talk

nekota

2024-02-01

1　今日のテーマ

データ分析をするには、当然データが必要になります。

ですから、必ずそのデータを準備する作業が必要になります。

そこで、今日のテーマは「データを準備する」部分に着目します。

その分析対象となるデータですが、

じゃ、「コレを分析してください」という形で、はじめから用意されているわけではありません。

まず、

- 何処にあるのかを探す
- それを手に入れる
- 分析できる形にする

という「準備作業」が必ず必要になります。

さて、「作業」というのは、どレくらいのイメージでしょうか？

実は意外と準備のために扱うデータは大量にあります。

どれくらい大量かといえば、通常は、手作業でやるのが嫌になるくらいです。そして、ある一定の量を超えると、手作業では根性があっても実質的には無理なくらいの作業になります。

そこで、今日のテーマは、

大量のデータを効率的に処理する方法に着目し、

その方法を具体的に紹介することにしました。

で、今回は

- 人口データ

を扱ってみました

ここで、

データ分析とデータの具体的なイメージとして、地価公示を担当されている先生方には馴染み深い「価格動向報告」にある人口の推移の分析を思い浮かべてください。

人口データは、不動産の受給動向の要因として分析される馴染み深いデータの一つです。

人口データはどんなものを想像するでしょうか？

1.1　人口データ

「人口データ」とはどんなものか具体的に見てみます。

- 総人口
- 男女
- 年齢別（５歳区切り、年少、生産、老齢）
- 増減（自然、社会）
- 世帯

86 地域、5 年（６０ヶ月）以上のデータ

これらの項目について、年若しくは月毎に大阪府下の各市町村について計測、若しくは、推計されその数値がまとめられています。

大阪府下の各市町村については大阪府の Web ページで公開されています。

https://www.pref.osaka.lg.jp/toukei/jinkou/jinkou-xlslist.html

このページを開いて、具体的なデータを少しだけ覗いてみましょう。

1.2　意外と大量のデータ

大阪府のページを見ながら、ダウンロード。

まず、全部のデータをダウンロードするのが煩雑です

＜実際に開く＞

新しいデータ（月次）

古いデータ（年毎で、シートが月次）

年齢別ファイル（縦に総合、男、女）となっている

等、ファイルによって違いがあることを確認

実際のファイルの中身を少し見てみると、年によって形式が違うことがわかります。

人口のデータを分析するためには、データがどんなかたちであるとよいのか？

例えば、大阪府全体の人口の推移を見たい場合、、、

大阪府全体の人口を月毎に横か縦に並べたい

ファイルは月毎（シート毎）なので、各ファイルから大阪府全体のデータを取り出し、順番に並べる必要がある。５年ならば６０行（列）。地域全てをやろうと思うと、この作業を地域分である８６回繰り返す。

セル毎のコピーを１秒間に１回やったとしても、多分１時間半以上、この単純作業を繰り返すことになります。

これがデータ分析にあたって、必ずしなければならないデータの準備であり、先に言ったとおり、これらの作業は、手作業でやるのが嫌になる（若しくは、不可能）なくらい、**大量のデータ**を処理する必要が生じます。

この作業は、１回だけならば、頑張って準備すればよいのですが、準備すべきデータの形というのは、分析したいものによって変わります。

つまり、同じようなデータソースであったとしても、やってみたい分析によって準備すべきデータの形が変わります。この時に、データの形を変えるのに時間がかかるとすると、データの分析自体が時間がかかってしまい。面倒な分析になってしまうことになります。

一方で、短期間でデータの準備が出来れば、いろんな分析をすぐに出来るようになり、結果として、そのデータから得られるものが多くなるのです。

なので、データの準備の効率化はデータ分析の効率化に繋がります。

2　プログラムによる自動化

「データの準備とは」をまとめれば

- データの入手
- 入手したデータの内容確認
- データ分析しやすい形へデータを編集

全ての場面でプログラミングが役に立ちます

今日もデータ分析を得意とする R 言語での例

簡単に一つづつ見ていきます

2.1　データの入手

手作業では、クリックしていますが、

プログラムではまず、

ページの情報を取得して

それをもとにファイルをダウンロードする

という、方法が一般的。

2.2　入手したデータ内容の確認

先に確認したとおり、ファイルの形式はいろいろです。

それらを目視で確認するだけでなくプログラム的に確認できます。

例えば、地域の割り振りが全て同じなのか？等

その前提としてまず、エクセルファイルを R のプログラム内に読み込む必要がありますが、R ではコレを簡単に行うことが出来るパッケージがあります。

- データファイルを R のデータとして読み込む
- データの構造を把握
- どのように編集するかを決める

2.3　データ分析しやすい形へデータを編集

- データを R のプログラムで編集
- データをエクセル等のファイルとして出力

R プログラム内で編集し、最終的に R ユーザーでない人でも使うことが出来るエクセルファイルとして出力します。

R のデータをエクセルのファイルとして出力させることも R では簡単に出来ます。

2.4　具体的な成果

https://github.com/rea-osaka/seminar20240207

今日のセミナーで話している、「データの準備」に関する一つの成果として、大阪府で配布されている人口データを自動収集して、分析しやすい一つの形として取りまとめたものを出力するためのプログラム例を作成しました。

