RESUMO
Ciência Política na era do Big Data: automação na coleta de dados digitais

Nosso objetivo neste artigo é introduzir uma forma simples de automatizar a coleta de dados na internet para fins acadêmicos. Argumentamos que esta solução trás três principais vantagens em relação à coleta manual: (1) ela reduz drasticamente o tempo gasto na coleta; (2) ela elimina totalmente erros de imputação e de digitação; e, (3), ela aumenta a transparência das pesquisas que a utilizam. Como exemplo, coletamos mais de 25 mil proposições legislativas do site da Assembleia Legislativa de Minas Gerais e oferecemos um passo-a-passo para reproduzir este procedimento no ambiente de programação R. Concluímos discutindo as vantagens desta solução para a Ciência Política, especialmente nesta época em que cada vez mais dados estão disponíveis na internet.

Palavras-chave: coleta de dados. R project, automação, assembleias legislativas.

ABSTRACT

This article introduces a simple way of gathering data on the internet that is useful for academic research. We argue that this method has three main advantages compared to manual data gathering: (1) it dramatically reduces the time spent on the task; (2) it completely eliminates imputation errors; and, (3), it improves data access and research transparency (DA-RT). To exemplify this approach, we collected over 25k legislative proposals from the Assembleia Legislativa de Minas Gerais website, providing a step-by-step tutorial to reproduce our approach using the programming environment R. We conclude highlighting the contributions that this method has to offer to the Political Science.

Keywords: data gathering; R project; web-scrapping; legislative studies.

Ciência Política na era do Big Data: automação na coleta de dados digitais

Denisson Silva[1]
Fernando Meireles[2]

INTRODUÇÃO

Como coletar dados para uma investigação científica?A resposta a essa indagação é fundamental em qualquer incursão metodológica para definição de um desenho de pesquisa. Para trilhar uma resposta a essa pergunta, começamos afirmando que não há uma solução unívoca, pois isso dependerá do tipo de problema de pesquisa, ou seja, do que se quer saber. Um físico pesquisando sobre dilatação do metais não precisará aplicar um survey, do mesmo modo que um cientista político não precisará de um microscópio para pesquisar sobre interações sociais.

Deixando a caricatura de lado, podemos apontar aqui ao menos dois tipos de dados: os observacionais e os de percepção. Os dados de percepção podem capturar informações como a compreensão que os indivíduos têm sobre a realidade a partir dos conhecimentos adquiridos pelos sentidos (visão, audição, tato, paladar) no cotidiano, ou seja, a percepção é uma construção relacional que pode ocorrer através de um relação face-to-face ou de uma relação secundária. Assim, podemos entender como dados de percepção a opinião, as crenças e informações gerais sobre a experiência dos indivíduos (HAIR JR. & Et. al., 2005; SPECK, 2000).

A percepção dos indivíduos sobre algo ou alguma coisa não pode ser observada diretamente, ou seja, dados de percepção não podem ser coletados via procedimento de observação

1 Doutorando em Ciência Política pela Universidade Federal de Minas Gerais (UFMG) e pesquisador do Centro de Estudos Legislativo (CEL/UFMG). Mestrado em Sociologia e Graduação em Ciências Sociais pela Universidade Federal de Alagoas (UFAL).

2 Doutorando em Ciência Política pela Universidade Federal de Minas Gerais (UFMG) e pesquisador do Centro de Estudos Legislativos (CEL/UFMG). Possui Mestrado em Ciência Política pela Universidade Federal do Rio Grande do Sul (UFRGS) e Bacharelado em Ciência Sociais pela Universidade Federal de Santa Maria (UFSM).

direta. Assim, necessitamos de um arcabouço metodológico específico, a exemplo do survey, este método tem sido largamente utilizado para identificar a percepção dos indivíduos sobre corrupção tanto em estudos de escala mundial ou escala regionais e locais, entre essas pesquisas estão da Transparência Internacional, do Latino-Barômetro, World Values Survey (WVS).

A principal vantagem do survey é ir onde a observação não consegue, ou seja, coleta características referentes ao que as concepções políticas, religiosas, como também percepções da realidade se um dado local é mais ou menos democrático, mais ou menos liberal, mais ou menos burocratizado, mais ou menos corrupto. Sua principal desvantagem é que a percepção está sujeita ao tipo de treinamento que o pesquisador teve, como também a influências da mídia, inclusive como o indivíduo entrevistado acordou (HAIR JR. & Et. al., 2005; SPECK; 2000, BABBIE, 1999).

