The results below are generated from an R script.

~~~
---
title: "Citizenship Laws by Country"
description: "MY FINAL PROJECT"
author: "KRISTIN ABIJAOUDE"
date: "`r Sys.Date()`"
output: distill::distill_article
---

```{r}
library(readr)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(ggraph)
library(igraph)
library(collapsibleTree)
library(treemap)
library(knitr)

knitr::opts_chunk$set(echo = TRUE)
~~~

## Asking the Research Question

Eight billion humans. About 200 countries. Many laws with a million asterisks next to them.

Each country handles citizenship and immigration differently. Some countries permit dual citizenship, wh

A citizenship is a contract between the citizen and the country. The citizen pledges allegiance to the c

Question: Which countries have more lax citizenship laws, and which countries have more restrictive citi

## Open Dataset

The dataset, which is called GLOBALCIT Citizenship Law Dataset, originates from Italy-based European Uni

```{r}
# citizenship laws by country
citizenship <- read.csv("_data/GLOBALCIT Citizenship Law v1 EUI ResData/Data/data_v1.0_country-year.csv'

citizenship
```

```{r}
dim(citizenship)
```

There are three datasets; one dataset about citizenship acquisition, another dataset about loss of citiz

```{r}
colnames(citizenship)
```

~~~
```

To make this report easier for the reader, as well as myself, I will be broadly focusing on whether the

## Tidy and Manipulate Dataset

This is an extremely messy dataset, so with `mutate()` and `case_when()`, I reorganize and recode the va

First, let's recode the acquisition variables.

```{r}
# recode laws on acquisition of citizenship
citizen_tidy <- citizenship %>%
  mutate("acq_descborn" = case_when(A01a_bin == 1 ~ "Yes",   # Person born to a citizen of a country (bi
                                    A01a_bin == 0 ~ "No"),
         "acq_descabroad" = case_when(A01b_bin == 1 ~ "Yes", # Person born to a citizen of a country (bi
                                      A01b_bin == 0 ~ "No"),
         "acq_birthright" = case_when(A02a_bin == 1 ~ "Yes", # Person born in a country regardless of pa
                                      A02a_bin == 0 ~ "No"),
         "acq_parents"  = case_when (A02b_bin == 1 ~ "Yes", # Person born to a parent who was also born
                                    A02b_bin == 0 ~ "No"),
         "acq_found" = case_when(A03a_bin == 1 ~ "Yes", # Child found in a country of unknown parentage
                                 A03a_bin == 0 ~ "No"),
         "acq_parent_est" = case_when(A04_bin == 1 ~ "Yes", # Establishment of parentage
                                      A04_bin == 0 ~ "No"),
         "acq_residency" = case_when(A06_bin == 1 ~ "Yes", # Residence-based acquisition
                                     A06_bin == 0 ~ "No"),
         "acq_renounce" = case_when(A06b_bin == 0 ~ "No", # Person must renounce old citizenship first
                                    A06b_bin == 1 ~ "Yes",
                                    TRUE ~ "No"),
         "acq_lang" = case_when(A06c_bin == 0 ~ "No", # Person must know the language basics
                                A06c_bin == 1 ~ "Yes",
                                TRUE ~ "No"),
         "acq_good_chara" = case_when(A06e_bin == 1 ~ "Yes", # Person must be of good character
                                      A06e_bin == 0 ~ "No",
                                      TRUE ~ "No"),
         "acq_econ" = case_when(A06f_bin == 0 ~ "No", # Person must have sufficient income
                                A06f_bin == 1 ~ "Yes",
                                TRUE ~ "No"),
         "acq_childhood" = case_when(A07_bin == 0 ~ "No", # Person with a certain period of residence or
                                     A07_bin == 1 ~ "Yes"),
         "acq_marriage" = case_when(A08_bin == 0 ~ "No", # Person marries a citizen
                                    A08_bin == 1 ~ "Yes"),
         "acq_transfer" = case_when(A09_bin == 0 ~ "No", # Transfer to a child from a parent
                                    A09_bin == 1 ~ "Yes"),
         "acq_adopt" = case_when(A10_bin == 0 ~ "No", # Person who is adopted by a citizen
                                 A10_bin == 1 ~ "Yes"),
         "acq_relative" = case_when(A11_bin == 1 ~ "Yes", # Person who is another relative of a citizen
                                    A11_bin == 0 ~ "No"),
         "acq_rel_former" = case_when(A12a_bin == 1 ~ "Yes", # Person who is the relative of a former ci
                                      A12a_bin == 0 ~ "No"),
         "acq_rel_dead" = case_when(A12b_bin == 1 ~ "Yes", # Person who is the relative of a deceased ci
                                    A12b_bin == 0 ~ "No"),
         "acq_spouse" = case_when(A13_bin == 1 ~ "Yes", # Person who is the spouse or registered partner
~~~

