Functions + automation

Prof. Maria Tackett

Click for PDF of slides

Edinburgh College of Art Collection

Abstract Art

```
## # A tibble: 180 x 3
##    title                                artist       link
##    <chr>                                <chr>        <chr>
##  1 Untitled (1959)                      William Gear https://collections.ed.ac
##  2 Abstract Brush Drawing (2018)        William Joh… https://collections.ed.ac
##  3 Portrait of H.S. (1973)              William Joh… https://collections.ed.ac
##  4 Red and Black (1976)                 William Joh… https://collections.ed.ac
##  5 Untitled (Landscape) (1943)          William Joh… https://collections.ed.ac
##  6 Black Sitka (1961)                   William Joh… https://collections.ed.ac
##  7 Untitled (yellow triangle) (198… Mohamed Oun… https://collections.ed.ac
##  8 Untitled - Abstract Print of Fo… Rena R. Sim… https://collections.ed.ac
##  9 Untitled - Two Abstract Melting… Graeme Murr… https://collections.ed.ac
## 10 Earth Element (1972)                 William Joh… https://collections.ed.ac
## # … with 170 more rows
```

Click here to see the code used to scrape the data.

Untitled (1963)

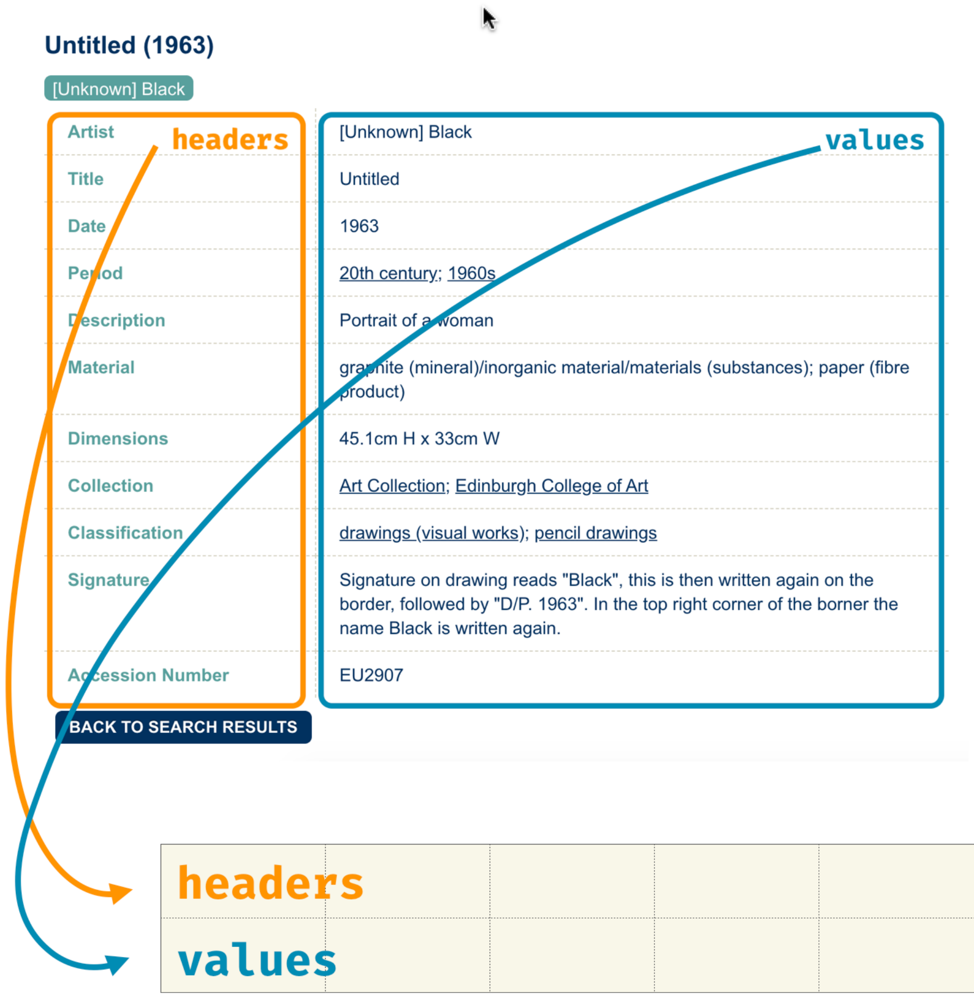
[Unknown] Black

Artist	[Unknown] Black
Title	Untitled
Date	1963
Period	20th century; 1960s
Description	Portrait of a woman
Material	graphite (mineral)/inorganic material/materials (substances); paper (fibre product)
Dimensions	45.1cm H x 33cm W
Collection	Art Collection; Edinburgh College of Art
Classification	drawings (visual works); pencil drawings
Signature	Signature on drawing reads "Black", this is then written again on the border, followed by "D/P. 1963". In the top right corner of the borner the name Black is written again.
Accession Number	EU2907

Untitled (1963)

[Unknown] Black

Artist **headers**	[Unknown] Black **values**
Title	Untitled
Date	1963
Period	20th century; 1960s
Description	Portrait of a woman
Material	graphite (mineral)/inorganic material/materials (substances); paper (fibre product)
Dimensions	45.1cm H x 33cm W
Collection	Art Collection; Edinburgh College of Art
Classification	drawings (visual works); pencil drawings
Signature	Signature on drawing reads "Black", this is then written again on the border, followed by "D/P. 1963". In the top right corner of the borner the name Black is written again.
Accession Number	EU2907



Untitled (1963)

[Unknown] Black

Artist	[Unknown] Black
Title	Untitled
Date	1963
Period	20th century; 1960s
Description	Portrait of a woman
Material	graphite (mineral)/inorganic material/materials (substances); paper (fibre product)
Dimensions	45.1cm H x 33cm W
Collection	Art Collection; Edinburgh College of Art
Classification	drawings (visual works); pencil drawings
Signature	Signature on drawing reads "Black", this is then written again on the border, followed by "D/P. 1963". In the top right corner of the borner the name Black is written again.
