# 応用プログラミング3
# 第12回　Rによる Webスクレイピング

専修大学ネットワーク情報学部
田中健太

# 1．課題の振り返り

1

# 2. Webからのデータ収集

2

今回は、Rを使って、Webからデータ収集する方法を紹介します。前半で、Webサイト・サービスが用意したAPIを利用する方法、後半でWebサイトをスクレイピングして必要な情報を抽出する方法を取り上げます。

## 2.1　データ収集の方法

- 研究においてもビジネスにおいても、Web上で公開されているデータを使用する機会は多い
- 公的統計、オープンデータ、企業サイト、SNS…
- Web上のデータを収集する方法として、主に2つの方法がある
- APIにアクセスする
- スクレイピングする


3

研究でもビジネスでも、データを活用して何か知見を得たいという場合に、まずは自分たちで集めた一次データを使います。実験やアンケート、あるいは社内の業務を通じて発生したデータです。しかし、それ以外に、自分たちでは収集できない、国や自治体規模の大量のデータや、SNSにおけるさまざまな発言、行動を分析したいこともあります。

国や自治体による調査結果は、オープンデータとして広く公開されています。国の統計データはe-StatというWebサイトに集約されています。

- e-Stat 政府統計の総合窓口 https://www.e-stat.go.jp/

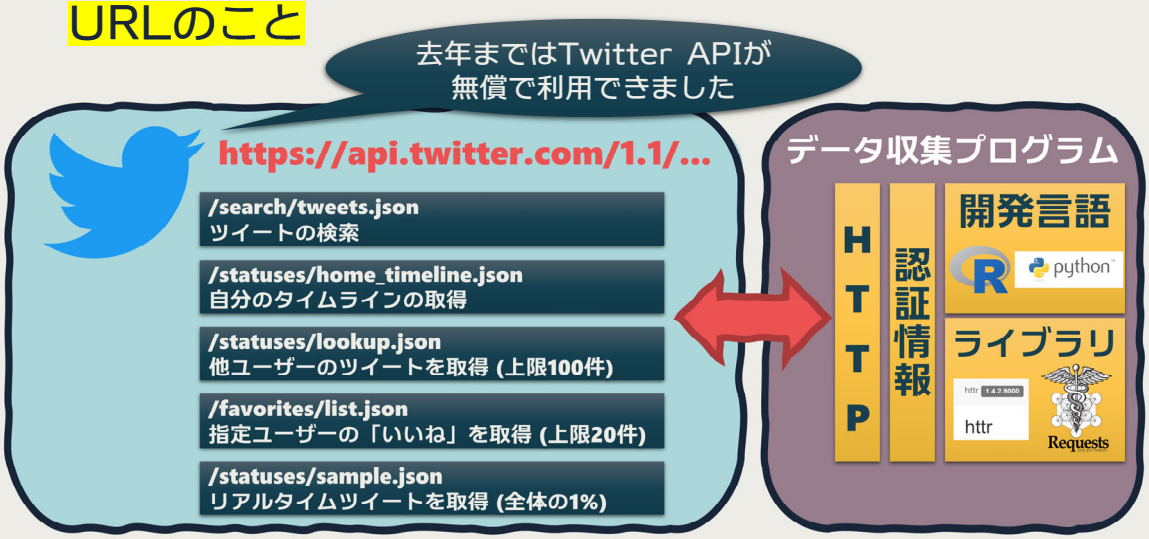
都道府県、市町村レベルの統計データなどは、それぞれの自治体のWebサイトなどで公開されています。

- 東京都オープンデータカタログサイト https://portal.data.metro.tokyo.lg.jp/
- 神奈川県オープンデータサイト https://www.pref.kanagawa.jp/dst/index.html
- 長崎市オープンデータカタログサイト https://odcs.bodik.jp/422011/

また、SNSや各種のWebサービス事業者も、利用者のデータなどを公開し、サービスの価値向上を図っています。これらの公開データの多くは、利用者の利便性とサーバー側の負荷低減のため、APIを介してアクセスできるようになっています。

## 2.2　APIによるデータ収集

- Application Programming Interface
- ITシステムの機能にアクセスするための統一的な手法
- Webにおいては、アクセスするとデータが取得できるURLのこと