Enquanto que através de observações sistemáticas (de pessoas, de eventos ou fenômenos) e muitas das vezes exaustivas, podendo ser coletados de forma humana, mecânica ou eletrônica, ou seja, os dados de observação podem ser coletados por pessoas em observação atenta ao que outras pessoas e/ou grupos sociais fazem (no caso das ciências humanas), podem também ser coletado de fontes secundárias como a busca por processos de improbidade administrativa na internet – feita por pessoas, ou esses processos de improbidade podem ser coletados de forma automatizada através de algoritmos, e nas três perspectivas não deixaram de ser dados observacionais (HAIR JR. & Et. al., 2005).

Esta observação pode fornecer tanto dados narrativos quanto numéricos. No primeiro tipo podem ser vídeos, fotografias, áudios, registros escritos; no segundo, serão registros de eventos realizados por observadores atentos ou por ferramenta de precisão (como algoritmos) que colete ações específicas como dados eleitorais e de perfil dos candidatos de uma dada eleição, ou padrão de consumo em supermercados virtuais, padrões de interação em redes sociais (twitter, facebook, instagram), noticias em jornais, etc.

Com o advento do desenvolvimento da tecnologia da informação milhões de conteúdo informativo são produzidos a cada segundo, em sites, blogs, redes sociais. No entanto, o problema que se coloca no horizonte dos cientistas é como transformar esses conteúdos em dados que possam ser transforma em conhecimento, em especial, conhecimento sobre os indivíduos, a sociedade, as instituições e as interações sobre estes. É nesse sentido que Baltar&Baltar (2013) coloca que um dos grandes desafios das Ciências Sociais é ter condições para coletar e armazenar os milhões de bytes que estão disponíveis na "era do zettabyte". Por exemplo, no Brasil tem-se desenvolvido a cultura da transparência pública cada vez mais informações sobre o funcionamento do governo estão disponíveis na internet; mas, na maioria das vezes, estes dados não estão sistematizados ou, na pior das situações, não estão nem sequer disponíveis para download direto, que é o mais frequente.

É certo que o menu de técnicas de coletas de dados não está por inteiro contemplado neste artigo, e nem é o objetivo. Nosso objetivo é tratar especificamente de automação de coleta de dados digitais, com o foco em WebScraping, que soluciona a coleta desses dados disponibilizados em sites governamentais e será abordado na próxima seção. A vantagem principal deste método, como ficará claro adiante, é a redução substancial do tempo de coleta de dados, o que justamente o torna útil para se trabalhar com a crescente massa de informações disponíveis na internet: mensagens em redes sociais, artigos em indexadores

acadêmicos, notícias, dados governamentais, listas de itens, serviços de busca, entre outros. Na seção posterior, exemplificaremos isto com a coleta de todas as proposições de projetos de lei, projetos de lei complementar e propostas de emenda constitucional da Assembleia Legislativa de Minas Gerais de 1990 a 2015.

2. WEB SCRAPING

Imaginemos o seguinte problema: queremos analisar o conteúdo das noticias de um jornal online qualquer, como o The New York Times, por exemplo. Suponhamos que queremos todas as páginas que tratam sobre corrupção, que são aproximadamente 100 mil. Para fazer tal análise, precisamos armazenar todas as informações sistematicamente em um banco de dados. Se fizermos isso manualmente, o procedimento é: (1) abrir as 100 mil páginas (já se vão cem mil clicks); (2) copiar todo conteúdo desejado, caso não haja nada para ser separado (aqui já são, no mínimo, 100 mil CRTL+C); (3) colar este conteúdo no software desejado, o que supõe abrir cem mil vezes este software e pressionar CRTL+V 100 mil vezes; (4) por fim, temos que fechar as páginas abertas mais 100 mil vezes. Assim, ao término da coleta temos feito 500 mil operações, só para montar o banco de dados com o conteúdo na íntegra, sem nenhum tratamento adicional.

Então surge a pergunta: é possível automatizar esse procedimento para economizar tempo de coleta e ganhar tempo para a análise? A resposta é sim, através de Web Scraping. Podemos compreender Web Scraping como um conjuntos de técnicas para extrair dados da Web, ou melhor, formas de raspar dados de páginas da internet (HADDAWAY, 2015; KUMAR, 2015; VARGIU & URRU, 2012). No entanto, para fins científicos, nos limitamos aos conteúdos disponíveis aos usuários na exibição em HTML (HyperTextMarkupLanguage), XML (ExtensibleMarkupLanguage) e API (ApplicationPragramming Interface). Nenhuma das técnicas facilmente encontrada para tal operação de coleta de dados captura informações sigilosas dos sites.

