Mini Project 2: Data Aquisition

Rain & Ziling

Table of contents

library(tidyverse)  
library(stringr)  
library(rvest)  
library(polite)  
library(sf)  
library(maps)  
library(viridis)  
library(leaflet)  
library(htmltools)  
library(janitor)  
library(httr2)  
library(httr)  
library(lubridate)  
library(tidycensus)  
library(purrr)

Github Source code [Link](https://github.com/z1l1ng/SDS264Mini-Project-2)

## Introduction

For our project, we decided to scrape data from the wiki pages of one of our favorite video games, *Stardew Valley*. *Stardew Valley* is a popular indie farming game that allows players to take on the role of a character who inherits a run-down farm from their grandfather. In the game, players can grow crops, raise animals, fish, mine, and engage in social activities with the towns people.

For our project, we were interested in compiling a list of items from the game that can be farmed or collected. The only way to make money from the game is by selling these items, and the price of the item depends on the quality of the item and the profession(s) of the player. Thus, our dataset includes information on the name, category, subcategory, and the different price points of the item depending on item quality (regular, silver, gold, and iridium) and player’s profession.

## Approach

All of our data has been accumulated from the [*Stardew Valley* Wiki](https://stardewvalleywiki.com/Stardew_Valley_Wiki) page. Since each item in the game has a different page and not all of the pages followed a similar structure, we used a combination of harvesting the data in both table form and anywhere on the webpage using rvest with html\_text. In the end, we were able to create a dataset from the more important item categories: crops, fish, animal products, and minerals.

### Crops

Crops was the most difficult item to scrape from the wiki, since not all of the pages are structured the same. However, we tried our best to automate where we could.

We start be getting a list of all the different crops in the game.

#check that we are allowed to scrape the wiki  
robotstxt::paths\_allowed("https://stardewvalleywiki.com/Stardew\_Valley\_Wiki")

[1] TRUE

session <- bow("https://stardewvalleywiki.com/Stardew\_Valley\_Wiki", force = TRUE)

crops <- bow("https://stardewvalleywiki.com/Crops", force = TRUE)  
  
result <- scrape(crops) |>  
 html\_nodes(css = "table") |>  
 html\_table(header = TRUE, fill = TRUE)  
  
seasonal\_crops <- result[[134]][2] #table of the season crops so we can use that list  
  
seasonal\_crops <- seasonal\_crops |>  
 mutate(Crops = strsplit(Crops, " • ", fixed = TRUE)) |>  
 unnest(Crops) |>  
 mutate(Crops = str\_replace\_all(Crops, " ", "\_")) |>  
 distinct(Crops)

Create our helper functions for crops:

# function for getting the price at a given page and css selector  
get\_price <- function(page, css\_selector) {  
 page |>  
 html\_nodes(css\_selector) |>  
 html\_text()  
}  
  
# function for creating a tibble of base prices, no profession, for a given crop page  
crop\_base\_prices <- function(crop, tiller = FALSE) {  
 url <- str\_c("https://stardewvalleywiki.com/", crop)  
 page <- read\_html(url)  
   
 qualities <- c("regular", "silver", "gold", "iridium")  
 prices <- list()  
   
 for (i in seq\_along(qualities)) {  
 if (tiller) {  
 selector <- str\_c("tr:nth-child(10) td+ td tr:nth-child(", i, ") td+ td")  
 } else {  
 selector <- str\_c("tr:nth-child(10) tr td:nth-child(1) tr:nth-child(", i, ") td+ td")  
 }  
 price <- get\_price(page, selector)  
 prices[[qualities[i]]] <- parse\_number(price)  
 }  
   
 tibble(  
 item = crop,  
 regular\_price = prices$regular,  
 silver\_price = prices$silver,  
 gold\_price = prices$gold,  
 iridium\_price = prices$iridium  
 )  
}

Create the tibbles for seasonal crops using the helper functions. Note that items 46 (Tea\_Leaves), 44(Sweet Gem Berry), 43(Qi\_Fruit), 41(Cactus\_Fruit), 36(Grape), 4(Coffee\_Bean) have issues when using the functions, so we will scrape the data manually without the functions.