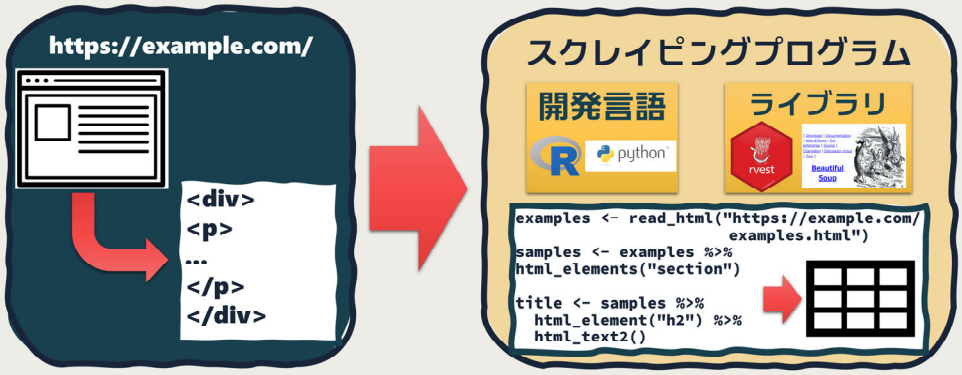

4

APIとは、Application Programming Interfaceの略称で、Webに限らずコンピューター周辺で一般的に使われる言葉です。大まかには、「他のプログラムとの通信のための仕組み」と言えるでしょう。「こういう形でアクセスしたら、こういうデータを返すよ」というルールを定めたもので、WebサービスのAPIであれば、HTTP (S) で指定したURL、パラメーターでアクセスすれば、サービスが規定した仕様でデータが取得できます。Webではない、社内ネットワークなどに閉じたITシステムでも、機能を分担するサーバー間でAPIを介した通信によって、データをやり取りしています。もっとミクロな、プログラム言語のライブラリが提供する機能にアクセスする仕組みもAPIです。

- 参考: 0からREST APIについて調べてみた
  https://qiita.com/masato44gm/items/dffb8281536ad321fb08

## 2.3　スクレイピングによるデータ収集

- APIが提供されていないサイトで、HTMLを解析してデータを入手する方法
- RやPythonからWebアクセスして取得したHTMLから、所望の要素を抽出する
- User-AgentやRefererなどで人間のブラウザアクセスと同じようにならない場合も




5

技術的な問題やコスト面から、APIが提供されていないWebサイトも多くあります。そのようなサイトで利用したいデータが公開されている場合、ブラウザでアクセスし、リンクをクリックしてダウンロードすることになります。ダウンロードするデータが少なかったり、アクセス頻度が低ければ、それでもよいですが、頻繁に更新される大量のデータを入手したいという時には不便です。

そこで、プログラムを使って自動的にWebサイトにアクセスし、HTMLを解析して必要なリンク、文字列を探し出してファイルのダウンロードなどを行います。これを、Webスクレイピング (scraping) と言います。スクレイピングには大きく2つ、パッケージ・ライブラリが提供するHTTPクライアント (ブラウザ) を使ってアクセスする方法と、Chromeなど人間が利用するブラウザを自動操作してアクセスする方法があります。パッケージ独自のHTTPクライアントを利用するほうが簡単ですが、サイト側でUserAgentなどによってプログラムによるアクセスと判定され、接続拒否される場合があります。また、JavaScriptなどを完全にサポートしてはいないので、サイトの構成によっては必要な情報に辿り着けないことがあります。一方、Chromeなどを操作する場合、人間によるアクセスと同じように振る舞わせることができますが、設定がやや面倒です。

まずはパッケージが提供するHTTPクライアントでアクセスしてみて、上手くいかないようであればブラウザの自動操作を検討するとよいでしょう。

# 3. httrパッケージによるAPIアクセス

6

ここから、RでWebアクセスする際に使われる、httrパッケージの使い方を紹介します。httrパッケージは、tidyverseなどとは別に、Rをより便利に活用するためのパッケージ群を開発する、r-libプロジェクトによって提供されています。こちらも、Hadley Wickhamらが中心となって活動しています。

- Tools for Working with URLs and HTTP • httr https://httr.r-lib.org/

なお、httrパッケージをリライトしてより便利になった、というhttr2パッケージも開発されています。今後はこちらが主流になるかもしれませんが、まだ情報が少ないため、今回はhttr (1) パッケージを使います。来年の授業では変わっているかもしれません。

- Perform HTTP Requests and Process the Responses • httr2 https://httr2.r-lib.org/