Antes de continuar é necessário esclarecer as três siglas acima. A linguagem HTML não é propriamente uma linguagem de programação, embora se tenha avançado nesse sentido com o HTML5. Como próprio nome diz, é uma linguagem de marcação, é o que está por trás do interface do web site que os usuários acessam cotidianamente, seja para ler noticias, ver vídeos, ler e-mail e todas as outras tarefas possíveis na internet. O HTML de um site está acessível para qualquer usuário não somente através da interface, mas também o seu código na integra: geralmente, basta abrir um site qualquer, clicar com botão direito do mouse e clicar na opção "exibir código da página".

Quanto ao XML , esta é uma linguagem para criação de documento hierarquizado, por exemplo, um banco de dados. Seu uso é muito comum, pois não precisa de plataforma de hardware ou de software. Os feeds de noticias de sites e blogs são estruturados em XML; outra aplicação comum são os portais de transparência como o da Assembleia Legislativa de Minas Gerais, que será explorado na próxima seção.

API, não é uma linguagem como as duas anteriores. É um conjunto de instruções e padrões de programação que combina linguagens de programação para

acessar um aplicativo baseado em Web. Ou seja, é uma ferramenta de acesso a determinados conteúdos do site. Por exemplo, com a API do facebook você pode coletar informações sobre as postagem em uma página, ou informações de usuários (nome, sexo, idade, local, etc), fazer analises de redes por exemplo das postagens feitas no twitter , pela API do desenvolvedor. É importante destacar que a API é uma ferramenta (porta de entrada) criada pelo facebook, no segundo pelo twitter.

Para fazer a coleta automatizada por qualquer uma das linguagens ou API disponíveis é necessário criar um algoritmo em uma linguagem que leia as demais. De modo intuitivo, podemos compreender algoritmo como um conjunto de passos para executar uma determinada tarefa, como sacar dinheiro em um banco, por exemplo:

INICIO

1º Ir à instituição bancária;

2º Verificar se a quantia desejada pode ser sacada no terminal de auto-atendimento;

3º Se sim, direcionar-se a um terminal de auto atendimento; se não, ir para um caixa de atendimento ao cliente para ser atendido por um funcionário da instituição;

4º Estando no lugar apropriado para sacar a quantia, aguardar a vez na fila de espera;

5º Quando chegar a sua vez, apresentar-se;

6º Solicitar a quantia desejada;

7º Receber a quantia requerida (o saldo é suficiente para tal);

8º Se o procedimento for feito com uma pessoa; se despesa; se não, não há necessidade;

9º Pode gastar o dinheiro;

10º Volte ao início (se ainda houver saldo);

FIM

O exemplo acima é uma sequência de passos para sacar uma determinada quantia em uma instituição bancária. Podemos pensar algoritmos para todas as nossas atividades do cotidiano; obviamente, algumas são mais fáceis e, outras, mais complexas. Automação de modo geral é identificar e aplicar um algoritmo para um problema específico, e nós aqui apresentamos na próxima seção duas automações para coleta de dados.

As principais vantagens desses modos de coleta são: (1) uma abordagem não-invasiva, (2) os indivíduos observados não são influenciados por atividades relacionadas à coleta de dados e (3) evitar a tendenciosidade dos dados coletados. Ou seja, reunindo esses três elementos, evitamos o viés da pesquisa tanto do pesquisador influenciar nas respostas e/ou comportamento dos indivíduos como o pesquisador ser contaminado por tipos extremos narrados e/ou apresentados pelo observado (HAIR JR. & Et. al., 2005). Podemos também afirmar que este procedimento também (4) evita erros de imputação e (5) aumenta o potencial de replicabilidade de um estudo. Tudo isto garante que o resultado seja o mais fiel possível dos dados disponíveis e cria condições importantes para replicabilidade, que é uma característica desejável em um trabalho cientifico (KING, 1995).