# list of all our seasonal crops  
seasonal\_crops\_list <- pull(seasonal\_crops) # list of our crops tibble  
  
# List of crops, excluding those with known issues  
valid\_crops\_list <- seasonal\_crops\_list[-c(46, 44, 43, 41, 36, 4)]  
  
# Base prices without profession  
base\_crop\_prices <- valid\_crops\_list |>  
 purrr::map\_dfr(~ crop\_base\_prices(.x)) |>  
 mutate(profession = as.character(NA))  
  
# Prices with Tiller profession  
tiller\_crop\_prices <- valid\_crops\_list |>  
 purrr::map\_dfr(~ crop\_base\_prices(.x, tiller = TRUE)) |>  
 mutate(profession = "tiller")  
  
# Combine base and tiller crop prices  
seasonal\_crop\_prices <- bind\_rows(base\_crop\_prices, tiller\_crop\_prices)  
seasonal\_crop\_prices

# A tibble: 80 × 6  
 item regular\_price silver\_price gold\_price iridium\_price profession  
 <chr> <dbl> <dbl> <dbl> <dbl> <chr>   
 1 Blue\_Jazz 50 62 75 100 <NA>   
 2 Carrot 35 43 52 70 <NA>   
 3 Cauliflower 175 218 262 350 <NA>   
 4 Garlic 60 75 90 120 <NA>   
 5 Green\_Bean 40 50 60 80 <NA>   
 6 Kale 110 137 165 220 <NA>   
 7 Parsnip 35 43 52 70 <NA>   
 8 Potato 80 100 120 160 <NA>   
 9 Rhubarb 220 275 330 440 <NA>   
10 Strawberry 120 150 180 240 <NA>   
# ℹ 70 more rows

Do the same for non seasonal crops:

# Non-seasonal crops list, excluding problematic items  
other\_crops <- c("Apple", "Blackberry", "Pomegranate", "Wild\_Plum", "Apricot",   
 "Cherry", "Spice\_Berry", "Peach", "Orange", "Crystal\_Fruit",   
 "Banana", "Mango", "Fiddlehead\_Fern")[-c(10, 7, 4, 2)]  
  
# Base prices without profession  
base\_other\_crops <- other\_crops |>  
 purrr::map\_dfr(~ crop\_base\_prices(.x)) |>  
 mutate(profession = as.character(NA))  
  
# Prices with Tiller profession  
tiller\_other\_crops <- other\_crops |>  
 purrr::map\_dfr(~ crop\_base\_prices(.x, tiller = TRUE)) |>  
 mutate(profession = "tiller")  
  
# Combine base and tiller prices into one table and arrange by item  
nonseasonal\_crop\_tbl <- bind\_rows(base\_other\_crops, tiller\_other\_crops) |>  
 arrange(item)

Finally, create a function for the weird crops that have missing quality or selector path was different

#function for the crops that do not have different qualities  
crop\_weird\_prices <- function(item, selector){  
 url <- str\_c("https://stardewvalleywiki.com/", item)  
 page <- read\_html(url)  
 regular\_price <- get\_price(page, selector)  
   
 tibble(item = item,  
 regular\_price = parse\_number(regular\_price))  
}  
  
#function for the crops that have different qualities. the Berry is for the fruits that have a weird selector that seems to follow a similar pattern.  
crop\_weird\_prices\_w\_quality <- function(crop, tiller = FALSE, berry = FALSE ){  
 url <- str\_c("https://stardewvalleywiki.com/", crop)  
 page <- read\_html(url)  
   
 qualities <- c("regular", "silver", "gold", "iridium")  
 prices <- list()  
   
 for (i in seq\_along(qualities)) {  
 if (tiller) {  
 selector <- str\_c("tr:nth-child(11) td+ td tr:nth-child(", i, ") td+ td")  
 } else if (berry){  
 selector <- str\_c("tr:nth-child(9) tr:nth-child(", i, ") td+ td")  
 }else {  
 selector <- str\_c("tr:nth-child(11) tr td:nth-child(1) tr:nth-child(", i, ") td+ td")  
 }  
 price <- get\_price(page, selector)  
 prices[[qualities[i]]] <- parse\_number(price)  
 }  
   
 tibble(  
 item = crop,  
 regular\_price = prices$regular,  
 silver\_price = prices$silver,  
 gold\_price = prices$gold,  
 iridium\_price = prices$iridium  
 )  
}

Now we make all of the tibbles for the weird crops.