LinuxやmacOSのコマンドラインで広く使われている、curlというHTTPクライアントをRから利用するためのパッケージがいくつか存在しますが、いずれもやや設計が古く、モダンなAPIアクセスなどには不便なので、httrパッケージはそれを代替するために開発されています。

httrパッケージもcurlのライブラリ (libcurl) を使っているため、HTTPアクセスした際のUserAgentは、以下のように標準ではcurlのものになります。そのため、サーバーの設定によってはアクセス拒否されることもあります。

```
"libcurl/7.64.1 r-curl/4.3.3 httr/1.4.4"
```

# 3.1　エンドポイントを確認する

- どんなAPIを使うにも、まずエンドポイント（URL）を確認する
- 各サイトの「開発者向け情報」に記載されている
- 多くは、ユーザー登録とアクセスキーの取得が必要


7

httrパッケージに限らず、何らかのプログラムからAPIにアクセスするには、まず接続先のURL（エンドポイント）を確認する必要があります。エンドポイントの情報は、各Webサービスの開発者向けページで公開されています。

また、ほとんどの場合、APIにアクセスするためにはOAuthなどの認証情報が必要です。開発者向けページでユーザー登録し、アクセス許可を得ます。

- 参考1: 新しい気象庁サイトからJSONデータが取得できる件 | MindTech
  https://mindtech.jp/?p=1754
- 参考2: データセット - 公共交通オープンデータセンター データカタログサイト
  https://ckan.odpt.org/dataset

## 3.2 GET() 関数による受信（1）

- API経由でデータを "受信" したい場合、GETリクエストを送信する
- httrパッケージでは GET() 関数を使う
- 返り値はAPIによってJSONやXMLなどさまざまなので、他のパッケージ、関数で処理する
- JSON，XMLの取り扱いについては第8回の資料を参照

```
library(httr)

url <- "https://httpbin.org/get"
res <- GET(url)
res

## Response [https://httpbin.org/get]
##   Date: 2022-12-10 16:36
##   Status: 200
...
```

8

ここから、httrパッケージの関数を紹介します。最も一般的なHTTPアクセスであるGETリクエストは、GET() 関数で実行できます。引数にURL（エンドポイント）を指定すれば、レスポンスが取得できます。何がレスポンスとして返ってくるかはAPI次第なので、あらかじめドキュメントを確認し、オブジェクトに格納したレスポンスを、適切な方法で処理します。また、後述する他の関数と共通で、config オプションにリストで認証情報やリファラーなどさまざまな指定ができます。なお、config という文字列は省略可能です。

- 参考: GET a url. — GET • httr https://httr.r-lib.org/reference/GET.html

ここでは、さまざまなHTTPアクセスの方法をテストできる、httpbinにアクセスしています。/get では、GETリクエストについて、その内容をそのままレスポンスとして返します。

- httpbin.org https://httpbin.org/

# 3.2 GET() 関数による受信（1）

```
# リファラーを付与する例
res <- GET(url, config = add_headers(referer =
"https://www.google.co.jp/search?q=httr+r"))
res

## Response [https://httpbin.org/get]

##   Date: 2022-12-10 16:36

##   Status: 200

...

##     "Host": "httpbin.org",

##     "Referer": "https://www.google.co.jp/search?q=httr+r",

##     "User-Agent": "libcurl/7.64.1 r-curl/4.3.3 httr/1.4.4",

##     "X-Amzn-Trace-Id": "Root=1-6394b5a5-67213f8241e70362379d49de"

##   },

## ...
```

# 3.2 GET() 関数による受信（1）

```
# UserAgentを設定（偽装）する例
res <- GET(url, user_agent("httr package"))
res

## Response [https://httpbin.org/get]
##   Date: 2022-12-10 16:36
##   Status: 200
...
##     "Host": "httpbin.org",
##     "User-Agent": "httr package",
##     "X-Amzn-Trace-Id": "Root=1-6394b5a6-554abd762428450d72e11933"
##   },
##   "origin": "106.73.7.130",
## ...
```

## 3.3 GET() 関数による受信（2）

- HTTPレスポンスヘッダーは、headers() 関数や status_code() 関数で確認できる
- ボディは content(オブジェクト, "text") で取得できる
- クエリを付与したアクセスもできる

```
# HTMLを取得し、xml2::read_html() で処理
url <- "https://httpbin.org/html"
res <- GET(url)
headers(res)

## $date

## [1] "Sat, 10 Dec 2022 16:36:54 GMT"

##

...
```