```
                                  A13_bin == 0 ~ "No"),
         "acq_dep_citizen" = case_when(A14_bin == 0 ~ "No", # Person who is the dependent of the citizen
                                       A14_bin == 1 ~ "Yes"),
         "acq_regain" = case_when(A16_bin == 1 ~ "Yes", # Person who was once a former citizen and regai
                                  A16_bin == 0 ~ "No"),
         "acq_specific" = case_when(A18_bin == 0 ~ "No", # Person who possesses the citizenship of a spe
                                    A18_bin == 1 ~ "Yes"),
         "acq_cxn" = case_when(A19_bin == 0 ~ "No", # Person who has a cultural affinity
                               A19_bin == 1 ~ "Yes"),
         "acq_presume" = case_when(A20_bin == 0 ~ "No", # Person who is a presumed citizen acted in good
                                   A20_bin == 1 ~ "Yes"),
         "acq_longterm" = case_when(A21_bin == 0 ~ "No", # Person who has resided in a country for a ver
                                    A21_bin == 1 ~ "Yes"),
         "acq_refugees" = case_when(A22_bin == 0 ~ "No", # Person who is a recognised refugee
                                    A22_bin == 1 ~ "Yes"),
         "acq_stateless" = case_when(A23_bin == 0 ~ "No", # Person who is stateless or of undetermined c
                                     A23_bin == 1 ~ "Yes"),
         "acq_exceptional" = case_when(A24_bin == 1 ~ "Yes", # Person who has special achievements
                                       A24_bin == 0 ~ "No"),
         "acq_service" = case_when(A25_bin == 1 ~ "Yes", # Person who is in the public service
                                   A25_bin == 0 ~ "No"),
         "acq_invest" = case_when(A26_bin == 0 ~ "No", # Person who invests in the country
                                  A26_bin == 1 ~ "Yes"))

# remove unnecessary columns
citizen_tidy <- citizen_tidy[,-5:-73]

# sanity check point
citizen_tidy
```

Next, time to recode laws dealing with loss of citizenship.

```{r}
# recode laws on loss of citizenship
citizen_tidy <- citizen_tidy %>%
  mutate("loss_volunteer" = case_when(L01_bin == 1 ~ "Yes", # Person who voluntarily renounces the citiz
                                      L01_bin == 0 ~ "No"),
         "loss_abroad" = case_when(L02_bin == 1 ~ "Yes", # Person who resides outside the country of whi
                                   L02_bin == 0 ~ "No"),
         "loss_foreignarmy" = case_when(L03_bin == 1 ~ "Yes", # Person who renders military service to a
                                        L03_bin == 0 ~ "No"),
         "loss_foreignserv"  = case_when (L04_bin == 1 ~ "Yes", # Person who renders other services to a
                                          L04_bin == 0 ~ "No"),
         "loss_newcitizen" = case_when(L05_bin == 1 ~ "Yes", # Person who acquires a foreign citizenship
                                       L05_bin == 0 ~ "No"),
         "loss_mustchoose" = case_when(L06_bin == 1 ~ "Yes", # Non-renunciation of foreign citizenship (
                                       L06_bin == 0 ~ "No"),
         "loss_disloyal" = case_when(L07_bin == 1 ~ "Yes", # Loss of citizenship due to disloyalty or tr
                                     L07_bin == 0 ~ "No"),
         "loss_crime" = case_when(L08_bin == 1 ~ "Yes", # Loss of citizenship due to other criminal offe
                                  L08_bin == 0 ~ "No"),
         "loss_fraud" = case_when(L09_bin == 1 ~ "Yes", # Person who has acquired citizenship by fraud
                                  L09_bin == 0 ~ "No"),
         "loss_birth_acq" = case_when(L10_bin == 1 ~ "Yes", # Person who retains a foreign citizenship c
                                      L10_bin == 0 ~ "No"),
         "loss_byparent" = case_when(L11_bin == 1 ~ "Yes", # Person whose parent loses citizenship of a
                                     L11_bin == 0 ~ "No"),
         "loss_byspouse" = case_when(L12_bin == 1 ~ "Yes", # Person whose partner loses citizenship of a
                                     L12_bin == 0 ~ "No"),
         "loss_parent_annul" = case_when(L13a_bin == 1 ~ "Yes", # Person whose descent from a citizen is
                                         L13a_bin == 0 ~ "No"),
         "loss_adopt_abroad" = case_when(L13b_bin == 1 ~ "Yes", # Loss through adoption or guardianship
                                         L13b_bin == 0 ~ "No"),
         "loss_former_stateless" = case_when(L14_bin == 1 ~ "Yes", # Former stateless person who acquire
                                             L14_bin == 0 ~ "No"))
# remove unnecessary columns
citizen_tidy <- citizen_tidy[,-5:-34]