# Tea Leaves  
base\_tea\_leaves <- crop\_weird\_prices("Tea\_Leaves",  
 "tr:nth-child(10) tr td:nth-child(1) td+ td")  
tiller\_tea\_leaves <- crop\_weird\_prices("Tea\_Leaves",  
 "tr:nth-child(10) td+ td td+ td")  
  
tea\_leaves <-bind\_rows(base\_tea\_leaves, tiller\_tea\_leaves)  
  
# Qi\_Fruit  
base\_qi\_fruit <-crop\_weird\_prices("Qi\_Fruit",  
 "tr:nth-child(9) tr td:nth-child(1) td+ td")  
tiller\_qi\_fruit <-crop\_weird\_prices("Qi\_Fruit",  
 "tr:nth-child(9) td+ td td+ td")  
  
qi\_fruit <-bind\_rows(base\_qi\_fruit, tiller\_qi\_fruit)  
  
# Cactus fruit  
cactus\_fruit <- crop\_weird\_prices\_w\_quality("Cactus\_Fruit")  
cactus\_fruit\_tiller <- crop\_weird\_prices\_w\_quality("Cactus\_Fruit", tiller = TRUE)  
  
cactus\_fruit <-bind\_rows(cactus\_fruit, cactus\_fruit\_tiller)  
  
# Grape  
grape <- crop\_weird\_prices\_w\_quality("Grape")  
grape\_tiller <- crop\_weird\_prices\_w\_quality("Grape", tiller = TRUE)  
  
grape <-bind\_rows(grape, grape\_tiller)  
  
# Coffee\_bean  
coffee\_bean <- crop\_weird\_prices\_w\_quality("Coffee\_Bean")  
  
# Wild\_plum  
wild\_plum <- crop\_weird\_prices\_w\_quality("Wild\_Plum", berry = TRUE)  
  
# Spice\_berry  
spice\_berry <- crop\_weird\_prices\_w\_quality("Spice\_Berry", berry = TRUE)  
  
# Crystal\_Fruit  
crystal\_fruit <- crop\_weird\_prices\_w\_quality("Crystal\_Fruit", berry = TRUE)  
  
# Finally, blackberry is just weird and likes to be different, so we did not use a function for it.   
#Blackberry  
  
# Base  
url <- str\_c("https://stardewvalleywiki.com/", "Blackberry")  
page <- read\_html(url)  
  
qualities <- c("regular", "silver", "gold", "iridium")  
prices <- list()  
  
# Loop to retrieve and parse prices  
for (i in seq\_along(qualities)) {  
 price <- get\_price(page, str\_c("tr:nth-child(9) tr td:nth-child(1) tr:nth-child(", i, ") td+ td"))  
 prices[[qualities[i]]] <- parse\_number(price)  
}  
  
blackberry <- tibble(  
 item = "Blackberry",  
 regular\_price = prices$regular,  
 silver\_price = prices$silver,  
 gold\_price = prices$gold,  
 iridium\_price = prices$iridium  
)

Now, we can combine all of the crop tibbles into one:

# First chunks of crops   
draft\_crops <- bind\_rows(seasonal\_crop\_prices,  
 nonseasonal\_crop\_tbl,  
 tea\_leaves,   
 qi\_fruit,   
 cactus\_fruit,   
 grape,   
 coffee\_bean,   
 wild\_plum,   
 blackberry,   
 spice\_berry,   
 crystal\_fruit) |>  
 arrange(item)

Lastly, we can add in the category variable and the subcategory variable. to makes things easier, we decided the subcategory would be the crop’s season. Then, we write it to a csv in case the website changes or updates.

