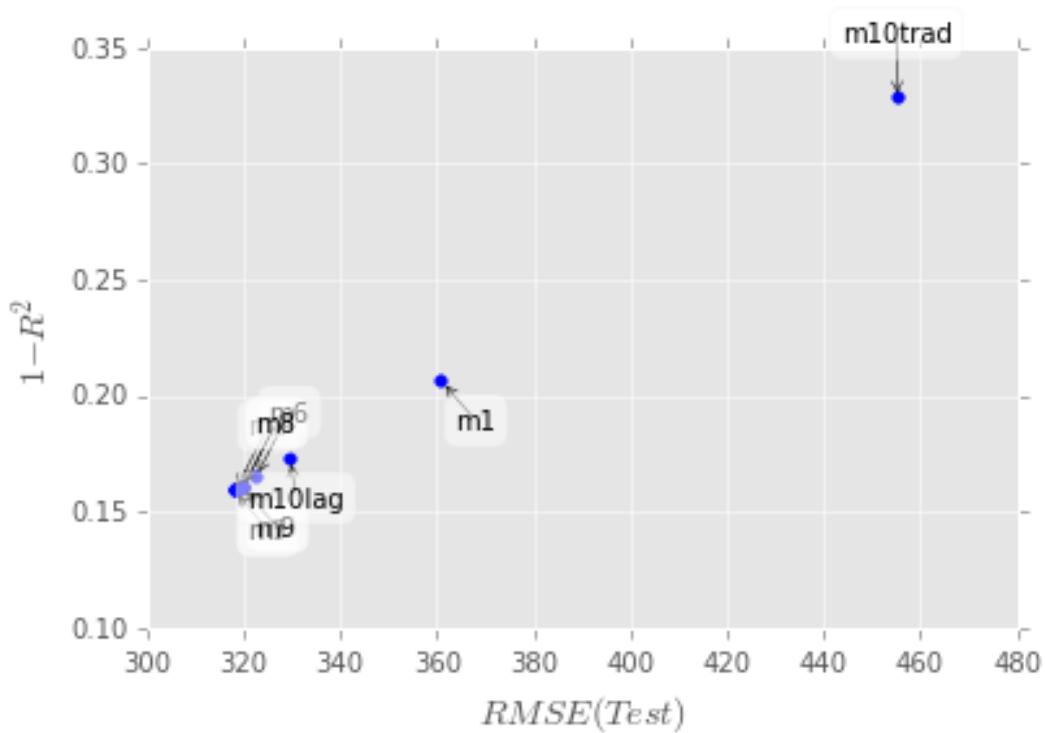
CPU times: user 2.01 s, sys: 300 ms, total: 2.31 s
Wall time: 1.98 s

In [110]: lm,_,_ = z.ols(df_m10,avoid_plow=True, remove_plow_by_step=True)
        to_html(lm,'m10lag')
        del df_m10

In [111]: plot_score()
```

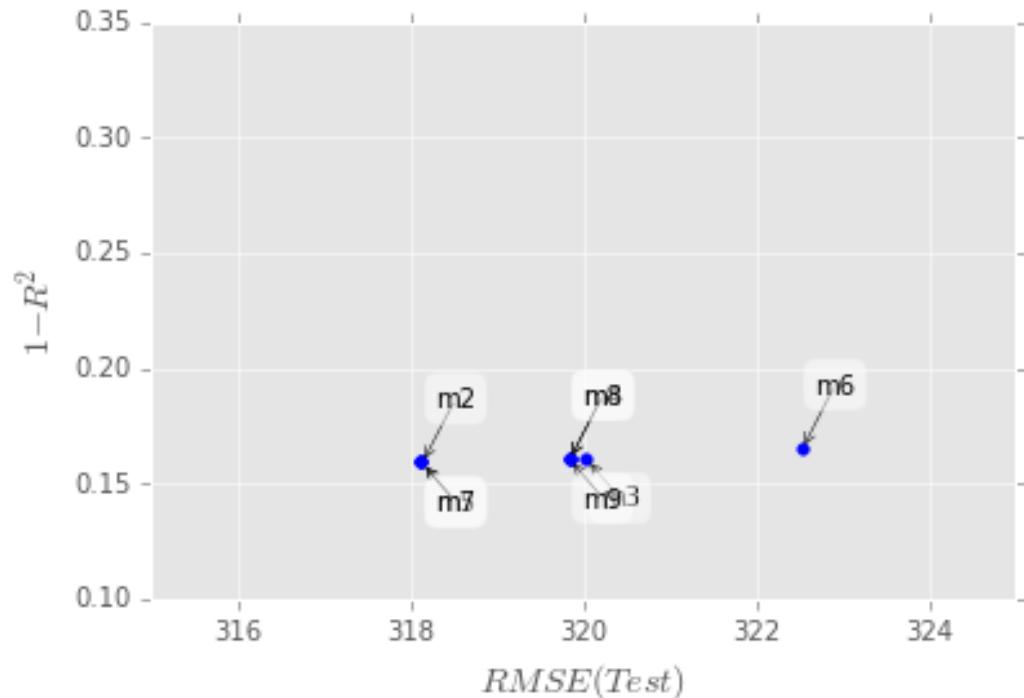


In [112]: `plot_rmser2()`