# sanity check point
citizen_tidy
```
I recoded countries that permit dual citizenship.

```{r}
# does the country allow dual citizenship?
citizen_tidy <- citizen_tidy %>%
  mutate("dual_permit" = case_when(dualcit_comb == 0 ~ "No",
                                   dualcit_comb > 0 ~ "Yes")) %>%
  select(-c(dualcit_comb)) %>%
  relocate("dual_permit", .after = "year") %>%
  relocate("country", .after = "year")

# sanity check point
citizen_tidy
```
# Visualizing the Dataset

After all of the `mutate()` and `case_when()` coding, visualizing the data was the next step. To start t

```{r}
# get sum of each variable
citizen_tidy %>%
  group_by(`dual_permit`) %>%
  summarise(n_dual = sum(!is.na(`country`)))
```
Since I was dealing with 47 variables, I looked for codes to cut down time and space. Insetad of repeati

```{r}
# now let's try to repeat that process across the board
citizen_sum <- citizen_tidy %>%
  select(5:50) %>%
  as.tibble()

# compute unique levels in data frame
lvls <- unique(unlist(citizen_sum))
```

```
# apply the sum per value
citizen_sum <- sapply(citizen_sum,
                      function(x) table(factor(x, levels = lvls,
                                               ordered = TRUE)))

# touch ups and add ons required
citizen_sum <- as.data.frame(citizen_sum)
citizen_sum$Law <- add_column("law")
citizen_sum[1, 47] = "yes"
citizen_sum[2,47] = "no"

# sanity check point
citizen_sum

dim(citizen_sum)
```

After running the above codes, I collapsed the dataset into two rows with 47 variables. Since I was deal

```{r}
# time to graph
# let's not repeat ourselves
for(i in 1:ncol(citizen_sum)) {
  print(ggplot(citizen_sum, aes(y = Law, x = citizen_sum[ , i])) +  # ggplot within for-loop
          geom_bar(stat = "identity", fill = "black")) +
    ylab("Number of Countries") +
    coord_flip()
}
# i can't add the labels for each chart, so i suggest that one follows the dataset from left to right, 
```

Next, I condensed the 47 bar graphs into one stacked on one another. In order to execute this, I need to

```{r}
# pivot time
citizen_pivot <- citizen_sum %>% pivot_longer(cols=c(1:46),
                                              names_to='law_type',
                                              values_to='answers')
# sanity check point
citizen_pivot
```
```{r}
ggplot(citizen_pivot,
       aes(x = law_type,
           y = answers,
           fill = Law)) +
  geom_bar(stat = "identity",
           position = "stack") +
  xlab("Citizenship law") +
  ylab("Number of countries") +
  ggtitle("Citizenship Laws by Country") +
  coord_flip()
```
From my results, virtually all of the countries permit citizenship based on one being born in the countr
```
```

~~~
In regards to loss of citizenship, most countries let former citizen renounce their citizenship voluntar