Accession Number	EU2907

```r
# load packages ----------------------------------------------------------------
library(tidyverse)
library(rvest)

# first url

## set url
first_info_url <- "https://collections.ed.ac.uk/art/./record/20144?highlight=*:*"

## read page at url
page <- read_html(first_info_url)

## scrape headers
headers <- page %>%
  html_nodes("th") %>%
  html_text()

## scrape values
values <- page %>%
  html_nodes("td") %>%
  html_text() %>%
  str_squish()

## put together in a tibble and add link to help keep track ----
tibble(headers, values) %>%
  pivot_wider(names_from = headers, values_from = values) %>%
  add_column(link = first_info_url)
```

Functions

When should you write a function?

When should you write a function?

Whenever you've copied and pasted a block of code more than twice.

When should you write a function?

Whenever you’ve copied and pasted a block of code more than twice.

How many times will we need to copy and paste the code we developed to scrape additional data on each abstract art piece in the Edinburgh College of Art Collection?

When should you write a function?

Whenever you’ve copied and pasted a block of code more than twice.

How many times will we need to copy and paste the code we developed to scrape additional data on each abstract art piece in the Edinburgh College of Art Collection?

179 more times!

Why functions?

- Automate common tasks in a more powerful and general way than copy-and-pasting:
  - You can give a function an evocative name that makes your code easier to understand.
  - As requirements change, you only need to update code in one place, instead of many.
  - You eliminate the chance of making incidental mistakes when you copy and paste (i.e. updating a variable name in one place, but not in another).

Why functions?

- Automate common tasks in a more powerful and general way than copy-and-pasting:
  - You can give a function an evocative name that makes your code easier to understand.
  - As requirements change, you only need to update code in one place, instead of many.
  - You eliminate the chance of making incidental mistakes when you copy and paste (i.e. updating a variable name in one place, but not in another).
- Down the line: Improve your reach as a data scientist by writing functions (and packages!) that others use

How many inputs in the following code?

```r
## set url ----
first_info_url <- "https://collections.ed.ac.uk/art/./record/20144?highlight=*:*"

## read page at url ----
page <- read_html(first_info_url)

## scrape headers ----
headers <- page %>%
  html_nodes("th") %>%
  html_text()

## scrape values ----
values <- page %>%
  html_nodes("td") %>%
  html_text() %>%
  str_squish()

## put together in a tibble and add link to help keep track ----
tibble(headers, values) %>%
  pivot_wider(names_from = headers, values_from = values) %>%
  add_column(link = first_info_url)
```