```
In [113]: plot_rmser2()
           xlim(315,325)
```

```
Out[113]: (315, 325)
```



11 10. Avaliar evolução do modelo escolhido, m10, adicionando uma variável por vez

```
In [114]: %%time
import re
import IPython

if False:

    score_vars = []
    cols = []
    for var_ in dcorrabs.index.tolist():
        if var_ in df_m10.columns.tolist():
            cols.append(var_)
            dfx = df_m10[['preco']+cols]
```

```

        print 'Modelo com {} variaveis. Ultima: {}.\'
              format(dfx.shape[1]-1,var_)
        lm_,_,_ = z.ols(dfx, [],[],False, True)
        score_vars.append(['n_{}'.format(i), z.rmse(lm_.resid),
                           lm_.rsquared, dfx.shape[1]-1])

IPython.display.clear_output()

score_vars = array(score_vars);
x_name = score_vars[:,0];
y_rmse = score_vars[:,1];
f,a = subplots();
f.set_size_inches(12,6);
a.scatter(range(len(cols)),y_rmse);
xticks(range(len(cols)),cols, rotation=90);
xlim(-1,len(cols));
ylabel('RMSE')

savefig('../texto/img/vars_evolucao_rmse.png');
z=pd.DataFrame(score_vars ,
               columns=['model','rmse_train','rmse_test']).\
               to_csv('../modelo_selecionado/score_variaveis.csv', index=False)

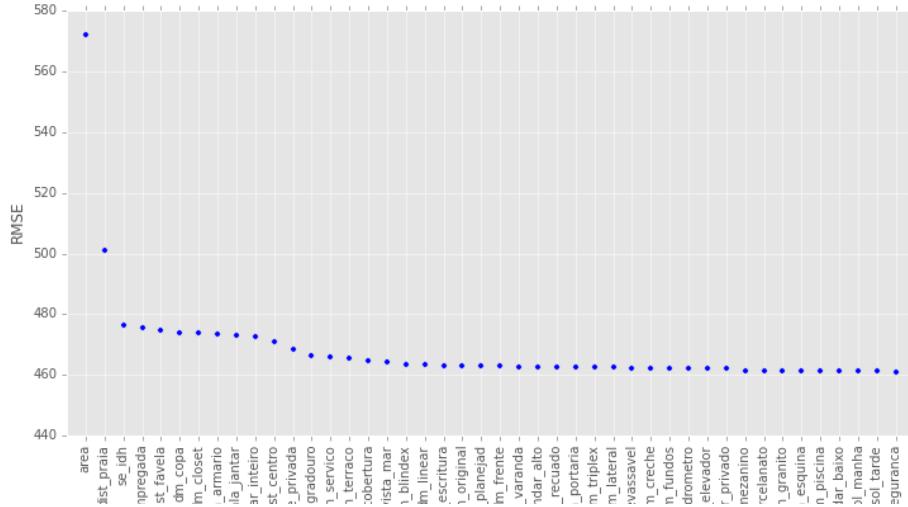
```

CPU times: user 9 μ s, sys: 1 μ s, total: 10 μ s

Wall time: 13.1 μ s

In [115]: IPython.display.Image(filename='../texto/img/vars_evolucao_rmse.png')

Out[115]:



12 Salvar resultados

In [116]: z=pd.DataFrame(score ,
 columns=['model','rmse_train','rmse_test','r2']).\