seasons <- result[[134]] %>%  
 select(Season = 1, Crops = 2) |>  
 mutate(Crops = strsplit(Crops, " • ", fixed = TRUE)) |>  
 unnest(Crops) |>  
 mutate(Crops = str\_replace\_all(Crops, " ", "\_"))

crop\_prices <- draft\_crops |>  
 left\_join(seasons, join\_by(item == Crops))|>  
 mutate(category = "crop",  
 sub\_category = str\_c(Season, " Crop"))|>  
 select(-Season)  
  
write.csv(crop\_prices, "crop\_prices.csv")  
  
head(crop\_prices, n = 10)

# A tibble: 10 × 8  
 item regular\_price silver\_price gold\_price iridium\_price profession category  
 <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>   
 1 Amar… 150 187 225 300 <NA> crop   
 2 Amar… 165 205 247 330 tiller crop   
 3 Anci… 550 687 825 1100 <NA> crop   
 4 Anci… 605 755 907 1210 tiller crop   
 5 Apple 100 125 150 200 <NA> crop   
 6 Apple 110 137 165 220 tiller crop   
 7 Apri… 50 62 75 100 <NA> crop   
 8 Apri… 55 68 82 110 tiller crop   
 9 Arti… 160 200 240 320 <NA> crop   
10 Arti… 176 220 264 352 tiller crop   
# ℹ 1 more variable: sub\_category <chr>

### Fish

Fish was the second most difficult item to scrape from the wiki, since again not all of the pages are structured the same. However, we were able identify 4 different pages in which we could write functions to automate.

We start be getting a list of all the different fish in the game.

# Making sure that this irl is scrapable  
fish <- bow("https://stardewvalleywiki.com/Fish", force = TRUE)   
  
# Scraping table to get a list of all the fish   
result <- scrape(fish) |>  
 html\_nodes(css = "table") |>  
 html\_table(header = TRUE, fill = TRUE)  
  
# The correct table for the list of fish, and only keeping the names of the fish column  
fishes <- result[[225]][2]   
  
# However, it is formatted very poorly so we need to tidy it up   
fishes <- fishes |>  
 mutate(Fish = strsplit(Fish, " • ", fixed = TRUE)) |>  
 unnest(Fish) |>  
 # splitting the string since " • " was used to separate all fish  
 mutate(Fish = str\_replace\_all(Fish, " ", "\_")) |>   
 distinct(Fish) |>  
 # this is a fish that is in the data set twice but with different spacing   
 filter(Fish != "\_Super\_Cucumber")   
  
# This is a tibble with the subcategories of the fish and the fish name for joining later  
subcategory <- result[[225]] |>   
 select(Location = 1, Fish = 2) |>   
 mutate(Fish = strsplit(Fish, " • ", fixed = TRUE)) |>   
 unnest(Fish) |>  
 mutate(Fish = str\_replace\_all(Fish, " ", "\_"))

Create our helper functions for fish:

# function for getting the price at a given page and css selector  
get\_price <- function(page, css\_selector) {  
 page |>  
 html\_nodes(css\_selector) |>  
 html\_text()  
}  
  
# function for creating a tibble of prices for a given fish  
  
# this functions output a tibble of our fish   
# and the 4 different prices of the fish dependent on quality  
  
# fish\_base\_prices takes our fish name,   
# and takes a profession if we specify true or false,   
# as well as the "nthchild\_num" value for where the price is being store on that website  
  
fish\_base\_prices <- function(fish, fisher = FALSE, angler = FALSE, nthchild\_num) {  
 url <- str\_c("https://stardewvalleywiki.com/", fish)  
 page <- read\_html(url)  
   
 qualities <- c("regular", "silver", "gold", "iridium")  
 prices <- list()  
   
 for (i in seq\_along(qualities)) {  
 if (fisher) {  
 selector <- str\_c("tr:nth-child(", nthchild\_num,") tr td:nth-child(2) tr:nth-child(", i, ") td+ td")  
 } else if (angler) {  
 selector <- str\_c("tr:nth-child(", nthchild\_num,") tr td:nth-child(3) tr:nth-child(", i, ") td+ td")  
 }   
 else {  
 selector <- str\_c("tr:nth-child(", nthchild\_num,") tr td:nth-child(1) tr:nth-child(", i, ") td+ td")  
 }  
 price <- get\_price(page, selector)  
 prices[[qualities[i]]] <- parse\_number(price)  
 }  
   
 tibble(  
 item = fish,  
 regular\_price = prices$regular,  
 silver\_price = prices$silver,  
 gold\_price = prices$gold,  
 iridium\_price = prices$iridium  
 )  
}

As well as the function for the fish with a different webpage format.