How many inputs in the following code?

```r
## set url ----
first_info_url <- "https://collections.ed.ac.uk/art/./record/20144?highlight=*:*"

## read page at url ----
page <- read_html(first_info_url)

## scrape headers ----
headers <- page %>%
  html_nodes("th") %>%
  html_text()

## scrape values ----
values <- page %>%
  html_nodes("td") %>%
  html_text() %>%
  str_squish()

## put together in a tibble and add link to help keep track ----
tibble(headers, values) %>%
  pivot_wider(names_from = headers, values_from = values) %>%
  add_column(link = first_info_url)
```

Turn your code into a function

- Pick a short but informative name, preferably a verb.

```
scrape_art_info <-
```

Turn your code into a function

- Pick a short but evocative name, preferably a verb.
- List inputs, or arguments, to the function inside **`function`**. If we had more arguments the call would look like **`function(x, y, z)`**.

```
scrape_art_info <- function(x){



}
```

Turn your code into a function

- Pick a short but informative name, preferably a verb.
- List inputs, or arguments, to the function inside **`function`**. If we had more the call would look like **`function(x, y, z)`**.
- Place the code you have developed in body of the function, a **`{}`** block that immediately follows **`function(...)`**.

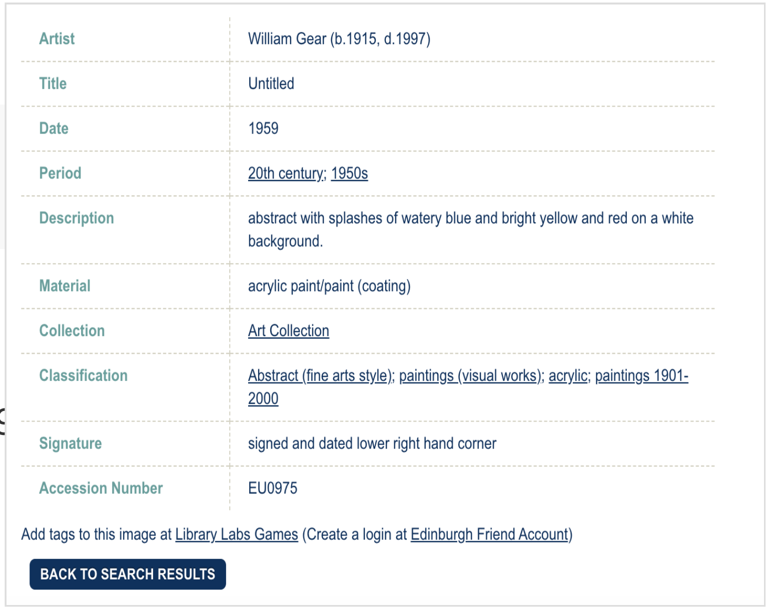
```r
scrape_art_info <- function(x){

  # code we developed earlier to scrape info
  # on single art piece goes here

}
```

```r
scrape_art_info <- function(x){

  # read page at url ----
  page <- read_html(x)

  # scrape headers ----
  headers <- page %>%
    html_nodes("th") %>%
    html_text()

  # scrape values ----
  values <- page %>%
    html_nodes("td") %>%
    html_text() %>%
    str_squish()

  # put together in a tibble and add link to help keep track ----
  tibble(headers, values) %>%
    pivot_wider(names_from = headers, values_from = values) %>%
    add_column(link = x)

}
```

Function in action

```r
scrape_art_info(uoe_art$link[1]) %>%
  glimpse()
```

```
## Rows: 1
## Columns: 11
## $ Artist             <chr> "William Gear (b.1915, d.19
## $ Title              <chr> "Untitled"
## $ Date               <chr> "1959"
## $ Period             <chr> "20th century; 1950s"
## $ Description        <chr> "abstract with splashes of watery blue and bright …
## $ Material           <chr> "acrylic paint/paint (coating)"
## $ Collection         <chr> "Art Collection"
## $ Classification     <chr> "Abstract (fine arts style); paintings (visual wor…
## $ Signature          <chr> "signed and dated lower right hand corner"
## $ `Accession Number` <chr> "EU0975"
## $ link               <chr> "https://collections.ed.ac.uk/art/./record/20144?h…
```