```
to_csv('~/modelo_selecionado/score_lm.csv', index=False)

In [117]: # Copia arquivos
!cp ./modelos/m10lag.bin ./modelo_selecionado/
!cp ./modelos/m10trad.bin ./modelo_selecionado/

In [ ]:
```

Apêndice J

Listagem do IPython Notebook

Reg_Linear_-

_Modelos_Bairro.ipynb

Código para construção e execução de modelos específicos a cada bairro.

Essa listagem está disponível no formato digital em http://nbviewer.ipython.org/github/srodriguez/fgv_dissertacao/blob/master/ipython_notebook/Lista%20de%20Arquivos.ipynb.

Reg_Linear_-_Modelos_Bairro

June 15, 2015

1 Table of Contents

- 1. Inicialização
- 2. Execução
- 3. Salvar resultados

2 1. Inicialização

```
In [1]: %pylab inline
import zap_util as z
import statsmodels.api as sm
from sklearn import cross_validation as cv
import IPython
z.set_style()

Populating the interactive namespace from numpy and matplotlib

In [2]: # Método para salvar os resultados em uma pasta específica.
import os

def save_results(folder, name, lm_result):
    for ext in ['txt', 'tex', 'bin']:
        dir_folder = '{}/{}'.format(folder, ext)
        if not os.path.isdir(dir_folder):
            os.makedirs(dir_folder)
        fname = '{}/{}.{}'.format(dir_folder, name, ext)
        with open(fname, 'w') as f:
            if ext == 'txt':
                f.write(lm_result.summary().as_text())
            elif ext == 'text':
                f.write(lm_result.summary().as_latex())
            elif ext == 'bin':
                lm_result.save(fname)

def vars_constants(dataframe, zero=0.001):
    x = dataframe.std()
    return x[x<zero].index.tolist()
```

3 2. Execução

```
In [3]: # Carrega datafram somente com imóveis já tratados.
df = z.get_imoveis_dataframe(False)
```

```

nome_bairros = df.bairro_g.unique()
nome_bairros.sort()
count_bairros = df.bairro_g.value_counts()
idx_bairro = df.bairro_g.copy()
df = z.prep_statsmodels(df, False)
del df['bairro_g']

df.shape

Out[3]: (28111, 75)

In [4]: preco_lag = False
if preco_lag:
    reload(z)
    import pysal
    w = z.getW()
    df['preco_lag'] = pysal.lag_spatial(w, df.preco)
df.shape

In [6]: %%time
# 3min

# Variáveis a serem tratadas como categóricas.
#vars_catgr = ['suites', 'quartos', 'garagem']
#vars_ignore = ['preco', 'condominio', 'm2', 'bairro_g'] + vars_catgr
#n_vars_ignore = len(vars_ignore)

# Remover variáveis constantes globais.

# Execução das regressões.
score = []
k=1
n = len(nome_bairros)
bairros_low = []
bairros_erro = []
for b in nome_bairros:
    print('Processando {} de {}: {}'.format(k,n,b))
    k+=1
    dfb = df.loc[idx_bairro == b].copy()

    '''
    # Tratar variáveis categóricas.
    for var_ in vars_catgr:
        dfcat = z.pd.get_dummies(dfb[var_])
        del dfcat[dfcat.columns.tolist()[0]]
        dfcat.columns = [var_+'_'+str(int(i)) for i
                        in dfcat.columns.tolist()]
        dfb = z.pd.concat([dfb, dfcat], axis=1)
    '''

    # Remover variáveis constantes.
    const = vars_constants(dfb)
    for i in const:
        del dfb[i]

```

```

# Verifica se dataset de bairro tem observações suficientes.
if len(dfb)> len(dfb.columns)-1:
    try:
        lm,_,_= z.ols(dfb, [], [],
                        avoid_corr=False, avoid_plow=True,
                        remove_plow_by_step=True)