Na próxima seção, vamos apresentar o exemplo de coleta de proposições legislativa da Assembleia Legislativa de Minas Gerais (ALE-MG). Escolhemos esse exemplo porque é a única Assembleia que tem API disponível para coleta de dados[3], sendo que, o API de transparência disponibiliza o banco de dados em dois formatos: XML e JSON[4]. Nosso exemplo explorará em uma primeira etapa a coleta em HTML e, na segunda etapa, a coleta usando o API em formato XML.

3. COLETA DE DADOS LEGISLATIVO SUBNACIONAL

Como citado na seção anterior, para aplicar Web Scraping é necessário o uso de uma linguagem para ler o conteúdo de outras linguagens, por exemplo, PHP[5], Java[6], Python[7] ou R[8], entre outras disponíveis no mercado. Adotamos o R por ser uma linguagem estatística que é multiplataforma, tem pacotes que oferecem muitas outras funções e integração com diversas outras linguagens, além de ser livre e colaborativa.

O exemplo de webscraping que vamos trabalhar aqui se divide em duas etapas: (1) coleta do número, do ano, do tipo e link das proposições de interesse. (2) coleta de algumas informações contidas nos links: tipo, sigla, ano, número, autor, partido, ementa, local fim e situação. Fizemos separadamente as duas etapas para mostrar possibilidades diferentes de fazer a coleta automatizada. Na primeira etapa, vamos coletar os dados direto da página[9]web da Assembleia de Minas Gerais, ou seja, direto do HTML da página; na segunda etapa, vamos usar o API[10] desta Assembleia, que disponibiliza os dados tanto em JSON quanto em XML, para coletar deste último formato, no nosso exemplo. O procedimento que vamos adotar é comentar o código que desenvolvemos para coleta (para facilitar o uso dele por outros, vamos comentar dentro do mesmo, já no formato da linguagem R: o R entende como comentário as linhas iniciadas por #, assim ele não executa como parte do algoritmo).

a) Parte um da coleta:

```
# usamos o pacote  "RSelenium" porque nele já vem incluso outros pacotes
como o  XML", além disso, com ele é possível fazer a coleta direto do
navegador web, aqui usamos o firefox.
library(RSelenium)
# para facilitar a nossa coleta desenvolvemos as quatros primeiras fun-
ções e adotamos a ultima do Stack Overflow, um fórum onde programadores
colaboram expondo e tirando dúvidas (stackoverflow.com)
# A função scrap_value recuperar o conteúdo de um determinado campo, sem
os elementos da linguagem HTML e em uma lista.
scrap_value =function(adress, font){
    x =unlist(lapply(getNodeSet(font, adress), xmlValue))
```

3 Em termos legislativos, outro API disponível é da Câmara do Deputados, que é chamado de dados abertos: http://www2.camara.leg.br/transparencia/dados-abertos/dados-abertos-legislativo

4 Para saber mais sobre o formato JSON, ver: http://www.json.org/json-pt.html

5 Para saber mais sobre PHP, ver: http://php.net

6 Para uma introdução a linguagem Java, ver: http://www.tiexpert.net/programacao/java/

7 Para saber mais sobre python, ver: http://wiki.python.org.br

8 Para saber sobre documentação e pacotes, ver: https://www.r-project.org

9 http://www.almg.gov.br/atividade_parlamentar/tramitacao_projetos/index.html?