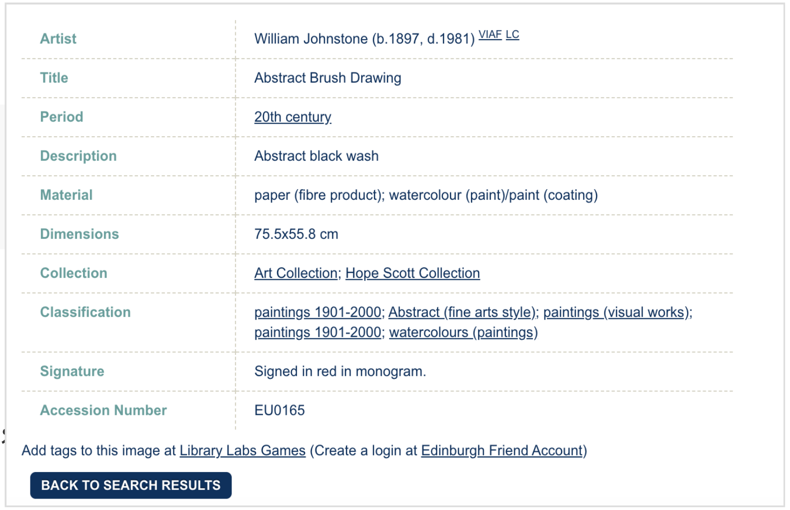
Artist	William Gear (b.1915, d.1997)
Title	Untitled
Date	1959
Period	20th century; 1950s
Description	abstract with splashes of watery blue and bright yellow and red on a white background.
Material	acrylic paint/paint (coating)
Collection	Art Collection
Classification	Abstract (fine arts style); paintings (visual works); acrylic; paintings 1901-2000
Signature	signed and dated lower right hand corner
Accession Number	EU0975

Add tags to this image at Library Labs Games (Create a login at Edinburgh Friend Account)

Function in action

```r
scrape_art_info(uoe_art$link[2]) %>%
  glimpse()
```

```
## Rows: 1
## Columns: 11
## $ Artist             <chr> "William Johnstone (b.1897,
## $ Title              <chr> "Abstract Brush Drawing"
## $ Period             <chr> "20th century"
## $ Description        <chr> "Abstract black wash"
## $ Material           <chr> "paper (fibre product); watercolour (paint)/paint …
## $ Dimensions         <chr> "75.5x55.8 cm"
## $ Collection         <chr> "Art Collection; Hope Scott Collection"
## $ Classification     <chr> "paintings 1901-2000; Abstract (fine arts style); …
## $ Signature          <chr> "Signed in red in monogram."
## $ `Accession Number` <chr> "EU0165"
## $ link               <chr> "https://collections.ed.ac.uk/art/./record/388?hig…
```



Artist	William Johnstone (b.1897, d.1981) VIAF LC
Title	Abstract Brush Drawing
Period	20th century
Description	Abstract black wash
Material	paper (fibre product); watercolour (paint)/paint (coating)
Dimensions	75.5x55.8 cm
Collection	Art Collection; Hope Scott Collection
Classification	paintings 1901-2000; Abstract (fine arts style); paintings (visual works); paintings 1901-2000; watercolours (paintings)
Signature	Signed in red in monogram.
Accession Number	EU0165

Add tags to this image at Library Labs Games (Create a login at Edinburgh Friend Account)

What goes in / what comes out?

- They take input(s) defined in the function definition

```
function([inputs separated by commas]){
  # what to do with those inputs
}
```

- By default they return the last value computed in the function

```
scrape_page <- function(x){
  # do bunch of stuff with the input...

  # return a tibble
  tibble(...)
}
```

- You can define more outputs to be returned in a list as well as nice print methods (but we won't go there for now...)