11

レスポンスのヘッダー情報は、headers() 関数で確認できます。また、ステータスコードは、status_code() 関数で確認できます。

レスポンスのボディ（本体）にアクセスするには、content() 関数を使います。ヘッダーのContent-Type属性に合わせて、テキストとして出力するか、バイナリデータとして出力するかは判定されます。また、HTMLやCSV、JSONなどのよく知られたフォーマットについては、readrパッケージの read_csv() など、対応する関数を自動的に呼び出して処理してくれます。逆に、レスポンスをテキストのベクトルとして取得したい場合は、as = "text" と指定します。画像などをバイトデータとして取得したい場合は、as = "raw" とします。

HTTPリクエストにクエリを付与してアクセスすることもできます。GET(..., query = list(キー = 値)) と指定すると、https://.../?key=value といったリクエストになります。

# 3.3 GET() 関数による受信（2）

```
# HTMLを取得し、xml2::read_html() で処理
url <- "https://httpbin.org/html"
res <- GET(url)
status_code(res)

## [1] 200

content(res)

## {html_document}

## <html>

## [1] <head><meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"> ...

## [2] <body>¥n      <h1>Herman Melville - Moby-Dick</h1>¥n¥n      <div>¥n
...
```

# 3.3 GET() 関数による受信（2）

```
# HTMLを取得し、テキストとして出力
res <- GET(url)
content(res, "text")
```

```
## [1] "<!DOCTYPE html>¥n<html>¥n  <head>¥n  </head>¥n  <body>¥n
<h1>Herman Melville - Moby-Dick</h1>¥n¥n      <div>¥n        <p>¥n
Availing himself of the mild, summer-cool weather that now reigned in these
latitudes, and in preparation for the peculiarly active pursuits shortly to
be anticipated, Perth, the begrimed, blistered old blacksmith, had not
removed his portable forge to the hold again, after concluding his
contributory work for Ahab's leg, but still retained it on deck, fast
lashed to ringbolts by the foremast; being now almost incessantly invoked
by the headsmen, and harpooneers, and bowsmen to do some little job for
them; altering, or repairing, or new shaping their various weapons and boat
furniture. Often he would be surrounded by an eager circle, all waiting to
be served; holding boat-spades, pike-heads, harpoons, and lances, and
jealously watching his every sooty movement, as he toiled. Nevertheless,
this old man's was a patient hammer wielded by a patient arm. ...¥n
</p>¥n      </div>¥n  </body>¥n</html>"
```

# 3.3 GET() 関数による受信（2）

```
# HTMLを取得し、jsonlite::fromJSON() で処理
url <- "https://httpbin.org/json"
res <- GET(url)
content(res)

## $slideshow

## $slideshow$author

## [1] "Yours Truly"

...

## [1] "Overview"

##

## $slideshow$slides[[2]]$type

## [1] "all"

## $slideshow$title

## [1] "Sample Slide Show"
```

14

# 3.3 GET() 関数による受信（2）

```
url <- "https://httpbin.org/get"
query_res <- GET(url, query = list(key1 = "test", key2 = "テスト"))
content(query_res)

## $args

## $args$key1

## [1] "test"

##

## $args$key2

## [1] "テスト"

...

##

## $url

## [1] "https://httpbin.org/get?key1=test&key2=テスト"
```

15

## 3.4 POST() 関数による送信

- APIを使ってデータの登録、更新などを行う場合、POSTメソッドを使う
- POSTするデータを POST() 関数の body オプションで指定する
- Webページのフォームと同様、ファイルの添付なども可能

```
url <- "https://httpbin.org/post"
res <- POST(url, body = "これはテストです")
res

...

##   "data": "¥u3053¥u308c¥u306f¥u30c6¥u30b9¥u30c8¥u3067¥u3059",

## ...

content(res)$data

## [1] "これはテストです"
```

16

POSTメソッドを使ってデータを送信する（その結果を受け取る）場合、POST() 関数を使います。Webサイトのフォームと同様、送信するデータをボディとして指定します。フォームの各要素（<input type="..." name="...">）と対応する値をセットで指定する場合は、list形式で与えます。また、データをJSON形式で送信したい場合は、encode = "json" とオプションを指定します。

また、これもフォームと同様、ファイルを「添付」することができます。これには、body オプションに upload_file() 関数を指定します。

- 参考: POST file to a server. — POST • httr https://httr.r-lib.org/reference/POST.html

