Scraping online pricing data

August 2024

Prices in html format

1. To start, install the r packages *xml2*, *tidyverse*, *janitor*, *writexl*, *readxl* and *knitr*. Then, run the library. Delete the # to run the installation.

```
#install.packages("xml2")
#install.packages("tidyverse")
#install.packages("janitor")
#install.packages("writexl")
#install.packages("knitr")
#install.packages("readxl")

library(xml2)
library(tidyverse)
library(janitor)
library(writexl)
library(knitr)
library(readxl)
```

2. Create variables for the url and other basic information.

```
webpage_url <- "https://web.archive.org/web/20230322070257/https://www.safaricom.co.ke/personal/m-pesa/r
country <- "Kenya"
provider <- "Safaricom"
mobile_money <- "M-PESA"
access_date <- Sys.Date()
```

3. Load the csv file on the currency exchange rates. This has 3 columns on country, currency and the exchange rate against a dollar.

```
currency_exchange <- read_csv("Currency Exchange Rates (1-1-2022).csv")
head(currency_exchange[,1:3],10)  #Prints the 1st 10 rows to illustrate
```

```
## # A tibble: 10 x 3
##    country       currency                     exchange_rate
##    <chr>         <chr>                                <dbl>
##  1 Bangladesh    Taka                                  84.2
##  2 Colombia      Colombian Peso                      4064.
##  3 Côte d'Ivoire West African CFA Franc (XOF)         577.
##  4 Ethiopia      Birr                                  49.0
##  5 Ghana         New Cedi                               6.11
##  6 India         Indian Rupee                          74.4
```

```
##  7 Indonesia     Rupiah                          14215
##  8 Kenya         Kenyan Shilling                   112.
##  9 Mali          West African CFA Franc (XOF)      577.
## 10 Mexico        Mexican Peso                       20.5
```

4. Read the webpage and standardize the table.

```
#read the webpage
webpage <- xml2::read_html(webpage_url)
feetable <- rvest::html_table(webpage)[[2]] %>%
  tibble::as_tibble(.name_repair = "unique") %>%  # repair the repeated columns
  clean_names()
feetable %>% view()

#rename columns to standardized variable names and clean data
colnames(feetable)<- c("value_min","value_max","on-network p2p transfer_fee", "off-network p2p transfer

# Remove first 2 rows
feetable[-c(1, 2),] ->feetable

# Replace FREE with 0
feetable <- feetable %>%
  mutate(across(everything(), ~ ifelse(. == "FREE", 0, .)))

# Convert all variables to numeric
feetable %>%
  mutate_all(parse_number) %>%
  view() -> feetable

# Pivot longer
feetable <- feetable %>%
  pivot_longer(ends_with("fee"),
               names_to = "transaction_type",
               values_to = "fee") %>%
  select(value_min,value_max, transaction_type, fee) %>%
  mutate(rn=row_number()) %>%
  mutate(transaction_type = gsub("_fee","", transaction_type)) %>%
  select(-rn)

# Add channel
feetable <- feetable |>
  mutate(channel = "agent")

# Create cash-in, which is free according to the website
feetable[nrow(feetable) + 1,] <- list(1, 150000, "cash-in", 0, "agent")

# Create customer_type var and arrange by transaction_type
feetable %>% mutate(customer_type = "registered") %>%
  arrange(transaction_type) -> feetable

# Manual disaggregation of taxes (TCI confirmed listed prices are inclusive of taxes and confirmed exci
feetable %>%
  rename("fee_gross" = "fee") ->feetable
```

```
feetable %>%
  mutate(fee = round(fee_gross/1.12,2)) %>%
  mutate(tax = round(fee_gross - fee,2)) ->feetable

feetable %>%
  select(-c(fee_gross)) ->feetable

# Add additional variables
feetable<- feetable %>%
  mutate(country = country) %>%
  mutate(provider = provider) %>%
  mutate(mobile_money = mobile_money) %>%
  mutate(date_collection = access_date) %>%
  mutate(web_address = webpage_url) %>%
  left_join(currency_exchange, by = "country")

# Convert currency to USD
feetable<- feetable %>%
  mutate_at(c("value_min","value_max","fee","tax"), list(USD = ~round(./exchange_rate, digits =2)))

# Reorder columns
col_order <- c("country", "mobile_money", "provider", "transaction_type", "channel", "customer_type", "
               "tax", "currency", "exchange_rate", "value_min_USD",
               "value_max_USD","fee_USD", "tax_USD", "date_collection","web_address")
feetable<- feetable[,col_order]
```

5. View fee table and save as an excel file.

```
kable(feetable[1:5,1:10]) #prints the first 5 rows and 5 columns
```

country	mobile_money	provider	transaction_type	channel	customer_type	value_min	value_max	fee	tax
Kenya	M-PESA	Safaricom	cash-in	agent	registered	1	150000	0.00	0.00
Kenya	M-PESA	Safaricom	on-network p2p transfer	agent	registered	200	2500	30.36	3.64
Kenya	M-PESA	Safaricom	on-network p2p transfer	agent	registered	2501	5000	59.82	7.18
Kenya	M-PESA	Safaricom	on-network p2p transfer	agent	registered	5001	10000	100.00	12.00
Kenya	M-PESA	Safaricom	on-network p2p transfer	agent	registered	10001	20000	175.89	21.11

```
write_xlsx(feetable, paste(country, provider, Sys.Date(),".xlsx",sep="_"))
```

Prices in non-html formats

For non-html formats of price lists, we manually copy the data into excel first then load it in R to clean and shape using the same procedure above. This happens if the following scenarios occur: a) the provider does not have pricing information in table format and rather under various FAQs, b) the pricing information is through customer care response, and c) pricing information is provided through an image embedded in the webpage.

```
feetable1 <- read_excel("(raw) Uganda_Airtel_2023.07.18 (raw data copied from webpage).xlsx", sheet = 1)
webpage_url<- "https://www.airtel.co.ug/airtelmoney/transaction_fees"
country <- "Uganda"
mobile_money <- "Airtel Money"
provider <- "Airtel"
access_date <- ymd("2023-07-18") #For pdfs (non-html) update to date of access per listed prices tracki
```

Prices in pdf format

For pdf formats, we can use a pdf convertor to excel with API. Each API key gives 50 pages free conversion.

To start, install the r package pdftables and run library. Delete the # to run the code.

```
install.packages("pdftables")
library(pdftables)
```

Make sure to save the pdf in the same working directory. In my example, my pdf file name is "Tigo-Pesa.pdf" and so we enter the file name (case sensitive) into the code. To run the code, you also need an API Key. To get the API, create an account with pdftables using this link: https://pdftables.com/pdf-to-excel-api. You get 50 free pages of pdf conversion with each account ($40 for 1000 after 50 pages). Remove the # to run the code.

```
convert_pdf('Tigo-pesa.pdf', output_file = NULL, format = "xlsx-single", message = TRUE, api_key = "eq15
```

Run this code to see how many pages you have remaining. Remove the # to run the code.

```
get_remaining("eq15t4aqiqg1")
```