        # Avaliação do modelo. Salva os seguintes resultados:
        # bairro, rmse, r2,tamanho, variáveis,mean(preco)
        # median(preco),mean(preco_lag),median(preco_lag)
        score.append([
            b, z.rmse(lm.resid),lm.rsquared,
            len(dfb),len(lm.params)-1,
            mean(dfb.preco),median(dfb.preco)
            #,mean(dfb.preco_lag),median(dfb.preco_lag)
            ])
        save_results('./modelos_bairro', b, lm)
    except Exception as E:
        print 'ERRO ao processar: {} \n {}'.format(b,E)
        bairros_erro.append(b)
    else:
        bairros_low.append(b)
    IPython.display.clear_output()

print 'FIM'

FIM
CPU times: user 1min 53s, sys: 1.02 s, total: 1min 54s
Wall time: 1min 54s

In [8]: count_bairros_low = count_bairros[count_bairros.index.isin(bairros_low)]
count_bairros_low = count_bairros_low.sort_index();

In [9]: print 'Total: {}'.format(count_bairros_low.shape[0])
        print 'Com modelo: {}'.format(len(score))
        print 'Sem modelo: {}'.format(count_bairros_low.shape[0])

Total: 155. Com modelo: 73. Sem modelo: 82

In [10]: str_bairros_low = ', '.join(['{}({})'.format(k[0],k[1]) for k in
                                     [i for i in count_bairros_low.iteritems()]])
        with open('../texto/tex/bairros_baixo_m.tex','w') as f:
            f.write(str_bairros_low)
        print str_bairros_low

Abolicao(42), Acari(1), Agua Santa(39), Alto da Boa Vista(36), Anchieta(3), Bancarios(18), Bangu(18), B

In [11]: print '{} bairros com erro ao processar: {}'.format(len(bairros_erro))\
        + ', '.join(bairros_erro)

0 bairros com erro ao processar:

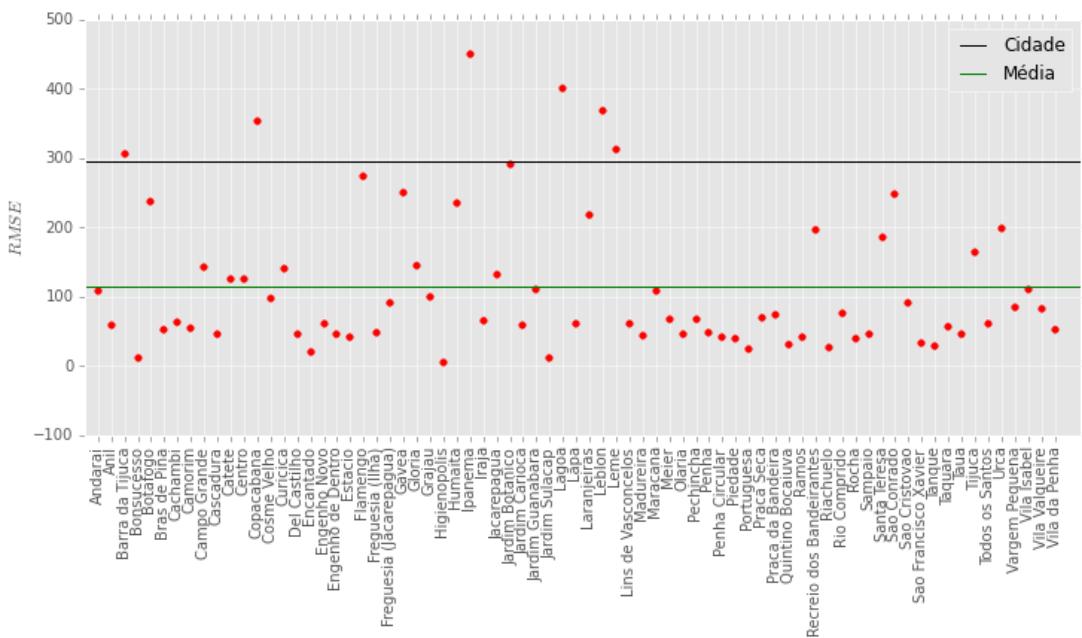
In [12]: def plot_score(lm_score):

    x_name = []
    y_test = []
    y_train = []