# 3.4 POST() 関数による送信

```
body <- list(a = 1, b = 2, c = 3)
res <- POST(url, body = body)
res

...
##   "form": {
##     "a": "1",
##     "b": "2",
##     "c": "3"
##   },
...
content(res)$form
## $a
## [1] "1"
## $b
## [1] "2"
## $c
## [1] "3"
```

17

# 3.4 POST() 関数による送信

```
body <- list(a = 1, b = 2, c = 3)
res <- POST(url, body = body, encode = "json")
res

...
##   "data": "{¥"a¥":1,¥"b¥":2,¥"c¥":3}",
##   "files": {},
##   "form": {},
##   "headers": {
##     "Accept": "application/json, text/xml, application/xml, */*",
##     "Accept-Encoding": "deflate, gzip",
##     "Content-Length": "19",
##     "Content-Type": "application/json",
## ...
content(res)$data
## [1] "{¥"a¥":1,¥"b¥":2,¥"c¥":3}"
```

# 3.4 POST() 関数による送信

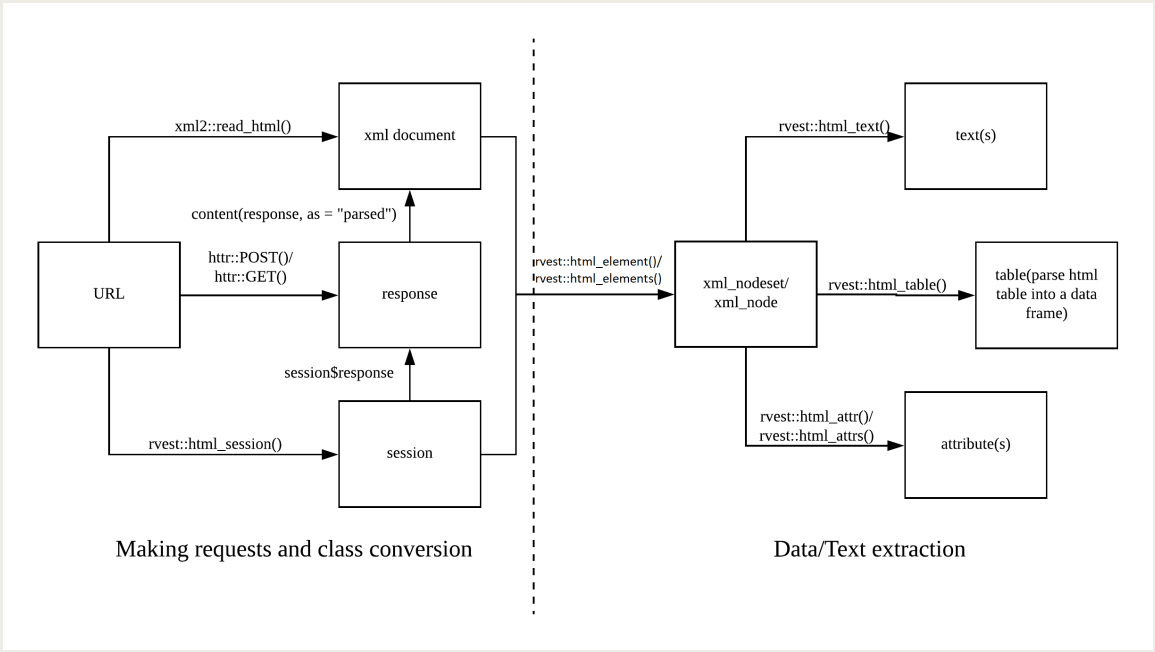
```
res <- POST(url, body = upload_file("test.txt"))
content(res)$data

## [1] "これはテストです。"
```

# 4. rvestパッケージによるWebスクレイピング

20

ここからは、APIが提供されていないWebサイトのデータを取得するための、Webスクレイピングについて紹介していきます。

# 4.1　rvestによるWebスクレイピングの全体像




出典: "R Web Scraping Cheat Sheet
https://awesomeopensource.com/project/yusuzech/r-web-scraping-cheat-sheet"