Unfortunately for stateless people, most countries have very few, if any, pathways for them to gain citi

As I mentioned before, I kept the original dataset so I could work on the exceptions for acquiring or lo

```{r}
# dataset for exceptions
citizen_loss <- citizenship %>%
  transmute("loss_newcitizen" = case_when(L05_bin == 1 ~ "Yes",
                                          L05_bin == 0 ~ "No"),
            "categoryloss1" = case_when(L05_cat == 1 ~ "automatic loss lapse",
                                       L05_cat == 2 ~ "some exceptions lapse",
                                       L05_cat == 3 ~ "lapse applicable only to naturalized citizens",
                                       L05_cat == 4 ~ "automatic loss withdrawal",
                                       L05_cat == 5 ~ "some exceptions withdrawal",
                                       L05_cat == 6 ~ "withdrawal applicable only to naturalized citizen
                                       L05_cat == 0 ~ "no case"),
            "loss_byparent" = case_when(L11_bin == 1 ~ "Yes",
                                        L11_bin == 0 ~ "No"),
            "categoryloss2" = case_when(L11_cat == 0 ~ "no case",
                                       L11_cat == 1 ~ "generally applicable",
                                       L11_cat == 2 ~ "some exceptions",
                                       L11_cat == 3 ~ "lost on particular grounds",
                                       L11_cat == 4 ~ "loss on particular grounds with exceptions"),
            "loss_foundling" = case_when(L14_bin == 1 ~ "Yes",
                                         L14_bin == 0 ~ "No"),
            "categoryloss3" = case_when(L14_cat == 1  ~ "loss applies with citizenship as foundling",
                                        L14_cat == 2 ~ "loss applies with citizenship as stateless",
                                        L14_cat == 3 ~ "generally applicable",
                                        L14_cat == 0 ~ "no case"))
# sanity checkpoint
citizen_loss
~~~
I found this `ggplot` code really helpful: these interactive dendrograms branched off based on exceptio

```{r}
# interactive dendrogram

# loss of citizenship from acquiring another citizenship since the new country restricts dual citizensh
# create dataset
loss_newcitizen <- select(citizen_loss, c(loss_newcitizen, categoryloss1))
loss_newcitizen <- as.data.frame(loss_newcitizen)


collapsibleTree(loss_newcitizen,
                hierarchy = c("loss_newcitizen", "categoryloss1"),
                nodeSize = "leafCount",
                width = 500,
                zoomable = TRUE)
# I'm not sure why the no node has multiple branches when it should have one

# loss of citizenship from parents losing their citizenship
```

```
loss_byparent <- select(citizen_loss, c(loss_byparent, categoryloss2))
loss_byparent <- as.data.frame(loss_byparent)

collapsibleTree(loss_byparent,
                hierarchy = c("loss_byparent", "categoryloss2"),
                nodeSize = "leafCount",
                width = 500,
                zoomable = TRUE)

# when a former stateless person acquires citizenship from another country
loss_founding <- select(citizen_loss, c(loss_foundling, categoryloss3))
loss_founding <- as.data.frame(loss_founding)

collapsibleTree(loss_founding,
                hierarchy = c("loss_foundling", "categoryloss3"),
                nodeSize = "leafCount",
                width = 500,
                zoomable = TRUE)