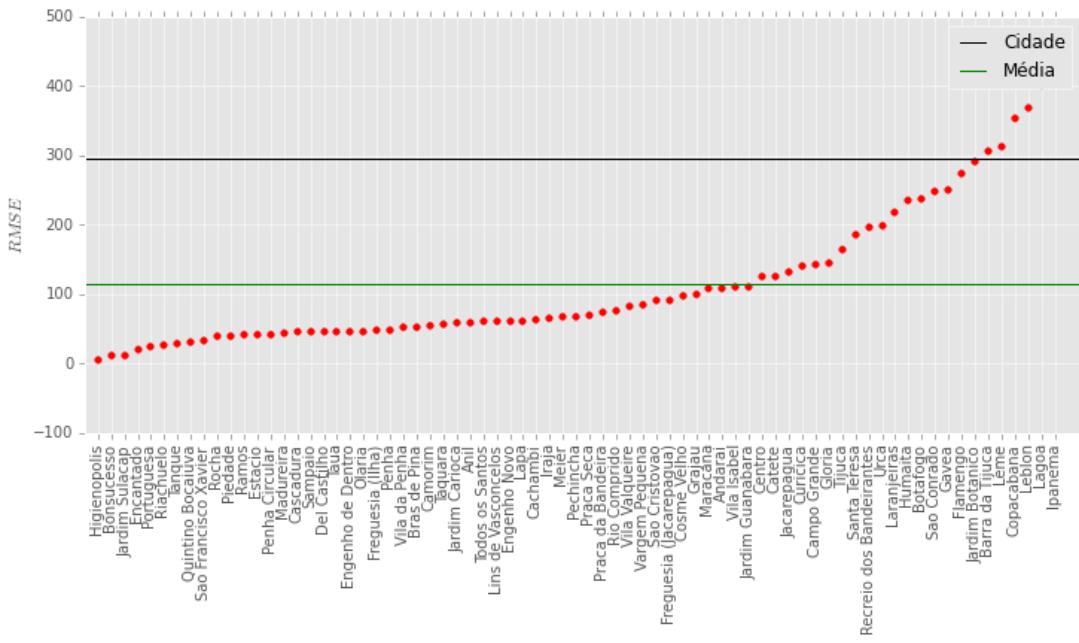
```

```
for i in range(len(lm_score)):
    x_name.append(lm_score[i][0])
    y_train.append(lm_score[i][1])
    y_test.append(NaN)
fig,ax = subplots()
ax.scatter(range(len(x_name)), y_train,color='r');
ax.set_ylabel(r'$RMSE$')
xticks(range(len(x_name)), x_name, rotation=90);
axhline(y=295.1687623761913, color='black', label='Cidade');
axhline(y=mean(y_train), color='green', label=u'Média');
legend();
xlim(-1,len(x_name)+1);

size(12,5)
t_score(score)
```



```
In [13]: plot_score(sorted(score, key=lambda x: x[1]))
```



4 3. Salvar resultados

```
In [18]: # bairro, rmse, r2, tamanho, variáveis, mean(preco)
# median(preco), mean(preco_lag), median(preco_lag)