10 http://dadosabertos.almg.gov.br/ws/proposicoes/ajuda

```
    x =gsub("[\n;\r;\t]"," ", x)
    x =gsub("\\s+"," ", x)
    }

# A função scrap_attr recupera um valor de um atributo em uma tag, como
o link em um site, que fica no atributo href da tag a, exemplo:
# <a href="http:www.site.com/inicio"> home </a).
scrap_attr =function(adress, font, attr){
  x =unlist(lapply(getNodeSet(font, adress), xmlGetAttr, attr))
  }

# A função str_d1 recorta um elemento da esquerda para direita a partir
de um caractere.
    str_d1 =function(var, string1, n_in){
var_ini =unlist(gregexpr(string1, var))
    var =substr(var,var_ini+n_in,nchar(var))
    }

# A função srt_d2 recorta um elemento da direita para esquerda a partir
de um caractere.
   str_d2 =function(var, string2, n_fin){
var_fin =unlist(gregexpr(string2, var))
   var =substr(var, 1, var_fin-n_fin)
   }

# A função right recorta um elemento da direita para esquerda sem pre-
cisar informar um caractere, apenas especificando a quantidade de carac-
teres desejados.

right =function(string, char){
substr(string, nchar(string) -(char -1), nchar(string))
    }

# Carregadas estas funcões, passamos à coleta das proposições.
# Iniciamos o servidor web com R.
RSelenium::startServer()

# Salvamos as informações do servidor para browser (navegadorinternet),
nesse caso o firefox.
firefox =remoteDriver(remoteServerAddr ="localhost" , port =4444, brow-
serName ="firefox")

# Abrimos uma aba do firefox.
firefox$open()

# Salvando o endereço com os dados que queremos coletar.
baseurl = "http://www.almg.gov.br/atividade_parlamentar/tramitacao_pro-
jetos/index.html?txtAssunto=&txtAutor=&txtIdProj=&txtAno=yyyy&sltTi-
po=tttt&txtEmTram=&txtTramEnc=&txtPeriodoDe=&txtPeriodoAte=&sltSitua-
cao=&txtTramitacao=&txtTh=&search=&ordem=0&advanced=simples&tp=20&txtPa-
lavras=&first=false&aba=js_tabpesquisaSimples&pagina=1"

# data_n_tipo é  um banco de dados vazio que receberá as informações
coletadas.
data_n_tipo =NULL
```

```
# Especificamos os tipos de proposições e os salvamos em um objeto.
tipo_proposi =c("PL", "PLC", "PEC")

# A função for serve para criar um loop, ou seja, repetir o procedimen-
to de coleta quantas vezes se faça necessário. No primeiro for é o das
proposições,o segundo, dos anos de interesses.
  for (t in tipo_proposi){
  for (y in 1990:2015) {

# O objeto url irá substitui no objeto baseurl os valores "tttt" e
"yyyy" pelotipo e o ano.
url =gsub("tttt", t, baseurl)
url =gsub("yyyy", y, url)

# A função firefox$navigate irar inserir o link produzido no objeto url
direto no navegador web.
firefox$navigate(url)

# A função htmlParse irá ler o código fonte (HTML) da página salvar no
R.
  cod_fonte1 =htmlParse(firefox$getPageSource()[[1]])

# No objeto n_pag será salvo o número de páginas nas quais será neces-
sário fazer a coleta.
n_pag =scrap_attr('//div/input',cod_fonte1,"value")[1]

# Com função if será determindo se não houve na página o número de pági-
nas disponíveis, esse valor será 1, porque só há uma página a ser cole-
tada.
if(length(n_pag) ==0L){n_pag =1}

# Com este for, entraremos em cada página de um determinado tipo de pro-
posição e ano.
   for(i in 1:n_pag){

# O objeto url2 substitui a numeração da página na qual será coletada
as informações.
    url2 =url
    url2 =gsub("pagina=1", "xxxx", url2)
    url2 =gsub("xxxx", paste0("pagina=", i), url2)

# Com a função firefox$navigate será inserido o novo link no navegador
de internet.
firefox$navigate(url2)

# No objeto cod_fonte será armazenado o código HTML da página acessada.
cod_fonte =firefox$getPageSource()
cod_fonte =htmlParse(cod_fonte[[1]])

# No objeto resultado 1 será armazenada informações sobre a proposição
legislativa.
   resultado1=scrap_value('//div[@style="float:left; padding:0 18px;
width:100px;"]',  cod_fonte)

# No objeto sep será armazenada uma lista contento o ano, o numero da
proposição, o tipo.
```

```
sep =unlist(strsplit(resultado1, " "))

# Usando o objeto sep será criado o objeto ano, n_proposições, tipo, que
# serão colunas no nosso banco de dados.
    ano =sep[seq(4, 80, 4)]
n_proposicao =sep[seq(3, 80, 4)]
    tipo =sep[seq(2, 80, 4)]

# O objeto link é criado usando os três objetos acimas separados por
"/"(barra).
    link =paste0(tipo, "/", n_proposicao, "/", ano)

# O objeto tabela guardará as informações coletada em cada página.
     tabela =data.frame(n_proposicao =n_proposicao, ano = ano,
tipo = tipo,  link = link)

# O banco de dados da_n_tipo que estava vázio logo no início do código
passa a receber a informações do objeto tabela a cada loop, assim, nofi-
nal será compilado o banco com todos os dados coletados.
data_n_tipo =rbind(data_n_tipo, tabela)

# A função print exibe qual página foi coletada, assim, se acontecer al-
gumerro, é possível indentificar em qual página foi.
print(paste(i, "-",  y, "-", t))
    }
  }
}
# As chaves aqui fecha os três for (loops) iniciados anteriormente.

# Com a função saveRDS salvamos nosso banco de dados no formato RDA,
que é o formato original do R, mas nada impede se for caso salvar quem
qualquer outro formato.
saveRDS(data_n_tipo, "prod_ale_numero_ano_mg_13_12_15.RDA")

# Fim coleta da primeira etapa
```