What is going on here?

```r
add_2 <- function(x){
  x + 2
  1000
}
```

```r
add_2(3)
```

```
## [1] 1000
```

```r
add_2(10)
```

```
## [1] 1000
```

Naming functions

> "There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

Naming functions

> "There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

- Names should be short but clearly evoke what the function does

Naming functions

> "There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

- Names should be short but clearly evoke what the function does
- Names should be verbs, not nouns

Naming functions

> "There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

- Names should be short but clearly evoke what the function does
- Names should be verbs, not nouns
- Multi-word names should be separated by underscores (**`snake_case`** as opposed to **`camelCase`**)

Naming functions

> "There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

- Names should be short but clearly evoke what the function does
- Names should be verbs, not nouns
- Multi-word names should be separated by underscores (**`snake_case`** as opposed to **`camelCase`**)
- A family of functions should be named similarly (**`scrape_page`**, **`scrape_art_info`** OR **`str_squish`**, **`str_trim`**, **`str_remove`** etc.)

Naming functions

> "There are only two hard things in Computer Science: cache invalidation and naming things." - Phil Karlton

- Names should be short but clearly evoke what the function does
- Names should be verbs, not nouns
- Multi-word names should be separated by underscores (**`snake_case`** as opposed to **`camelCase`**)
- A family of functions should be named similarly (**`scrape_page`**, **`scrape_art_info`** OR **`str_squish`**, **`str_trim`**, **`str_remove`** etc.)
- Avoid overwriting existing (especially widely used) functions

```
# JUST DON'T
mean <- function(x){
  x * 3
  }
```

Automation

Define the task

- Goal: Scrape info on all 180 abstract art in the collection
- So far:

```
scrape_art_info(uoe_art$link[1])
scrape_art_info(uoe_art$link[2])
scrape_art_info(uoe_art$link[3])
```

- What else do we need to do?
  - Run the **`scrape_art_info()`** function on all 180 links
  - Combine the resulting data frames from each run into one giant data frame with 180 rows

Inputs

You now have a function that will scrape the relevant info on art pieces given the URL of its individual info page. Where can we get a list of URLs of each of the art pieces in the collection?

Inputs

You now have a function that will scrape the relevant info on art pieces given the URL of its individual info page. Where can we get a list of URLs of each of the art pieces in the collection?

From the original data frame!

```
uoe_art
```

```
## # A tibble: 180 x 3
##    title                              artist       link
##    <chr>                              <chr>        <chr>
##  1 Untitled (1959)                    William Gear https://collections.ed.ac.uk/a…
##  2 Abstract Brush Drawing (2018)      William Joh… https://collections.ed.ac.uk/a…
##  3 Portrait of H.S. (1973)            William Joh… https://collections.ed.ac.uk/a…
##  4 Red and Black (1976)               William Joh… https://collections.ed.ac.uk/a…
```

Automation

How can we tell R to apply the **`scrape_art_info()`** function to each link in **`uoe_art$link`**?

Automation

How can we tell R to apply the **`scrape_art_info()`** function to each link in **`uoe_art$link`**?

- Option 1: Write a for loop, i.e. explicitly tell R to visit a link, apply the function, store the result, then visit the next link, apply the function, append the result to the stored result from the previous link, and so on and so forth.

Automation

How can we tell R to apply the **`scrape_art_info()`** function to each link in **`uoe_art$link`**?

- Option 1: Write a for loop, i.e. explicitly tell R to visit a link, apply the function, store the result, then visit the next link, apply the function, append the result to the stored result from the previous link, and so on and so forth.
- Option 2: Map the function to each element in the list of links, and let R take care of the storing and appending of results.

Automation

How can we tell R to apply the **`scrape_art_info()`** function to each link in **`uoe_art$link`**?

- Option 1: Write a for loop, i.e. explicitly tell R to visit a link, apply the function, store the result, then visit the next link, apply the function, append the result to the stored result from the previous link, and so on and so forth.
- Option 2: Map the function to each element in the list of links, and let R take care of the storing and appending of results.

We'll go with Option 2 for now.

How does mapping work?

Suppose we have exam 1 and exam 2 scores of 4 students stored in a list...

```r
exam_scores <- list(
  exam1 <- c(80, 90, 70, 50),
  exam2 <- c(85, 83, 45, 60)
)
```

How does mapping work?