```

I repeated this process with the laws in regards to acquiring citizenship:

```{r}
citizen_acq <- citizenship %>%
  transmute("acq_borndescent" = case_when(A01a_bin == 1 ~ "Yes",
                                          A01a_bin == 0 ~ "No"),
            "categoryacq1" = case_when(A01a_cat == 1 ~ "generally applicable provision",
                                       A01a_cat == 2 ~ "dual citizenship restrictions",
                                       A01a_cat == 3 ~ "wedlock restriction",
                                       A01a_cat == 4 ~ "only if father is a citizen",
                                       A01a_cat == 5 ~ "only if citizen is part of a particular group",
                                       A01a_cat == 0 ~ "no provision"),
            "acq_bornabroad" = case_when(A01b_bin == 1 ~ "Yes",
                                         A01b_bin == 0 ~ "No"),
            "categoryacq2" = case_when(A01b_cat == 1 ~ "generally applicable provision",
                                       A01b_cat == 2 ~ "dual citizenship restrictions",
                                       A01b_cat == 3 ~ "wedlock restriction",
                                       A01b_cat == 4 ~ "only if father is a citizen",
                                       A01b_cat == 5 ~ "only if citizen is part of a particular group",
                                       A01b_cat == 6 ~ "generational restrictions",
                                       A01b_cat == 0 ~ "no provision"),
            "acq_marriage" = case_when(A08_bin == 1 ~ "Yes",
                                       A08_bin == 0 ~ "No"),
            "categoryacq3" = case_when(A08_cat == 1 ~ "generally applicable provision",
                                       A08_cat == 2 ~ "residence required",
                                       A08_cat == 3 ~ "only for female spouse of male citizen (no reside
                                       A08_cat == 4 ~ "only for female spouse of male citizen (residence
                                       A08_cat == 5 ~ "only for male spouse of female citizen",
                                       A08_cat == 6 ~ "provisions differ by gender",
                                       A08_cat == 7 ~ "only if spouse is a member of a particular group'
                                       A08_cat == 0 ~ "no provision"))
citizen_acq
```

~~~
```

```{r}
# interactive dendrogram
# gain citizenship by being born in the country to a citizen
# create dataset
acq_borndescent <- select(citizen_acq, c(acq_borndescent, categoryacq1))
acq_borndescent <- as.data.frame(acq_borndescent)

collapsibleTree(acq_borndescent,
                hierarchy = c("acq_borndescent", "categoryacq1"),
                nodeSize = "leafCount",
                width = 500,
                zoomable = TRUE)

# gain citizenship by being born abroad to a citizen
acq_bornabroad <- select(citizen_acq, c(acq_bornabroad, categoryacq2))
acq_bornabroad <- as.data.frame(acq_bornabroad)

collapsibleTree(acq_bornabroad,
                hierarchy = c("acq_bornabroad", "categoryacq2"),
                nodeSize = "leafCount",
                width = 500,
                zoomable = TRUE)

# gain citizenship through marriage
acq_marriage <- select(citizen_acq, c(acq_marriage, categoryacq3))
acq_marriage <- as.data.frame(acq_marriage)

collapsibleTree(acq_marriage,
                hierarchy = c("acq_marriage", "categoryacq3"),
                nodeSize = "leafCount",
                width = 500,
                zoomable = TRUE)
```

To answer my research question, I condensed the columns by the frequencies of values `yes` and `no`, anc

```{r}
# what countries have more lax citizenship laws and what countries have more restrictive citizenship la
citizen_yes <- rowSums(citizen_tidy == "Yes") %>%
  as.data.frame()

citizen_no <-rowSums(citizen_tidy == "No") %>%
  as.data.frame()

citizen_yes$country <- add_column(citizen_tidy$country)
citizen_no$country <- add_column(citizen_tidy$country)

# joint together
lax_or_restrict <- merge(citizen_yes,citizen_no,by=c("country"))
lax_or_restrict <- rename(lax_or_restrict, "Lax_Provisions" = "..x")
lax_or_restrict <- rename(lax_or_restrict, "Restrictive_Provisions" = "..y")
~~~

```
# sanity check point
lax_or_restrict

# most lax laws
lax_or_restrict %>%
  arrange(desc(`Lax_Provisions`))

# most restrictive laws
lax_or_restrict %>%
  arrange(desc(`Restrictive_Provisions`))

# tree maps
lax <- lax_or_restrict %>%
  arrange(desc(`Lax_Provisions`))
lax <- lax[1:10,]

treemap(lax,
            index="country",
            vSize="Lax_Provisions",
            type = "index",
            title= "Top 10 Countries of with Lax Citizenship Laws")

restrict <- lax_or_restrict %>%
  arrange(desc(`Restrictive_Provisions`))
restrict <- restrict[1:10,]

treemap(restrict,
            index="country",
            vSize="Restrictive_Provisions",
            type = "index",
            title= "Top 10 Countries of with Restrictive Citizenship Laws")
```
```

The countries with the most lax citizenship laws are Germany, Greece, Austria, the Netherlands, and Finl

From all of these visualizations I coded, one could see stark differences between countries, even those

## Reflection and Conclusion

I had no prior experience with R before starting my masters in data analytics, and after many classes, 

Choosing a topic that interested me made the project more manageable. I was always interested in learnin

I felt like I hit the jackpot when I stumbled upon this dataset on the database archival website Data is

The most challenging part of data analytics as a whole is trying to figure out which codes to use for my

However, I wish I learned to code earlier in my childhood. I would have caught up with the rest of the c

Overall, the world is not our oyster and we must follow the laws wherever we go. It is always ideal to c

## Resources

Cookbook for R, http://www.cookbook-r.com/.
```

```
Grolemund, Hadley Wickham and Garrett. R For Data Science. https://r4ds.had.co.nz/introduction.html.

Hoare, Jake. "How to Aggregate Data in R: R-Bloggers." R, 12 July 2018, https://www.r-bloggers.com/2018/

Holtz, Yan. "Help and Inspiration for R Charts." The R Graph Gallery, https://r-graph-gallery.com/.

"How to Count Number of Times a Character Appears in a Row." Stack Overflow, 23 Nov. 2013, https://stack

VINK, Maarten Peter, et al. "Globalcit Citizenship Law Dataset." Cadmus Home, European University Instit

## Error:  attempt to use zero-length variable name
```