21

RでWebスクレイピングを行う場合、rvestパッケージが最も広く使われています。

- Easily Harvest (Scrape) Web Pages • rvest https://rvest.tidyverse.org/

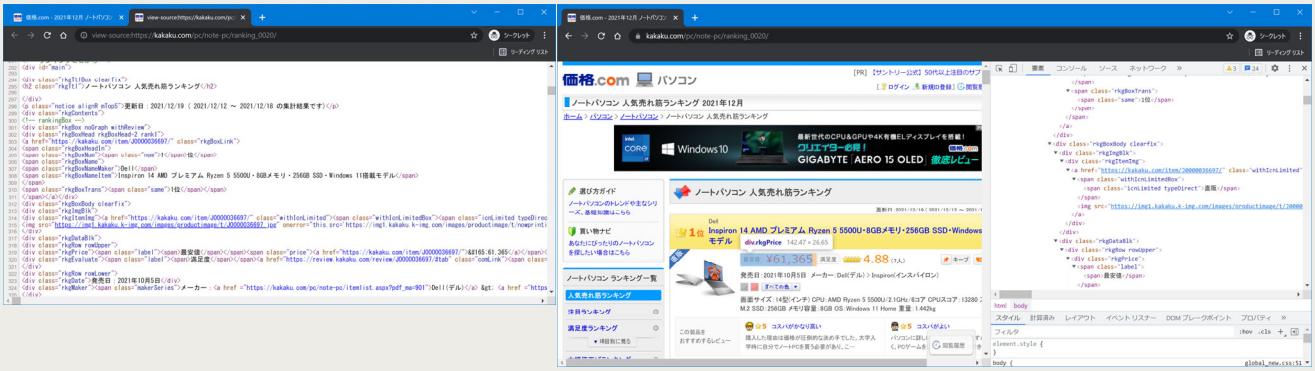
rvestパッケージはtidyverseの一部で、自身でもHTTPアクセスの機能を持っていますが、基本的には、前述のhttrパッケージでHTMLを取得し、それを解析する部分を担います。

httr、rvestパッケージを組み合わせたWebスクレイピングの全体像をスライドでは示しています。流れは、以下の通りです。

1. httrパッケージの `GET` 関数や `POST()` 関数で取得したデータを、（xml2パッケージが提供する）xml_document型に変換する。
2. xml_document型のデータに対して、rvestパッケージの `html_elements()` や `html_element()` 関数を適用し、xml_nodeset / xml_node型のデータに変換する。
3. このオブジェクトに対して、rverstパッケージのさまざまな関数を適用することで、タグに囲まれた文字列やタグの属性、表データなどを抽出できる。

# 4.2　スクレイピングに必要な知識

- まずHTMLを理解している必要がある
- フレーム構造（iframe）やCSS，JavaScriptなど、サイトを構成する要素技術も知っておかないと、欲しい要素を指定できない
- ブラウザの「ソース表示」や「開発者ツール」を活用してサイトの構造を理解する




22

前のページで述べたように、httrパッケージとrvestパッケージを使えば、Webスクレイピングは容易に実行できます。しかし、実際に欲しいデータを取得するためには、Webサイトのどこにデータがあるか、どのようにして「それ」を特定できるかが重要です。

そのため、Webスクレイピングを行うには、まずHTMLやCSS、JavaScriptなどの言語や、HTTPを中心としたWebの仕組みなどを理解しなければなりません。

また、具体的な取得対象の箇所は、CSSセレクターやXPathで指定することになるため、ブラウザの「開発者ツール」などの使い方も知っておく必要があります。

- 参考1: HTML: HyperText Markup Language | MDN https://developer.mozilla.org/ja/docs/Web/HTML
- 参考2: CSS: カスケーディングスタイルシート | MDN https://developer.mozilla.org/ja/docs/Web/CSS
- 参考3: JavaScript | MDN https://developer.mozilla.org/ja/docs/Web/JavaScript
- 参考4: とほほのWWW入門 https://www.tohoho-web.com/www.htm (私が学生の頃からのド定番ですが、常に新しい情報が取り入れられています)
- 参考5: Google Chromeデベロッパーツールの基本的な使い方をわかりやすく解説 https://willcloud.jp/knowhow/dev-tools-01/

## 4.3 read_html() 関数によるWebアクセス

- Webページにアクセスするには read_html() 関数を使う
- 返り値はxml_document型のオブジェクト

```
library(rvest)

html <- read_html("http://rvest.tidyverse.org/")
html

## {html_document}

## <html lang="en">

## [1] <head>¥n<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8 ...

## [2] <body>¥n    <a href="#container" class="visually-hidden-
focusable">Skip t ...
```

```
library(httr)

res <- GET("http://rvest.tidyverse.org/")
class(content(res))

## [1] "xml_document" "xml_node"
```

23

まず、httrパッケージなどで取得した、あるいはローカルに保存したHTMLファイルを読み込むための、read_html() 関数を紹介します。これは、xml2パッケージが提供しています。rvestパッケージを読み込むと、依存関係で自動的に読み込まれます。