Suppose we have exam 1 and exam 2 scores of 4 students stored in a list...

```
exam_scores <- list(
  exam1 <- c(80, 90, 70, 50),
  exam2 <- c(85, 83, 45, 60)
)
```

...and we find the mean score in each exam

```
map(exam_scores, mean)
```

```
## [[1]]
## [1] 72.5
##
## [[2]]
## [1] 68.25
```

...and suppose we want the results as a numeric (double) vector

```r
map_dbl(exam_scores, mean)
```

```
## [1] 72.50 68.25
```

...and suppose we want the results as a numeric (double) vector

```
map_dbl(exam_scores, mean)
```

```
## [1] 72.50 68.25
```

...or as a character string

```
map_chr(exam_scores, mean)
```

```
## [1] "72.500000" "68.250000"
```

`map_something`

Functions for looping over an object and returning a value (of a specific type):

- **`map()`** - returns a list
- **`map_lgl()`** - returns a logical vector
- **`map_int()`** - returns an integer vector
- **`map_dbl()`** - returns a double vector
- **`map_chr()`** - returns a character vector
- **`map_df()`** / **`map_dfr()`** - returns a data frame by row binding
- **`map_dfc()`** - returns a data frame by column binding
- ...

Go to each page, scrape art info

- Map the **`scrape_art_info()`** function
- to each element of **`uoe_art$link`**
- and return a data frame by row binding

```r
uoe_art_info <- map_df(uoe_art$link, scrape_art_info)
```

```
## # A tibble: 180 x 14
##    Artist Title Date  Period Description Material Collection Classificatio
##    <chr>  <chr> <chr> <chr>  <chr>       <chr>    <chr>      <chr>
##  1 Willi… Unti… 1959  20th … abstract w… acrylic… Art Colle… Abstract (fin
##  2 Willi… Abst… <NA>  20th … Abstract b… paper (… Art Colle… paintings 190
##  3 Willi… Port… 1973  20th … Charcoal o… charcoa… Art Colle… Abstract (fin
##  4 Willi… Red … 1976  20th … Abstract b… ink/coa… Art Colle… paintings 190
##  5 Willi… Unti… 1943  20th … Abstract b… paper (… Art Colle… paintings 190
##  6 Willi… Blac… 1961  20th … Black land… canvas … Art Colle… oil paintings
##  7 Moham… Unti… 1989  20th … abstract t… acrylic… Art Colle… Abstract (fin
##  8 Rena … Unti… 1982  20th … Print in b… paper (… Art Colle… fine art; Abs
##  9 Graem… Unti… 1985… 20th … Print of a… Print    Art Colle… fine art; pri
## 10 Willi… Eart… 1972  20th … Abstract b… ink/coa… Art Colle… paintings 190
## # … with 170 more rows, and 6 more variables: Signature <chr>, `Accession
## #   Number` <chr>, link <chr>, Dimensions <chr>, Subject <chr>, `Alternati
## #   Title` <chr>
## Rows: 180
## Columns: 14
## $ Artist               <chr> "William Gear (b.1915, d.1997)", "William Johns
## $ Title                <chr> "Untitled", "Abstract Brush Drawing", "Portrai
## $ Date                 <chr> "1959", NA, "1973", "1976", "1943", "1961", "19
## $ Period               <chr> "20th century; 1950s", "20th century", "20th ce
## $ Description          <chr> "abstract with splashes of watery blue and brig
## $ Material             <chr> "acrylic paint/paint (coating)", "paper (fibre
## $ Collection           <chr> "Art Collection", "Art Collection; Hope Scott C
## $ Classification       <chr> "Abstract (fine arts style); paintings (visual
## $ Signature            <chr> "signed and dated lower right hand corner", "St
## $ `Accession Number`   <chr> "EU0975", "EU0165", "EU0138", "EU0147", "EU014
## $ link                 <chr> "https://collections.ed.ac.uk/art/./record/201
## $ Dimensions           <chr> NA, "75.5x55.8 cm", "45.7x40.6 cm", "77.4x58.4
## $ Subject              <chr> NA, NA, NA, NA, NA, NA, "abstract", NA, NA, NA
## $ `Alternative Title`  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
```

What could go wrong?

```r
uoe_art_info <- map_df(uoe_art$link, scrape_art_info)
```

- This will take a while to run
- If you get **`HTTP Error 429 (Too many requests)`** you might want to slow down your hits by modifying your function to slow it down by adding a random wait (sleep) time between hitting each link

```r
scrape_art_info <- function(x){

  # Sleep for randomly generated number of seconds
  # Generated from a uniform distribution between 0 and 1
  Sys.sleep(runif(1))

  # Rest of your function code goes here...
}
```