# this functions output a tibble of our fish,   
# and the 2 different prices of the fish dependent on quality  
  
# fish\_base\_prices takes our fish name,   
# and takes a profession if we specify true or false,   
# as well as the "nthchild\_num" value for where the price is being store on that website  
  
fish\_base\_prices2 <- function(fish, fisher = FALSE, angler = FALSE, nthchild\_num) {  
 url <- str\_c("https://stardewvalleywiki.com/", fish)  
 page <- read\_html(url)  
   
 qualities <- c("regular", "silver", "gold", "iridium")  
 prices <- list()  
   
 for (i in seq\_along(qualities)) {  
 if (fisher) {  
 selector <- str\_c("tr:nth-child(", nthchild\_num,") tr td:nth-child(2) tr:nth-child(", i, ") td+ td")  
 } else if (angler) {  
 selector <- str\_c("tr:nth-child(", nthchild\_num,") tr td:nth-child(3) tr:nth-child(", i, ") td+ td")  
 }   
 else {  
 selector <- str\_c("tr:nth-child(", nthchild\_num,") tr td:nth-child(1) tr:nth-child(", i, ") td+ td")  
 }  
 price <- get\_price(page, selector)  
 prices[[qualities[i]]] <- parse\_number(price)  
 }  
   
 tibble(  
 item = fish,  
 regular\_price = prices$regular,  
 silver\_price = prices$silver,  
 )  
}

Now, we will load in our fishes lists so for the type of webpage format they have and then apply our function to the fishes to find their prices.

fishes\_list <- pull(fishes) # List of our fishes tibble to view, then dividing up the fish by their website format  
  
# Loading in the fish we know that are tr:nth-child(14) in the html (these fishes were found in the Fish QMD when first exploring and getting to know our website)  
fishfor14 <- readRDS("fishfor14.RDS")  
fishfor14

[1] "Mutant\_Carp" "Radioactive\_Carp" "Albacore" "Anchovy"   
 [5] "Eel" "Flounder" "Halibut" "Herring"   
 [9] "Octopus" "Pufferfish" "Red\_Mullet" "Red\_Snapper"   
[13] "Sardine" "Sea\_Cucumber" "Squid" "Super\_Cucumber"   
[17] "Tilapia" "Tuna" "Bream" "Catfish"   
[21] "Chub" "Dorado" "Goby" "Lingcod"   
[25] "Perch" "Pike" "Rainbow\_Trout" "Salmon"   
[29] "Shad" "Smallmouth\_Bass" "Sunfish" "Tiger\_Trout"   
[33] "Walleye" "Bullhead" "Carp" "Largemouth\_Bass"   
[37] "Midnight\_Carp" "Sturgeon" "Woodskip" "Ghostfish"   
[41] "Ice\_Pip" "Stonefish" "Sandfish" "Slimejack"   
[45] "Void\_Salmon" "Blobfish" "Midnight\_Squid" "Spook\_Fish"   
[49] "Blue\_Discus" "Lionfish" "Stingray"

# Loading in the fish we know that are tr:nth-child(15) in the html, same as above  
fishfor15 <- readRDS("fishfor15.RDS")  
fishfor15

[1] "Angler" "Crimsonfish" "Glacierfish"   
 [4] "Glacierfish\_Jr." "Legend" "Legend\_II"   
 [7] "Ms.\_Angler" "Son\_of\_Crimsonfish" "Lava\_Eel"   
[10] "Scorpion\_Carp"

# Loading in the fish we know that are tr:nth-child(10) in the html, same as above  
fishfor10 <- readRDS("fishfor10.RDS")  
fishfor10

[1] "Clam" "Cockle" "Mussel" "Oyster"

# Loading in the fish we know that are tr:nth-child(10) in the html, same as above  
fishleft <- readRDS("fishleft.RDS")  
fishleft