The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Monterey 12.6
##
## Matrix products: default
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.41            treemap_2.4-3         collapsibleTree_0.1.7
##  [4] igraph_1.3.5          ggraph_2.1.0          forcats_0.5.2
##  [7] stringr_1.5.0         purrr_1.0.0           tidyr_1.2.1
## [10] tibble_3.1.8          ggplot2_3.4.0         tidyverse_1.3.2
## [13] dplyr_1.0.10          readr_2.1.3
##
## loaded via a namespace (and not attached):
##  [1] viridis_0.6.2       httr_1.4.4          tidygraph_1.2.2     jsonlite_1.8.4
##  [5] viridisLite_0.4.1   modelr_0.1.10       shiny_1.7.4         assertthat_0.2.1
##  [9] highr_0.10          googlesheets4_1.0.1 cellranger_1.1.0    yaml_2.3.6
## [13] ggrepel_0.9.2       pillar_1.8.1        backports_1.4.1     glue_1.6.2
## [17] digest_0.6.31       promises_1.2.0.1    RColorBrewer_1.1-3  polyclip_1.10-4
## [21] rvest_1.0.3         colorspace_2.0-3    htmltools_0.5.4     httpuv_1.6.7
## [25] pkgconfig_2.0.3     broom_1.0.2         haven_2.5.1         xtable_1.8-4
## [29] scales_1.2.1        tweenr_2.0.2        later_1.3.0         tzdb_0.3.0
## [33] ggforce_0.4.1       timechange_0.1.1    googledrive_2.0.0   generics_0.1.3
## [37] farver_2.1.1        ellipsis_0.3.2      withr_2.5.0         lazyeval_0.2.2
## [41] cli_3.5.0           mime_0.12           magrittr_2.0.3      crayon_1.5.2
## [45] readxl_1.4.1        evaluate_0.19       data.tree_1.0.0     fs_1.5.2
## [49] fansi_1.0.3         MASS_7.3-58.1       xml2_1.3.3          tools_4.2.2
## [53] data.table_1.14.6   hms_1.1.2           gargle_1.2.1        lifecycle_1.0.3
## [57] gridBase_0.4-7      plotly_4.10.1       munsell_0.5.0       reprex_2.0.2
## [61] compiler_4.2.2      rlang_1.0.6         grid_4.2.2          rstudioapi_0.14
## [65] htmlwidgets_1.6.0   rmarkdown_2.19      labeling_0.4.2      gtable_0.3.1
## [69] DBI_1.1.3           graphlayouts_0.8.4  R6_2.5.1            gridExtra_2.3
## [73] lubridate_1.9.0     fastmap_1.1.0       utf8_1.2.2          stringi_1.7.8
## [77] Rcpp_1.0.9          vctrs_0.5.1         dbplyr_2.2.1        tidyselect_1.2.0
## [81] xfun_0.36
```

```
Sys.time()
```

```
## [1] "2022-12-28 20:06:50 EST"
```