So, as a little challenge for today, what are the most common animals among Beanie babies? Do I even need much webscraping to find this out?

How does one webscrape these days

I've always liked webscraping, as I think I enjoy getting and transforming data more than I enjoy analyzing it. Compared to my former self,

- I know a bit more about XPath (starting with knowing it exists!) so I don't use regular expression to parse HTML.
- I use polite for polite webscraping!
- I know it's best to spend more time pondering about strategies before hammering requests at a website.

As to rvest recent changes, I had a quick look at the changelog but since I hadn't used it in so long, it's not as if I had any habit to change!

How to harvest Beaniepedia

I've noticed Beaniepedia has a sitemap for all beanies so from that I can extract the URLs to all Beanie pages. That's a necessary step.

Now from there I could either

- Scrape each of these pages, respectfully slowly, and extract the table that includes the beanie's information;
- Use a more frugal strategy by parsing URLs. E.g. from the path of https://beaniepedia.com/beanies/beanie-babies/january-the-birthday-bear-series-2/ I can extract the category of the Beanie (a beanie baby as opposed to, say, an attic treasure) and the animal by splitting `january-the-birthday-bear-series-2` into pieces and see whether one is an animal. How would I recognize animals? By extracting the word coming after "the".

I'll choose the second strategy and leave the first one as an exercise to the reader. 😉

From XML to animal frequencies

Let's get to work! A *sine qua non* condition is obviously the website being ok with our scraping stuff. The polite package would tell us whether the robots.txt file were against our doing this, and I also took time looking whether the website had any warning. I didn't find any so I think we're good to go.

```
session <- polite::bow(
  url = "https://beaniepedia.com/beanies/sitemap.xml",
  user_agent = "Maëlle Salmon https://masalmon.eu"
  )
sitemap <- polite::scrape(session)
sitemap
```

```
#> {xml_document}
#>
#>  [1] \n  https://beaniepedia.com/beanies/\n  2021-01 ...
#>  [2] \n  https://beaniepedia.com/beanies/beanie-babies/jerry-the-mi ...
#>  [3] \n  https://beaniepedia.com/beanies/beanie-babies/snowball-the ...
#>  [4] \n  https://beaniepedia.com/beanies/beanie-babies/jemima-puddl ...
#>  [5] \n  https://beaniepedia.com/beanies/beanie-babies/jeff-gordon- ...
#>  [6] \n  https://beaniepedia.com/beanies/beanie-babies/jeff-burton- ...
#>  [7] \n  https://beaniepedia.com/beanies/beanie-babies/jeepers-the- ...
#>  [8] \n  https://beaniepedia.com/beanies/beanie-babies/jeanette-the ...
#>  [9] \n  https://beaniepedia.com/beanies/beanie-babies/jaz-the-cat/ ...
#> [10] \n  https://beaniepedia.com/beanies/beanie-babies/japan-the-be ...
#> [11] \n  https://beaniepedia.com/beanies/beanie-babies/laughter-the ...
#> [12] \n  https://beaniepedia.com/beanies/beanie-babies/righty-2000- ...
#> [13] \n  https://beaniepedia.com/beanies/beanie-babies/lefty-2004-t ...
#> [14] \n  https://beaniepedia.com/beanies/beanie-babies/lefty-the-do ...
#> [15] \n  https://beaniepedia.com/beanies/beanie-babies/lefty-the-do ...
#> [16] \n  https://beaniepedia.com/beanies/beanie-babies/january-the- ...
#> [17] \n  https://beaniepedia.com/beanies/beanie-babies/january-the- ...
#> [18] \n  https://beaniepedia.com/beanies/beanie-babies/janglemouse- ...
#> [19] \n  https://beaniepedia.com/beanies/attic-treasures/burrows-th ...
#> [20] \n  https://beaniepedia.com/beanies/attic-treasures/klause-the ...
#> ...
```