[1] "Crab" "Crayfish" "Lobster" "Periwinkle" "Shrimp"   
[6] "Snail"

# Creating list of tbl's to store prices so that we can bind into one big tibble  
fish\_prices <- vector("list", length = 12)  
  
# Base prices without profession for tr:nth-child(14)  
fish\_prices[[1]] <- fishfor14 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, nthchild\_num = 14)) |>  
 mutate(profession = as.character(NA))  
  
# Prices with Fisher profession  
fish\_prices[[2]] <- fishfor14 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, fisher = TRUE, nthchild\_num = 14)) |>  
 mutate(profession = "fisher")  
  
# Prices with Angler profession  
fish\_prices[[3]] <- fishfor14 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, angler = TRUE, nthchild\_num = 14)) |>  
 mutate(profession = "angler")

# Base prices without profession for tr:nth-child(15)  
fish\_prices[[4]] <- fishfor15 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, nthchild\_num = 15)) |>  
 mutate(profession = as.character(NA))  
  
# Prices with Fisher profession  
fish\_prices[[5]] <- fishfor15 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, fisher = TRUE, nthchild\_num = 15)) |>  
 mutate(profession = "fisher")  
  
# Prices with Angler profession  
fish\_prices[[6]] <- fishfor15 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, angler = TRUE, nthchild\_num = 15)) |>  
 mutate(profession = "angler")

# Base prices without profession for tr:nth-child(10)  
fish\_prices[[7]] <- fishfor10 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, nthchild\_num = 10)) |>  
 mutate(profession = as.character(NA))  
  
# Prices with Fisher profession  
fish\_prices[[8]] <- fishfor10 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, fisher = TRUE, nthchild\_num = 10)) |>  
 mutate(profession = "fisher")  
  
# Prices with Angler profession  
fish\_prices[[9]] <- fishfor10 |>  
 purrr::map\_dfr(~ fish\_base\_prices(.x, angler = TRUE, nthchild\_num = 10)) |>  
 mutate(profession = "angler")

# Base prices without profession for tr:nth-child(10) but only two qualities  
fish\_prices[[10]] <- fishleft |>  
 purrr::map\_dfr(~ fish\_base\_prices2(.x, nthchild\_num = 10)) |>  
 mutate(profession = as.character(NA))  
  
# Prices with Fisher profession  
fish\_prices[[11]] <- fishleft |>  
 purrr::map\_dfr(~ fish\_base\_prices2(.x, fisher = TRUE, nthchild\_num = 10)) |>  
 mutate(profession = "fisher")  
  
# Prices with Angler profession  
fish\_prices[[12]] <- fishleft |>  
 purrr::map\_dfr(~ fish\_base\_prices2(.x, angler = TRUE, nthchild\_num = 10)) |>  
 mutate(profession = "angler")

Finally we will take our fish prices and then create one big tibble.

# first tbl in fish prices assigned to our final tibble   
tidy\_fish\_prices <- fish\_prices[[1]]   
  
# for loop for iterating each tbl in our fish prices list to our final tibble  
for (i in 2:12){  
 tidy\_fish\_prices <- bind\_rows(tidy\_fish\_prices, fish\_prices[[i]])  
}  
  
# viewing and alphabetizing our tidy fish tbl   
# also joining our subcategories and assigning category  
(tidy\_fish\_prices <- tidy\_fish\_prices |>  
 left\_join(subcategory, join\_by(item == Fish)) |>  
 mutate(category = "fish") |>  
 rename(sub\_category = Location) |>  
 arrange(item))

# A tibble: 318 × 8  
 item regular\_price silver\_price gold\_price iridium\_price profession  
 <chr> <dbl> <dbl> <dbl> <dbl> <chr>   
 1 Albacore 75 93 112 150 <NA>   
 2 Albacore 93 116 140 187 fisher   
 3 Albacore 112 139 168 225 angler   
 4 Anchovy 30 37 45 60 <NA>   
 5 Anchovy 37 46 56 75 fisher   
 6 Anchovy 45 55 67 90 angler   
 7 Angler 900 1125 1350 1800 <NA>   
 8 Angler 900 1125 1350 1800 <NA>   
 9 Angler 1125 1406 1687 2250 fisher   
10 Angler 1125 1406 1687 2250 fisher   
# ℹ 308 more rows  
# ℹ 2 more variables: sub\_category <chr>, category <chr>