No exemplo acima, conectamos o R ao firefox para coletarmos os códigos HTML e extrair deles as informações desejadas. O mesmo algoritmo pode ser usado com leves alterações, sem que seja necessário abrir a página no browser. Na próxima, etapa vamos usar o API a Assembleia Legislativa de Minas Gerais para ler os dados armazenados em XML, mas não usaremos o navegador web para abrir a página desejada. No entanto, continuaremos a usar o pacote "RSelenium", pois, vamos continuar usando as funções do pacote XML, e as funções declaradas no início do exemplo anterior. Então é necessário carregar o pacote "RSelenium" e as funções do início.

b) Parte dois da Coleta

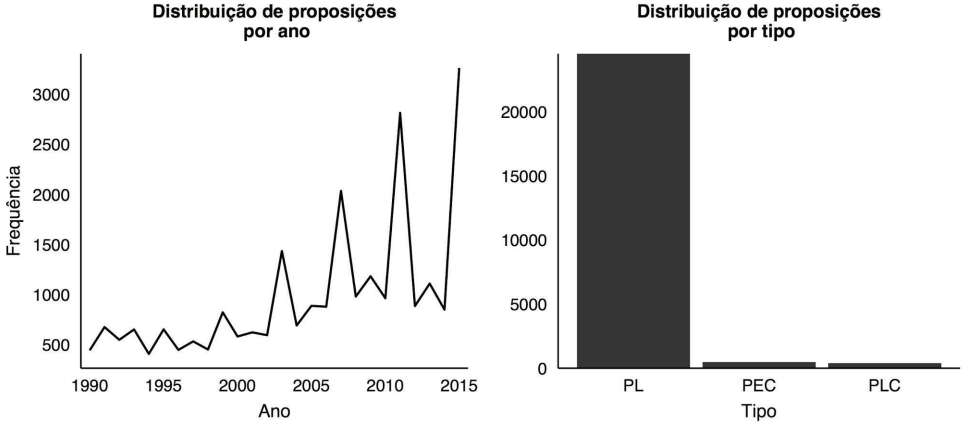
```
# Poderíamos continuar essa nova etapa a partir dos objetos criados na
etapa anterior, mas, vamos usar o banco de dados já salvo. Para isso,
precisamosler esse banco.
       base <-readRDS("prod_ale_numero_ano_mg_13_12_15.RDA")

# No exemplo anterior usamos a "=" igualdade para atribui um objeto,
nesse usamos "<-" tag mais traço, o efeito é o mesmo, fizemos diferente
para mostrar ao iniciante no R ou ao leitor com pouca familiaridade que
```

```
as duasformas de criar um objeto gera o mesmo resultado.

# O objeto baseurl2 guarda o link do API da Assembleia Legislativa de
Minas Gerais.
       baseurl2 <- “http://dadosabertos.almg.gov.br/ws/proposicoes/link”

# O objeto data será o nosso banco de dados, que inicialmente está va-
zio.
  data <-NULL

# O loop será feito usando a variável link do objeto base.
  for (n in base$link){


#No objeto end o link será salvo já com complemento coletado na etapa
anterior
end<-gsub(“link”, n, baseurl2)

# A função xmlParse serve para ele o código XML, na etapa anterior usa-
mos a função htmlParse porque estávamos coletando direto do HTML.
xmlfile<-xmlParse(end)

# O dados de cada proposição será salva no objeto page
page<-xmlRoot(xmlfile)

# A partir daqui, serão coletas as informações desejadas e já salvas no
objetodata, que é o nosso banco de dados. No exemplo anterior, criamos
um objeto tabela antes de transferir os dados para o banco de dados,
aqui fazemos o armazenamento direto.

# Salva o tipo de projeto.
data$tipo[n] <-scrap_value(“//tipoProjeto”, page)

# Salva a sigla.
data$sigla[n] <-scrap_value(“//siglaTipoProjeto”, page)

# Coleta o ano.
data$ano[n] <-scrap_value(“//projetoTramitacao/ano”, page)

# Coleta o número.
data$numero[n] <-scrap_value(“//numero”, page)

#O objeto autorge contem o autor do projeto, mas não salva na base de
dados.
autorge<-scrap_value(“//autor”, page)

# Como existe uma diferença de formatação da informação sobre o autor,
então o
# objeto autor1 identifica se o autor é um deputado ou deputada, mas não
salva
# na base de dados essa informação.
    autor1 <-grep(“DEPUTAD”, autorge)

# Após identificado o cargo do autor, o nome dele é salvo na base de da-
dos na variável autor.
data$autor[n] <-ifelse(length(autor1)==1L,substr(autorge,1,nchar(autor-
```

```
ge)-7),autorge)

# Salva a variável partido.
data$partido[n] <-ifelse(length(autor1)==1L,right(autorge,7),"")

# Salva a ementa da proposição.
data$ementa[n] <-ifelse(length(scrap_value("//ementa", page))==0L,"",s-
crap_value("//ementa", page))

# Salva o ultimo local onde a proposição está na data da coleta.
data$localfim[n] <-scrap_value("//local", page)[1]

# Salva a situação da proposição.
data$situacao[n] <-scrap_value("//situacao", page)

# print mostra que proposição já foi coletada.
print(n)
    }
# A chave encerra o loop, no exemplo anterior tínhamos três loops, ou
seja, três chaves.

# A função write.csv2 salva o obejto data em arquivo no formato csv que
podeser aberto por todos os principais software estatístico.
write.csv2(data,"prod_ale_mg_13_dezembro.csv", fileEncoding ="ISO-8859-
1")

# Por fim, salvamos o arquivo no formato padrão do R.
saveRDS(data,"prod_ale_mg_13_dezembro.RDA")

# Fim da coleta.
```