- Read HTML or XML. — read_xml • xml2
  https://xml2.r-lib.org/reference/read_xml.html

引数に、URLやローカルのファイルを指定します。帰り値は、xml_document型のオブジェクトです。

httrパッケージの GET() 関数でHTMLを返すURLにアクセスした場合も、content() 関数の返り値は、xml_document型になります。

## 4.4 html_elements() 関数による要素へのアクセス

- 基本的に、スクレイピングで取得したいのは HTMLの一部（表、文字列、数値など）
- 対象が属するタグ・要素を html_elements() 関数などで指定する
- html_element() は最初の要素だけ、html_elements() は条件に一致する要素をすべて返す

```
# pタグをすべて出力
html <- read_html("./simple_example.html")
html %>% html_elements("p")

## {xml_nodeset (4)}

## [1] <p>¥n                これはサンプルテキストです。<a href="https://www.senshu-
u.ac.jp/"  ...

## [2] <p>¥n                これもサンプルテキストです。¥n            </p>

## [3] <p>¥n                表のテストです。出典：<a href="https://jra.jp/news/202212/1208 ...

## [4] <p>ファン投票1位の馬の獲得票数の推移は以下のようになっています。データ出典：<a
href="https://ja.wikipedia.or ...
```

24

xml_document型のオブジェクトから、取得したいデータを抽出していきます。まず、HTMLの中の対象となる要素を取り出します。html_element() 関数とhtml_elements() 関数は、複数形のsがあるかないかの違いで、単数形の html_element() 関数は指定した条件に一致する最初の要素だけ、複数形の html_elements() 関数は一致したすべての要素を取得します。それぞれ、引数にはオブジェクトと、タグやCSSセレクターなどを指定します。

- 参考: Select elements from an HTML document — html_element • rvest
  https://rvest.tidyverse.org/reference/html_element.html

# 4.4 html_elements() 関数による要素へのアクセス

```
# CSSセレクターによる指定
html %>% html_elements("p > a")

## {xml_nodeset (4)}

## [1] <a href="https://www.senshu-u.ac.jp/" style="color: coral;">専修大学
</a>

## [2] <a href="https://www.r-project.org/" target="_blank">The R Project
for St ...

## [3] <a href="https://jra.jp/news/202212/120804.html">有馬記念ファン投票
最終結果発表！  JRA</a>

## [4] <a
href="https://ja.wikipedia.org/wiki/%E6%9C%89%E9%A6%AC%E8%A8%98%E5%BF% ...
```

## 4.4 html_elements() 関数による要素へのアクセス

```
# pタグの最初の1つを出力
html %>% html_element("p")
```

```
## {html_node}
## <p>
## [1] <a href="https://www.senshu-u.ac.jp/" style="color: coral;">専修大学
</a>
## [2] <br>
## [3] <br>
## [4] <a href="https://www.r-project.org/" target="_blank">The R Project
for St ...
```

```
# imgタグを出力（Base64エンコードされた画像）
html %>% html_element("img")
```



26

## 4.5 html_attrs()/html_attr() による属性の取得

- a タグの href 属性（リンク）などを取得したい場合は多い
- html_attrs() は指定したタグのすべての属性を、html_attr("属性") は指定した属性のみを出力する

```
html %>% html_elements("p > a") %>% html_attrs()
## [[1]]
##                         href                        style
## "https://www.senshu-u.ac.jp/"              "color: coral;"
## [[2]]
##                         href                       target
## "https://www.r-project.org/"                    "_blank"
## [[3]]
##                                     href
## "https://jra.jp/news/202212/120804.html"
## [[4]]
##                                     href
## "https://ja.wikipedia.org/wiki/..."
```

27

ファイルを一括してダウンロードするなど、<a> タグに設定されているリンク先URLを取得したいことなどがよくあります。それには、html_attrs() 関数または html_attr() 関数を使います。html_attrs() 関数は、指定したタグのすべての属性を返します。html_attr() 関数は、引数に指定した属性だけを取り出すことができます。

- 参考: Get element attributes — html_attr • rvest
  https://rvest.tidyverse.org/reference/html_attr.html