＜実際にアクセスしてディレクトリを説明＞

しかし、今日はこれのこまかな話をするのではなく、データ準備の中でよく利用されるプログラムの技術についての話をします。

プログラムってこんなことができるんだ！という興味をもっともらえたら良いなと思っています。

3　データを入手する技術

まずは、データを入手する時点の技術に関して話です。

大まかに 2 つの方法があります。

- API(Application Programming Interface)
- ページの構造を読み解くスクレイピング

ひとつは、データを直接やりとりするためのサービスというものがあります。情報を配布する側も機械化に対応した API という方法は、ダイレクトにデータを得ることが出来ることが多いです。

不動産取引価格情報の API

もうひとつは、普通の人が見るような形式の Web から、必要な情報を取得する html ページの構造を読み解くスクレイピングという方法を使う。

きょうは、スクレイピングの方

3.1　スクレイピングとは

スクレイピングとは、一般に、Web ページ上の情報を機械的に取得することを言います。

有名なスクレイピングパッケージ

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- https://rvest.tidyverse.org/

スクレイピングはあちこちで色んな形で紹介されていて情報は多い方

例えば、大阪府の人口データのページで、人口データのエクセルファイルへのリンクが沢山ありますが、このリンクが指している URL 自体を、プログラムで読み取ることが出来ます。

メジャーな高級プログラム言語、例えば、Python や Ruby、Go、そして、R 言語にも、スクレイピングを行うための関数が用意されていますので、これを利用することで比較的簡単にスクレイピングが行えます。

3.2　スクレイピングの仕組み

スクレイピングを行うための前提知識として、Web ページが **html 言語**で書かれており、各内容が html タグで構造化されている事を知っている必要があります。

例えば、html でのリンクは **a タグ**で、表現され、その href 属性にリンク先の URL が書かれています。

```
<html>
...
  <body>
  ...
  <a href="https://google.com" >google のページ</a>
  ...
  <a href="https://www.yahoo.co.jp/" >Yahoo  Japan のページ</a>
  ...
  </body>
</html>
```

なので、ページの中にある全ての a タグの href リファレンスを調べることで、ダウンロードしたいファイルの URL を収集することが出来るのです。

3.3　R 言語でスクレイピング

R 言語の場合、rvestパッケージにスクレイピングの関数があります。

```
library(rvest)
library(tidyverse)

# 人口データを公開している大阪府の URL
target_url <- "https://www.pref.osaka.lg.jp/toukei/jinkou/jinkou-xlslist.html"

# ページを取得
node_data <- read_html(target_url, encoding = "shift_jis")

# 構造で選別
href_data <-
    node_data %>%
    html_nodes("tbody td a") %>%
    html_attr("href")
```

- read_html() 関数
- html_nodes() 関数
- html_attr() 関数

実際にどんなものが得られるのか見てみましょう

samplecode001

3.4　ある規則の文字列を表現する正規表現

ページ内のリンク先を全て集めると、欲しい URL 以外の URL も一緒に集まります。この時、欲しいデータの共通項は最後が「xlsx」となっているリンク先なので、これだけを抜き出す必要があります。

プログラムでは一般的に、ある規則になっている文字の並び方だけを取り出したい時に例えば、

- “https” で始まっている
- “.xlsx” で終わっている
- “携帯電話の番号みたいな文字列”

これら**正規表現**を利用して、文字列の選別を行えます。

実際の表記は以下のようなもの

- “^https”
- “\.xlsx$”
- “0[89]0-[0-9]{4}-[0-9]{4}”

3.5　正規表現を使った文字列比較

R 言語の場合、stringrパッケージに使いやすい文字列処理の関数があります。

- str_subset() 関数

```
library(stringr)

#必要なリンクのみを取り出す
link_data <-
    href_data %>%
    stringr::str_subset(".xlsx$") %>%
    stringr::str_subset("5sai", negate = TRUE)
```

実際にどのように選別されるか確認します

samplecode001

3.6　スクレイピングで得られるもの

Web ページはいわゆる「文字列」で情報が表現されています。そして、ここで得られるものは、欲しいデータの在処を示した URL の文字列です。

先の R 言語の例示のコードでは、最終的に、URL 文字列（そのベースの一部）のベクトル（配列）が得られます。

3.7　データをダウンロードする

インターネット上のファイルをダウンロードする関数もプログラム言語には基本的に用意されています。R 言語の場合は、基本的なパッケージ (utils) の中にある download.file() 関数で、ファイルのダウンロードが出来ます。