Com os dois algoritmos coletamos 25.418 proposições legislativa da Assembleia de Minas Gerais, entre 1990 e 2015, em aproximadamente 2h.:00min. Os dados disponibilizados pela Assembleia vão desde 1958. Os gráficos abaixo sumarizam algumas informações descritivas dos dados coletados. Além dos dados da Assembleia Legislativa de Minas Gerais, foram coletados de mais sete estados usando essa técnica e estão sendo usados para produção acadêmica do Centro de Estudos Legislativo (CEL).

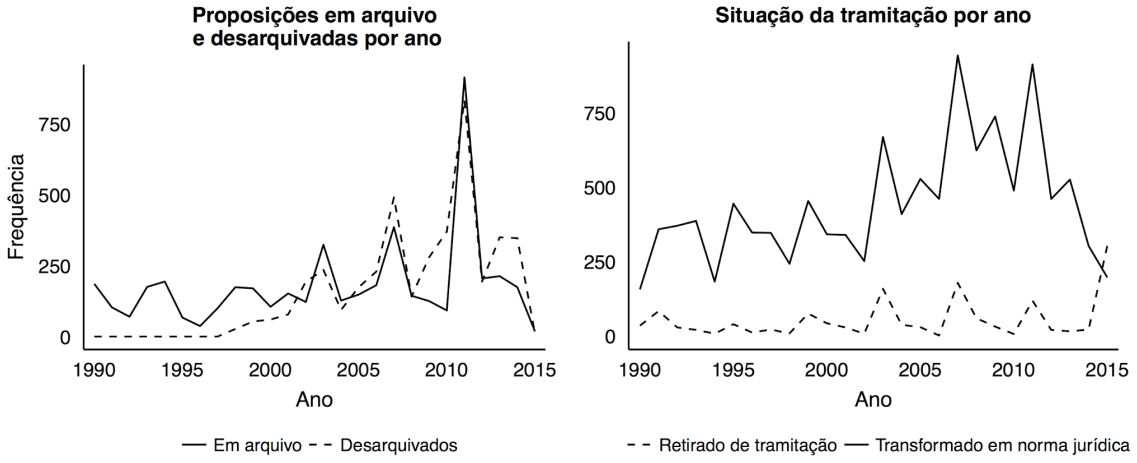
Gráfico - 1: Distribuição de frequência das proposições por ano e tipo



Fonte: Elaboração dos Autores a partir de dados da Assembleia de Minas Gerais

A partir do gráfico 1, podemos ver que não há um crescimento linear da quantidade de proposições legislativa entre 1990 e 2015. No entanto, é possível visualizar que em ano eleitoral há uma queda da quantidade proposições. Considerando três tipos de proposições, vermos que projetos de lei (PL) é majoritariamente o tipo de proposição mais realizada na Assembleia Legislativa de Minas Gerais (ALE-MG).