## 4.5 html_attrs()/html_attr() による属性の取得

```
html %>% html_elements("p > a") %>% html_attr("href")
## [1] "https://www.senshu-u.ac.jp/"
## [2] "https://www.r-project.org/"
## [3] "https://jra.jp/news/202212/120804.html"
## [4]
"https://ja.wikipedia.org/wiki/%E6%9C%89%E9%A6%AC%E8%A8%98%E5%BF%B5#%E6%AD%
B4%E4%BB%A3%E3%83%95%E3%82%A1%E3%83%B3%E6%8A%95%E7%A5%A81%E4%BD%8D%E9%A6%AC
"
```



28

## 4.6 html_text() 関数によるテキストの取得

- タグに囲まれたテキストを取得するには、html_text() / html_text2() 関数を使う
- 両者の違いは、HTMLをそのまま返す（html_text()）か、多少解釈して返す（html_text2()）か

```
html %>% html_elements("p") %>% html_text()
## [1] "¥n                これはサンプルテキストです。専修大学へのリンクです。The R Project for Statistical ComputingからRをダウンロードできます。¥n"

...
html %>% html_elements("p") %>% html_text2()
## [1] "これはサンプルテキストです。専修大学へのリンクです。¥n¥nThe R Project for Statistical ComputingからRをダウンロードできます。"

...
## [4] "ファン投票1位の馬の獲得票数の推移は以下のようになっています。データ出典: 有馬記念 - Wikipedia"
```

29

タグに囲まれたテキストを取得するには、html_text() または html_text2() 関数を使います。2つの関数の違いは、html_text() 関数がそのまま文字列を出力するのに対し、html_text2() 関数は、<br /> タグによる改行を改行コード ¥n に置き換えて出力する、  によるスペースを半角スペースとして出力するなど、「ブラウザでどう表示されているか」を意識しているところです。一方、HTMLソースにおける改行は取り除かれます。

## 4.7 html_table() 関数による表構造データの取得

- 欲しい数値は表（table）の中にあることも多い
- html_table() 関数は、tableタグをデータフレームとして出力する

```
url <- "https://db.netkeiba.com/?pid=jockey_leading"
html <- read_html(url, encoding = "EUC-JP") # イマドキEUCだった!

# rowspan属性のせいで見出しの読み込みが意図しないものになる
res <- html %>% html_elements("table") %>% html_table()
# 列名を結合する
colnames(res[[1]])[9:18] <- paste(colnames(res[[1]])[9:18], res[[1]][1,
9:18], sep = "_")

# 1行目を削除して、型を自動判定する（readr::type_convert() 関数）
df <- res[[1]][-1, ] %>% type_convert()
df
```

```
##    順位 騎手名      所属         生年月日   `1着` `2着` `3着`  着外 重賞_出走
##   <dbl> <chr>       <chr>        <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>
## 1     1 川田将雅    [西]フリー   1985-10-15   137    95    77   213        61
## 2     2 戸崎圭太    [東]田島俊明 1980-07-08   125   106    74   451        51
## 3     3 横山武史    [東]鈴木伸尋 1998-12-22   119    97    88   418        52
```

30

HTMLの表（<table>）からデータを抽出することもできます。html_table() 関数は、指定した <table> 要素をデータフレームに変換して返します。

# 4.8　参考: Yahoo!ニュースのトピックを取得する

```
topics <- read_html("https://www.yahoo.co.jp/") %>%
    html_element("#tabpanelTopics1 > div > div") %>%
    html_nodes("li") %>%
    html_text2() %>%
    str_replace_all("NEW$", "")
topics
```

```
## [1] "救済新法 効果限定的との見方も"    "半導体 対中規制で米が協力要請"
## [3] "ノーベル平和賞授与式 表情は硬く" "新法成立ゴールじゃない 宗教2世"
## [5] "休眠削除 TW故人アカの行方は"      "宇野昌磨、GPファイナル初優勝"
## [7] "THE W「天才ピアニスト」が優勝"    "鬼滅・刀鍛冶の里編 23年4月放送"
```

# 5. まとめ



32

## 5.1　今日の内容

- APIとスクレイピング
- httrパッケージによるAPIアクセス
- rvestパッケージによるWebスクレイピング

33

## 5.2　次回までの課題

- Posit Cloudプロジェクト内の `12_scraping_exercise.R` について、指示に従ってプログラムを作成してください
- 編集したファイルは、ファイル一覧でチェックを入れ、[more] メニューから [Export] を選択し、[Download] ボタンを押してダウンロードしてください
- ダウンロードしたファイルを、Classroomの課題ページから提出してください
- **提出期限: 2023-12-18 23:59**

34