dfx = z.pd.DataFrame(score,
    columns=['Bairro', 'RMSE', 'R2', 'm', 'n',
              'MediaPreco', 'MedianaPreco',
              #'MediaLag', 'MedianaLag']
    ])
dfx.to_csv('../ipython_notebook/modelos_bairro/score_bairros.csv',
           index=False)
```


Apêndice K

Resultados do Modelo Hedônico para os 5 bairros em maior quantidade de observações

K.0.1 Copacabana

OLS Regression Results						
Dep. Variable:	preco	R-squared:	0.804			
Model:	OLS	Adj. R-squared:	0.802			
Method:	Least Squares	F-statistic:	428.6			
Date:	Thu, 25 Jun 2015	Prob (F-statistic):	0.00			
Time:	18:31:42	Log-Likelihood:	-21485.			
No. Observations:	2962	AIC:	4.303e+04			
Df Residuals:	2933	BIC:	4.320e+04			
Df Model:	28					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	-1304.3801	140.361	-9.293	0.000	-1579.595	-1029.165
dm_closet	130.3088	28.926	4.505	0.000	73.592	187.026
dist_saude_privada	1135.1263	114.086	9.950	0.000	911.430	1358.823
dm_fundos	-86.1673	38.316	-2.249	0.025	-161.295	-11.039
dm_andar_alto	-53.9791	27.018	-1.998	0.046	-106.956	-1.002
dist_trem	1729.2115	190.103	9.096	0.000	1356.463	2101.960
dist_lagoa	-236.6424	40.687	-5.816	0.000	-316.420	-156.865
dm_sala_jogos	-169.4532	83.073	-2.040	0.041	-332.341	-6.565
area	7.6144	0.136	56.049	0.000	7.348	7.881
dm_elevador	-37.7961	16.949	-2.230	0.026	-71.029	-4.563
dm_linear	-275.9856	88.001	-3.136	0.002	-448.536	-103.435
dist_logradouro	-434.4262	128.415	-3.383	0.001	-686.219	-182.633
dist_centro	-1648.8597	183.005	-9.010	0.000	-2007.692	-1290.028
preco_lag	0.5869	0.034	17.366	0.000	0.521	0.653
dm_banheira	113.5340	44.318	2.562	0.010	26.637	200.431
dm_andar_inteiro	-72.1679	28.445	-2.537	0.011	-127.942	-16.394
dm_sala_jantar	51.2158	19.243	2.662	0.008	13.485	88.947
suites_2	168.4928	28.143	5.987	0.000	113.311	223.675
suites_3	680.3362	58.151	11.699	0.000	566.315	794.358
suites_1	90.3524	14.872	6.075	0.000	61.191	119.514
garagem_4	946.9349	142.628	6.639	0.000	667.273	1226.597
garagem_1	194.1135	15.548	12.485	0.000	163.627	224.600
garagem_2	380.5427	31.061	12.251	0.000	319.639	441.447
garagem_3	282.2519	67.942	4.154	0.000	149.033	415.471
dm_vista_mar	108.8768	28.790	3.782	0.000	52.426	165.327
dm_sauna	98.5951	45.533	2.165	0.030	9.316	187.874
quartos_3	78.7241	15.608	5.044	0.000	48.121	109.328
quartos_2	70.6083	17.887	3.947	0.000	35.536	105.681
dist_metro	127.4569	40.864	3.119	0.002	47.333	207.581
Omnibus:	748.789				2.037	
Prob(Omnibus):	0.000				2967.415	
Skew:	1.191				0.00	
Kurtosis:	7.286				6.56e+04	

K.0.2 Barra da Tijuca

OLS Regression Results						
Dep. Variable:	preco	R-squared:	0.818			
Model:	OLS	Adj. R-squared:	0.816			
Method:	Least Squares	F-statistic:	349.0			
Date:	Thu, 25 Jun 2015	Prob (F-statistic):	0.00			
Time:	18:31:24	Log-Likelihood:	-12885.			
No. Observations:	1806	AIC:	2.582e+04			
Df Residuals:	1782	BIC:	2.595e+04			
Df Model:	23					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	-2.228e-05	5.91e-06	-3.772	0.000	-3.39e-05	-1.07e-05
dist_saude_privada	84.1017	33.403	2.518	0.012	18.589	149.614
dist_lagoa	167.8100	25.807	6.503	0.000	117.196	218.425
preco_lag	0.1982	0.028	6.996	0.000	0.143	0.254
area	7.6641	0.198	38.705	0.000	7.276	8.052
dm_elevador	-69.1757	21.661	-3.194	0.001	-111.660	-26.692
dm_frente	121.7274	29.855	4.077	0.000	63.172	180.283
dist_delegacia	-61.0618	14.561	-4.194	0.000	-89.620	-32.504