つまり、プログラムでファイルをダウンロードできます

- download.file() 関数

```
# ダウンロードしたいファイルの URL
target_dl_url <- "https://www.pref.osaka.lg.jp/attach/3387/00014690/jk20240101.xlsx"

# ファイルをダウンロードする
download.file(url = target_dl_url, destfile = basename(target_dl_url))
```

basename() 関数をつかって、ファイル名を作成

この関数を使う場合、必ず、保存するファイル名を明示しなければなりません。

3.8　複数ファイルをダウンロードする

当然、プログラムでダウンロードする利点は、自動的に複数のダウンロードを行うことです。

複数ファイルの自動ダウンロード

```
# ダウンロード先の URL の集合
urls <- link_data

for (i in seq_along(urls) ){

  #URL 文字列から最後の部分をファイル名として取り出す
```

```
  dest_path = basename(urls[i])

  # ファイルをダウンロードする
  download.file(url = urls[i], destfile = dest_path)
}
```

実際には、完全な URL が必要なので link_data を加工する必要があります。

4　エクセルファイルの読み込み

プログラムで処理するには、まず、読み込みが必要

R プログラム内では、data.frame というデータ構造で扱う

R 言語では、openxlsxパッケージが有名です。

人口データに関しては、複数のデータファイルがあります。これらのデータを統合するためには、データを加工して結合する必要があり、それらを行うためには、まず、**データを読み込む**必要があります。

```
library(openxlsx)

# エクセルファイルを読み込む
df_data <- read.xlsx("data/data.xlsx")
```

samplecode001.Rmd

4.1　細かな指示も行える

- シートの扱い
- 列目の扱い
- 読み込むセル範囲の指定

国勢調査年のファイルを読み込む際のサンプルです

censusdata_dlcode.nb.html にあります

```
# シートの指定やセルの指定をしたもの
df_data <-
  openxlsx::read.xlsx(
    file, colNames = FALSE,
    sheet = "表 9-1",
    rows = c(6:56,63:103))
```

4.2　ファイル読み込みのサンプルコード

read.xlsx() 関数に渡す引数で、エクセルファイルからどのシートを読み込むかや、行や列名をどうするか、セルのどの位置を読み込むか等を細かく指定できるので、実際に読み込むエクセルファイルのデータの配置に合わせて、プログラムで調節することが出来ます。

以下のコードは、国勢調査年のエクセルシートを読み込む際のサンプルです

```
# シートの指定やセルの指定をしたもの
df_data <-
  openxlsx::read.xlsx(
    file, colNames = FALSE,
    sheet = "表 9-1",
    rows = c(6:56,63:103))
```

5　データの整理

data.frame 構造のデータを操作する関数

関数	機能
filter()	行を選別
arrange()	行を並び替え
select()	列を選別
mutate	新しい列を作成

R 言語では、dplyrパッケージに属しています。

通常は「tidyverse」を呼び出す。

census_….html を見ながら実際のコード

エクセルのファイル

ファイル読み込みサブルーチン

5.1　複数のデータを貼り付け

複数の data.frame データを統合する関数

関数	機能
rbind()	表の下に追加するイメージ
left_join()	表の右に追加するイメージ

5.2　公開された技術書

インターネットで公開された書籍

R for Data Science (2e)

第 3 章の「Data transformation」が秀逸

英語が苦手なら ChatGPT でサポートしてもらう

6　エクセルファイルとして出力

データの書き出しもopenxlsxパッケージの関数を利用

- write.xlsx() 関数

```
library(openxlsx)

# 内蔵データ iris をエクセルファイルへ書き出し
write.xlsx(iris,"hoge.xlsx")
```

これは、R の data.frame データを扱う場合

6.1　エクセルのデータ構造を扱う

プログラム内部で、エクセルのデータを作成し、これをエクセルファイルとして書き出す方法があることを把握しておきましょう。

data.frame とエクセルの構造は違う

エクセルファイルの構造

- シートがある
- セルがある
- 見た目の属性

これを R のプログラムの中に作ることが出来る

```r
# エクセルデータを作る
obj <- openxlsx::createWorkbook()

# 枚方市という名前のシートを追加
openxlsx::addWorksheet(obj, sheetName = "枚方市")

# 枚方市という名前のシートに人口データを追加
# census_data が R の data.frame データ
openxlsx::writeData(obj, sheet = "枚方市", x = census_data)
```

6.2 エクセルデータをファイルに出力

プログラムの中で作ったエクセルデータを現実のエクセルファイルとして書き出します。

```r
# エクセルファイルとして書き出す
openxlsx::saveWorkbook(obj, file = "hirakata.xlsx", overwrite = TRUE)
```