Gráfico - 2: Distribuição da proposição em arquivo ou desarquivadas e situação de tramitação, por ano.



Fonte: Elaboração dos Autores a partir de dados da Assembleia de Minas Gerais

Já no gráfico 2, podemos observar que o número de proposições em arquivo é maior do que as desarquivadas até os anos 2000, depois desse ponto, é muito parecida a distribuição dos dois tipos. Enquanto a situação de tramitação das proposições legislativa foram mais transformada em norma do que retirado de tramitação para todo o período com exceção do ano 2015. É importante destacar que não pretendemos fazer nenhuma inferência com os dados acima nesse artigo, mas, apenas mostrar o que pode ser coletado em poucas horas, e dependendo do caso em minutos.

CONSIDERAÇÕES FINAIS

Atualmente, a produção de dados digitais é gigante, e uma parte significativa desses dados interessa à Ciência Política, a exemplo dos dados das instituições governamentais (Executivo, Legislativo e Judiciário). Por outro lado, grande parte deste volume de dados não estão disponíveis para download em um formato já aceito por softwares estatísticos, mesmo nas plataforma que disponibilizam um API. Assim, precisamos, enquanto cientistas políticos, investir no aprendizado de ferramentas capazes de fazer esse tipo de coleta.

Aqui, abordamos sobre a automatização de coleta de dados e apresentamos um exemplo usando o software R com o auxilio do pacote "RSelenium", para possibilitar tanto a coleta dos dados disponíveis na web quanto a sua automatização. Esse tipo de metodologia de coleta nos possibilita reduzir o tempo necessário gasto na etapa importante da pesquisa empírica. Uma outra vantagem é a eliminação de erros de imputação, digitação ou até mesmo do copiar e colar, pois, fazendo isso repetidas vezes, o pesquisador cansado pode errar o local onde colar uma determinada informação coletada. E, sem dúvida, esta técnica também aumenta a transparência do procedimento científico, o que facilita a replicabilidade, que é uma característica tão deseja da pesquisa científica.

No exemplo apresentado, foi possível resolver uma série de problema de coleta. Para cada site será uma solução diferente, e aqui apresentamos os dois caminhos dentre uma grande diversidade disponível, tanto gratuito quanto comercial. A solução apresentada possibilitou coletar 25.418 observações (linhas) e 9 variáveis (colunas) em 2h, ou seja, se fossemos copiar e colar seria necessário pelo menos repetir o procedimento 2.287.762 vezes. É importante destacar que os interessados em aprofundar na técnica apresentada aqui deve iniciar os estudo de introdução ao HTML, XML e ao R.

REFERÊNCIAS

BABBIE, Earl. (1999), Métodos de pesquisas desurvey. Ed. da UFMG.

BALTAR, Ronaldo, and BALTAR, Cláudia S. (2013), "As Ciências Sociais na Era do Zettabyte." Mediações-Revista de Ciências Sociais 18.1 p.11-19.

HADDAWAY, Neal R. (2015), "The Use of Web-scraping Software in Searching for GreyLiterature." GreyJournal (TGJ) 11.3.

HAIR JR., J. F.; BABIN, B.; MONEY, A. H.; & SAMOUEL, P. (2005), Fundamentos de método de pesquisa em administração. Porto Alegre: Bookman.

KING, Gary. (1995), "Replication, replication." PS: Political Science &Politics 28.03, p. 444-452.

KUMAR, ShyamNandan. (2015), "World towardsAdvance Web Mining: A Review." American Journalof Systems and Software 3.2, p. 44-61.

SPECK, B. (2000), Mensurando a Corrupção: uma revisão de dados proveniente de pesquisas empíricas. Cadernos Adenauer. ISSN 1519-0951.

VARGIU, Eloisa, and URRU, Mirko. (2012), "Exploiting web scraping in a collaborativefiltering-based approach to web advertising." Artificial IntelligenceResearch 2.1, p.44.

UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

CFCH
CENTRO DE FILOSOFIA E CIÊNCIAS HUMANAS

Departamento de
Ciência Política

Programa de Pós-Graduação
em Ciência Política