dist_favela	36.0574	8.794	4.100	0.000	18.810	53.305
dist_logradouro	183.8896	28.097	6.545	0.000	128.784	238.996
dist_centro	47.6055	16.297	2.921	0.004	15.642	79.569
dm_banheira	50.2987	24.343	2.066	0.039	2.555	98.043
dm_creche	-75.9800	31.246	-2.432	0.015	-137.263	-14.697
dm_duplex	161.6194	68.266	2.367	0.018	27.729	295.510
se_renda	-0.1523	0.040	-3.772	0.000	-0.231	-0.073
suites_4	284.0515	50.229	5.655	0.000	185.538	382.565
suites_2	104.9956	32.181	3.263	0.001	41.880	168.111
suites_3	99.1844	37.640	2.635	0.008	25.362	173.007
suites_1	77.7511	24.712	3.146	0.002	29.283	126.220
garagem_4	594.7200	60.170	9.884	0.000	476.708	712.732
dm_cobertura	-249.9021	56.715	-4.406	0.000	-361.138	-138.666
garagem_2	91.9413	19.513	4.712	0.000	53.671	130.212
garagem_3	222.9649	30.046	7.421	0.000	164.037	281.893
dm_vista_mar	64.5890	26.520	2.435	0.015	12.576	116.602
quartos_4	69.5159	22.549	3.083	0.002	25.291	113.740
<hr/>						
Omnibus:	685.350	Durbin-Watson:	2.007			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	7040.878			
Skew:	1.482	Prob(JB):	0.00			
Kurtosis:	12.208	Cond. No.	6.59e+17			

K.0.3 Tijuca

OLS Regression Results						
Dep. Variable:	preco	R-squared:	0.776			
Model:	OLS	Adj. R-squared:	0.774			
Method:	Least Squares	F-statistic:	447.4			
Date:	Thu, 25 Jun 2015	Prob (F-statistic):	0.00			
Time:	18:33:26	Log-Likelihood:	-16092.			
No. Observations:	2478	AIC:	3.222e+04			
Df Residuals:	2458	BIC:	3.234e+04			
Df Model:	19					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	-1.334e-05	2.18e-06	-6.118	0.000	-1.76e-05	-9.07e-06
dm_churrasqueira	-23.8207	11.772	-2.023	0.043	-46.906	-0.736
suites_4	3390.3091	161.461	20.998	0.000	3073.695	3706.923
suites_2	202.6422	17.077	11.866	0.000	169.155	236.129
preco_lag	0.4759	0.028	17.136	0.000	0.421	0.530
area	3.7075	0.116	32.028	0.000	3.480	3.934
dm_porcelanato	39.2437	16.271	2.412	0.016	7.338	71.149
garagem_4	482.2008	43.995	10.960	0.000	395.931	568.471
suites_1	62.0173	7.662	8.094	0.000	46.993	77.042
garagem_1	82.5785	8.620	9.580	0.000	65.675	99.482
garagem_2	200.0783	12.636	15.833	0.000	175.299	224.857
garagem_3	276.3656	25.482	10.846	0.000	226.398	326.333
dist_favela	85.9926	14.713	5.845	0.000	57.142	114.843
dm_dep_empregada	-18.6246	6.746	-2.761	0.006	-31.853	-5.396
dm_piscina	69.3464	12.880	5.384	0.000	44.089	94.604
se_renda	-0.0448	0.007	-6.118	0.000	-0.059	-0.030
quartos_3	21.1467	8.215	2.574	0.010	5.037	37.256
quartos_4	104.5117	15.405	6.784	0.000	74.303	134.721
dist_metro	-36.1353	4.988	-7.244	0.000	-45.917	-26.354
dm_play	20.7568	8.515	2.438	0.015	4.060	37.454
suites_3	116.5334	53.333	2.185	0.029	11.950	221.116
Omnibus:	1222.766	Durbin-Watson:	2.024			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	44819.939			
Skew:	1.672	Prob(JB):	0.00			
Kurtosis:	23.565	Cond. No.	8.82e+17			

K.0.4 Recreio dos Bandeirantes

OLS Regression Results						
Dep. Variable:	preco	R-squared:	0.738			
Model:	OLS	Adj. R-squared:	0.735			
Method:	Least Squares	F-statistic:	235.2			
Date:	Thu, 25 Jun 2015	Prob (F-statistic):	0.00			
Time:	18:33:06	Log-Likelihood:	-15276.			
No. Observations:	2284	AIC:	3.061e+04			
Df Residuals:	2256	BIC:	3.077e+04			
Df Model:	27					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	-1135.7838	498.712	-2.277	0.023	-2113.765	-157.802
dm_closet	43.8026	14.540	3.012	0.003	15.289	72.316