# writing our tbl as a csv so that we can join with the other items   
write.csv(tidy\_fish\_prices, "fish\_prices.csv")  
head(tidy\_fish\_prices, n = 10)

# A tibble: 10 × 8  
 item regular\_price silver\_price gold\_price iridium\_price profession  
 <chr> <dbl> <dbl> <dbl> <dbl> <chr>   
 1 Albacore 75 93 112 150 <NA>   
 2 Albacore 93 116 140 187 fisher   
 3 Albacore 112 139 168 225 angler   
 4 Anchovy 30 37 45 60 <NA>   
 5 Anchovy 37 46 56 75 fisher   
 6 Anchovy 45 55 67 90 angler   
 7 Angler 900 1125 1350 1800 <NA>   
 8 Angler 900 1125 1350 1800 <NA>   
 9 Angler 1125 1406 1687 2250 fisher   
10 Angler 1125 1406 1687 2250 fisher   
# ℹ 2 more variables: sub\_category <chr>, category <chr>

### Animal Products

Animal products was one of the easier items to scrape since we were able to scrape the data from a table.

#first be polite and check that we can scrape it   
robotstxt::paths\_allowed("https://stardewvalleywiki.com/Animal\_Products\_Profitability")

[1] TRUE

session <- bow("https://stardewvalleywiki.com/Animal\_Products\_Profitability", force = TRUE)  
  
#take the second table, because that is the one we are interested in  
result\_animals <- scrape(session) |>  
 html\_nodes(css = "table") |>   
 html\_table(header = TRUE, fill = TRUE)  
  
sd\_animal\_prices <- result\_animals[[2]]

From here all we have to do is clean up our tibble.

#clean up the sd\_animal\_prices tibble  
tidy\_sd\_animal\_price <- sd\_animal\_prices |>  
 clean\_names()|>  
 select(item,   
 profession,   
 quality,   
 sell\_price)|> #select only the columns we want  
 group\_by(item, profession)|>  
 pivot\_wider(names\_from = quality,   
 values\_from = sell\_price,   
 names\_glue = "{quality}\_price",  
 values\_fn = mean)|>  
 clean\_names()|>  
 mutate(category = "animal product",  
 profession = ifelse(profession == "—", NA, profession))  
  
#write the final version to a csv  
write.csv(tidy\_sd\_animal\_price, "animal\_product\_prices.csv")  
head(tidy\_sd\_animal\_price, n = 10)

# A tibble: 10 × 7  
# Groups: item, profession [10]  
 item profession regular\_price silver\_price gold\_price iridium\_price category  
 <chr> <chr> <dbl> <dbl> <dbl> <dbl> <chr>   
 1 Egg <NA> 50 62 75 100 animal …  
 2 Egg Rancher 60 75 90 120 animal …  
 3 Egg Artisan 50 62 75 100 animal …  
 4 Larg… <NA> 95 118 142 190 animal …  
 5 Larg… Rancher 114 142 171 228 animal …  
 6 Larg… Artisan 95 118 142 190 animal …  
 7 Void… <NA> 65 81 97 130 animal …  
 8 Void… Rancher 78 97 117 156 animal …  
 9 Void… Artisan 65 81 97 130 animal …  
10 Duck… <NA> 95 118 142 190 animal …

### Minerals

Minerals was one of the easier items to scrape since we were able to scrape the data from a table. However assigning the category and subcategories is what made the process a little more tedious.