The `sitemap` object is an XML document. I will extract URLs with the xml2 package.

```
sitemap <- xml2::xml_ns_strip(sitemap)
urls <- xml2::xml_text(xml2::xml_find_all(sitemap, ".//loc"))
head(urls)
```

```
#> [1] "https://beaniepedia.com/beanies/"
#> [2] "https://beaniepedia.com/beanies/beanie-babies/jerry-the-minion-2/"
#> [3] "https://beaniepedia.com/beanies/beanie-babies/snowball-the-snowman/"
#> [4] "https://beaniepedia.com/beanies/beanie-babies/jemima-puddle-duck-the-duck/"
#> [5] "https://beaniepedia.com/beanies/beanie-babies/jeff-gordon-24-the-bear/"
#> [6] "https://beaniepedia.com/beanies/beanie-babies/jeff-burton-no-31-the-bear/"
```

Now I need to parse the URLs. In an URL path like `beanies/beanie-babies/jerry-the-minion-2/` the second part is the category, the third part is the Beanie Baby name. I, as if I were a good collector 🙃, am not interested in Attic treasures, only in Beanie Babies.

```
urls_df <- urltools::url_parse(urls)
urls_df <- dplyr::filter(urls_df, stringr::str_detect(path, "beanie-
babies"))
```

This gives me 632 Beanie babies. Let's parse the last part of their path. An earlier attempt ignored that some Beanie babies don't have any "the" in their names, e.g. the Hello Kitty ones. This is a limitation of my stingy approach. The error messages by dplyr were most helpful! "The error occurred in row 185." is so handy!

```
get_animal <- function(parsed_path) {

  if (all(unlist(parsed_path) != "the")) {
    return(NA)
  }

  animals <- unlist(parsed_path)[which(unlist(parsed_path) == "the") +
1]
  animals[length(animals)] # thanks, "The End the bear"
}

library("magrittr")
animals_df <- urls_df %>%
  dplyr::rowwise() %>%
  dplyr::mutate(parsed_path = stringr::str_split(path, "/", simplify =
TRUE)[1,3]) %>%
  dplyr::mutate(parsed_path = stringr::str_split(parsed_path, "-")) %>%
  dplyr::mutate(animal = get_animal(parsed_path))
```

Now we're getting somewhere!

```
dplyr::count(
  animals_df,
  animal,
  sort = TRUE
)

#> # A tibble: 150 x 2
#> # Rowwise:
#>    animal      n
#>
#>  1 bear      222
#>  2 cat        38
#>  3 dog        32
#>  4 rabbit     23
#>  5 NA         15
#>  6 pig        12
#>  7 unicorn     9
#>  8 polar       7
#>  9 giraffe     6
#> 10 penguin     6
#> # … with 140 more rows
```

Is this result surprising? Probably not! Now, let's have a look at the ones we did not identify.

```
animals_df %>%
  dplyr::filter(is.na(animal)) %>%
  dplyr::pull(path)

#>  [1] "beanies/beanie-babies/hello-kitty-rainbow-with-cupcake/"
#>  [2] "beanies/beanie-babies/boston-red-sox-key-clip/"
#>  [3] "beanies/beanie-babies/i-love-you-bears/"
#>  [4] "beanies/beanie-babies/zodiac-horse/"
#>  [5] "beanies/beanie-babies/hong-kong-toy-fair-2017-brown/"
#>  [6] "beanies/beanie-babies/hello-kitty-bunny-costume/"
#>  [7] "beanies/beanie-babies/hello-kitty-pink-tartan/"
#>  [8] "beanies/beanie-babies/hello-kitty-gold-angel/"
#>  [9] "beanies/beanie-babies/rock-hello-kitty/"
#> [10] "beanies/beanie-babies/hello-kitty-i-love-japan-usa-version/"
#> [11] "beanies/beanie-babies/hello-kitty-i-love-japan-uk-version/"
#> [12] "beanies/beanie-babies/zodiac-ox/"
#> [13] "beanies/beanie-babies/zodiac-tiger/"
#> [14] "beanies/beanie-babies/happy-birthday-sock-monkey/"
#> [15] "beanies/beanie-babies/zodiac-goat/"
```

Fair enough, and nothing endangering our conclusion that bears win.

Conclusion

In this post I set out to find out what animals are the most common among Beanie babies. I thought I'd freshen my rvest-ing skill but thanks to the sitemap, that's my rusty dplyr knowledge I was able to update a bit. In the end, I learnt that 35% of Beanie babies, at least the ones registered on Beaniepedia, are bears. Thanks to Beaniepedia maintainer for allowing this fun!