preco_lag	0.3229	0.049	6.530	0.000	0.226	0.420
area	3.7348	0.099	37.685	0.000	3.540	3.929
dm_elevador	-40.1530	12.272	-3.272	0.001	-64.218	-16.088
dist_delegacia	-70.1052	10.255	-6.836	0.000	-90.216	-49.994
dist_logradouro	57.4010	14.736	3.895	0.000	28.503	86.299
dm_triplex	-228.5499	70.520	-3.241	0.001	-366.841	-90.258
dist_centro	398.7851	132.272	3.015	0.003	139.397	658.173
dist_bombeiro	-30.9431	13.115	-2.359	0.018	-56.663	-5.224
dist_praia	-301.0866	79.708	-3.777	0.000	-457.395	-144.778
dm_duplex	-77.4478	25.717	-3.011	0.003	-127.880	-27.015
dm_varanda	-26.3664	8.832	-2.985	0.003	-43.686	-9.046
dm_terraco	-86.4847	17.651	-4.900	0.000	-121.098	-51.872
dm_sala_o_jogos	32.8371	15.993	2.053	0.040	1.474	64.200
suites_4	440.5724	44.902	9.812	0.000	352.519	528.626
suites_2	136.6508	31.659	4.316	0.000	74.567	198.735
suites_3	204.7612	32.562	6.288	0.000	140.906	268.617
suites_1	61.8267	29.717	2.081	0.038	3.551	120.102
garagem_4	164.2201	25.223	6.511	0.000	114.758	213.682
garagem_1	-29.4435	11.717	-2.513	0.012	-52.422	-6.465
garagem_3	110.2874	12.406	8.890	0.000	85.959	134.615
dm_vista_mar	43.3667	19.427	2.232	0.026	5.270	81.463
dm_sauna	24.3671	10.573	2.305	0.021	3.634	45.100
quartos_3	-118.0474	38.395	-3.075	0.002	-193.341	-42.753
quartos_2	-142.0948	37.952	-3.744	0.000	-216.519	-67.670
dist_metro	-424.4926	157.190	-2.701	0.007	-732.745	-116.240
quartos_4	-131.6048	41.233	-3.192	0.001	-212.463	-50.747
Omnibus:	1124.724	Durbin-Watson:	1.954			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19915.715			
Skew:	1.891	Prob(JB):	0.00			
Kurtosis:	16.963	Cond. No.	1.26e+05			

K.0.5 Botafogo

OLS Regression Results						
Dep. Variable:	preco	R-squared:	0.833			
Model:	OLS	Adj. R-squared:	0.830			
Method:	Least Squares	F-statistic:	297.8			
Date:	Thu, 25 Jun 2015	Prob (F-statistic):	0.00			
Time:	18:31:27	Log-Likelihood:	-8353.6			
No. Observations:	1213	AIC:	1.675e+04			
Df Residuals:	1192	BIC:	1.686e+04			
Df Model:	20					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[95.0% Conf. Int.]	
Intercept	181.5312	100.346	1.809	0.071	-15.344	378.406
garagem_3	345.5150	76.802	4.499	0.000	194.833	496.198
suites_2	163.1523	29.011	5.624	0.000	106.233	220.071
suites_3	121.1842	54.319	2.231	0.026	14.613	227.755
area	7.8333	0.194	40.443	0.000	7.453	8.213
garagem_4	902.9871	108.170	8.348	0.000	690.762	1115.212
dm_elevador	-40.2075	17.353	-2.317	0.021	-74.253	-6.162
dist_saude_publica	151.2516	45.730	3.307	0.001	61.531	240.972
dist_saude_privada	-286.2695	101.195	-2.829	0.005	-484.810	-87.729
garagem_1	154.3974	18.271	8.450	0.000	118.550	190.245
garagem_2	390.0452	26.499	14.719	0.000	338.055	442.035
dist_bombeiro	91.1916	33.814	2.697	0.007	24.850	157.533
dist_favela	124.9822	53.959	2.316	0.021	19.117	230.848
dist_praia	231.2613	64.816	3.568	0.000	104.096	358.427
dm_fundos	-109.4616	43.267	-2.530	0.012	-194.349	-24.574
suites_4	965.2868	135.145	7.143	0.000	700.138	1230.436
quartos_3	55.5185	15.962	3.478	0.001	24.202	86.835
dist_centro	-451.1773	117.001	-3.856	0.000	-680.729	-221.626
suites_1	93.2356	16.414	5.680	0.000	61.033	125.439
dist_trem	368.9874	96.121	3.839	0.000	180.401	557.574
dm_piscina	113.0307	17.966	6.291	0.000	77.783	148.278
Omnibus:	193.398	Durbin-Watson:	1.870			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	992.857			
Skew:	0.635	Prob(JB):	2.53e-216			
Kurtosis:	7.247	Cond. No.	2.79e+03			