#first be polite and check that we can scrape it   
robotstxt::paths\_allowed("https://stardewvalleywiki.com/Minerals")

[1] TRUE

session <- bow("https://stardewvalleywiki.com/Minerals", force = TRUE)  
  
result\_minerals <- scrape(session) |>  
 html\_nodes(css = "table") |>   
 html\_table(header = TRUE, fill = TRUE)  
#interested in tables 1-4

#This function takes a scraped minerals table and preps it for joining with other datasets  
tidy\_minerals <- function(data, sub\_cat){  
 data|>  
 clean\_names()|>  
 mutate(item = name,  
 category = "mineral",  
 sub\_category = sub\_cat)|>  
 rename(regular\_sell\_price = sell\_price)|>  
 pivot\_longer(  
 cols = c(gemologist\_sell\_price,  
 regular\_sell\_price),  
 names\_to = "profession",  
 values\_to = "sell\_price"  
 )|>  
 select(item,   
 profession,   
 sell\_price,  
 category,  
 sub\_category)|>  
 mutate(sell\_price = as.numeric(str\_extract(sell\_price, '(?<=data-sort-value=")\\d+')),  
 profession = ifelse(profession == "gemologist\_sell\_price",  
 "gemologist", NA))  
   
   
}  
  
#use function for the 1-3 tables using a for loop  
minerals\_tbl <- vector("list", length = 4)  
mineral\_sub\_cat <- c("foraged mineral",  
 "gem",  
 "geode mineral",  
 "geode")  
for (i in 1:3){  
 minerals\_tbl[[i]] <- tidy\_minerals(result\_minerals[[i]], mineral\_sub\_cat[i])  
   
}  
  
#clean up the variable names so that it is ready for the row bind.  
# make sure the category is all mineral, and the sub\_category is correct  
minerals\_tbl[[4]]<- result\_minerals[[4]]|>  
 clean\_names()|>  
 mutate(item = name,  
 category = "mineral",  
 sub\_category = "geode",  
 sell\_price = as.numeric(str\_extract(sell\_price, '(?<=data-sort-value=")\\d+')),  
 profession = NA)|>  
 select(item, sell\_price, category, sub\_category, profession)  
  
tidy\_sd\_minerals\_price <- bind\_rows(minerals\_tbl)

Write it to a csv in case the website changes or updates.

write.csv(tidy\_sd\_minerals\_price, "minerals\_prices.csv")  
head(tidy\_sd\_minerals\_price, n = 10)

# A tibble: 10 × 5  
 item profession sell\_price category sub\_category   
 <chr> <chr> <dbl> <chr> <chr>   
 1 Quartz gemologist 32 mineral foraged mineral  
 2 Quartz <NA> 25 mineral foraged mineral  
 3 Earth Crystal gemologist 65 mineral foraged mineral  
 4 Earth Crystal <NA> 50 mineral foraged mineral  
 5 Frozen Tear gemologist 97 mineral foraged mineral  
 6 Frozen Tear <NA> 75 mineral foraged mineral  
 7 Fire Quartz gemologist 130 mineral foraged mineral  
 8 Fire Quartz <NA> 100 mineral foraged mineral  
 9 Emerald gemologist 325 mineral gem   
10 Emerald <NA> 250 mineral gem

#### Combined Dataset

Lastly, we merged together all of the data sets for each of the 4 categories: crops, fish, animal products, and minerals. This dataset is now ready to be used for different item comparisons depending item category.

# binding rows for all of different categories   
stardew\_items <- bind\_rows(crop\_prices,   
 tidy\_sd\_animal\_price,   
 tidy\_sd\_minerals\_price,  
 tidy\_fish\_prices)  
  
write.csv(stardew\_items, "stardew\_items.csv")  
  
head(stardew\_items, n = 10)

# A tibble: 10 × 9  
 item regular\_price silver\_price gold\_price iridium\_price profession category  
 <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>   
 1 Amar… 150 187 225 300 <NA> crop   
 2 Amar… 165 205 247 330 tiller crop   
 3 Anci… 550 687 825 1100 <NA> crop   
 4 Anci… 605 755 907 1210 tiller crop   
 5 Apple 100 125 150 200 <NA> crop   
 6 Apple 110 137 165 220 tiller crop   
 7 Apri… 50 62 75 100 <NA> crop   
 8 Apri… 55 68 82 110 tiller crop   
 9 Arti… 160 200 240 320 <NA> crop   
10 Arti… 176 220 264 352 tiller crop   
# ℹ 2 more variables: sub\_category <chr>, sell\_price <dbl>