Week 7 Assignment: Working with JSON, HTML, XML, and Parquet in R

Alina Vikhnevich

2025-03-16

Loading necessary libraries

Before we dive into working with different file formats, we need to make sure we have the right tools for the job. Each of these libraries plays a specific role in handling and processing structured and semi-structured data.

```
library(jsonlite)
library(xml2)
library(XML)
library(tidyverse)
library(arrow)
library(knitr)
library(kableExtra)
```

Introduction

In this assignment, we will work with four different data formats: JSON, HTML, XML, and Parquet to store and manipulate inventory data provided by CUNYMart, a fictional retailer located at 123 Example Street, Anytown, USA. The dataset includes product categories, item names, item IDs, brands, prices, variation IDs, and variation details for a variety of items such as electronics, home appliances, clothing, books, and sports equipment.

Our goal is to prepare the data for analysis by formatting it in each of these four formats and exploring how they differ in terms of structure, efficiency, and usability in R. Specifically, we will:

1. **Create a structured inventory dataframe**, containing all relevant product details.
2. **Convert the inventory into multiple formats**:
    - **JSON:** Frequently used in APIs and web applications due to its lightweight nature.
    - **HTML:** Ideal for displaying inventory data in a structured table format on the web.
    - **XML:** Useful for structured document storage and data interchange in applications.
    - **Parquet:** A high-performance format optimized for data analytics and large-scale processing.
3. **Compare these formats** in terms of: Storage efficiency, Readability, Processing speed, Suitability for different applications.
4. **Discuss which format is the most efficient** for this specific dataset and provide a recommendation for future use.

Throughout this process, we will discuss why certain transformations are necessary, how these formats handle structured data, and which format is best suited for different use cases. By the end of this assignment, we will have a deeper understanding of how to structure data efficiently for different applications.

Creating the inventory dataframe

The CUNYmart has provided a dataset containing a structured inventory of products across multiple categories. This dataset includes electronics, home appliances, clothing, books, and sports equipment. The inventory keeps track of various details for each product, such as:

- **Category**: The broad classification of the product (e.g., Electronics, Clothing).
- **Item Name**: The specific name of the product (e.g., Smartphone, Laptop).
- **Item ID**: A unique identifier assigned to each product.
- **Brand**: The manufacturer or brand associated with the product.
- **Price**: The listed price of the item.
- **Variation ID**: A unique identifier for each variation of the product (e.g., different colors, sizes, or specifications).
- **Variation Details**: Additional details describing each product variation (e.g., "Color: Black, Storage: 64GB" for smartphones or "Material: Graphite, Color: Black" for sports equipment).

To efficiently manage and transform this data in R, we first store it in a dataframe, which allows us to manipulate and structure the information effectively.

```
inventory <- data.frame(
  Category = c("Electronics", "Electronics", "Electronics", "Electronics", "Home Appliances", "Home Appl

  Item_Name = c("Smartphone", "Smartphone", "Laptop", "Laptop", "Refrigerator", "Refrigerator", "Washing

  Item_ID = c(101, 101, 102, 102, 201, 201, 202, 202,
              301, 301, 301, 302, 302,
              401, 401, 402, 402,
              501, 502, 502),
  Brand = c("TechBrand", "TechBrand", "CompuBrand", "CompuBrand", "HomeCool", "HomeCool", "CleanTech",

  Price = c(699.99, 699.99, 1099.99, 1099.99,
            899.99, 899.99, 499.99, 499.99,
            19.99, 19.99, 19.99, 49.99, 49.99,
            14.99, 14.99, 24.99, 24.99,
            29.99, 89.99, 89.99),
  Variation_ID = c("101-A", "101-B", "102-A", "102-B",
                   "201-A", "201-B", "202-A", "202-B",
                   "301-A", "301-B", "301-C", "302-A",
                   "302-B", "401-A", "401-B", "402-A",
                   "402-B", "501-A", "502-A", "502-B"),

  Variation_Details = c("Color: Black, Storage: 64GB", "Color: White, Storage: 128GB", "Color: Silver, S
"Format: Hardcover, Language: English", "Format: Paperback, Language: Spanish", "Format: eBook, Languag
)

# Display the first few rows
head(inventory)
```

```
##          Category    Item_Name Item_ID      Brand   Price Variation_ID
## 1     Electronics   Smartphone     101  TechBrand  699.99        101-A
## 2     Electronics   Smartphone     101  TechBrand  699.99        101-B
## 3     Electronics       Laptop     102 CompuBrand 1099.99        102-A
## 4     Electronics       Laptop     102 CompuBrand 1099.99        102-B
```

```
## 5 Home Appliances Refrigerator     201   HomeCool  899.99        201-A
## 6 Home Appliances Refrigerator     201   HomeCool  899.99        201-B
##                             Variation_Details
## 1                Color: Black, Storage: 64GB
## 2               Color: White, Storage: 128GB
## 3              Color: Silver, Storage: 256GB
## 4          Color: Space Gray, Storage: 512GB
## 5 Color: Stainless Steel, Capacity: 20 cu ft
## 6           Color: White, Capacity: 18 cu ft
```

Now that the inventory data is structured properly in a dataframe, the next step is to store and share it in different formats.

Convert dataframe to JSON

Now that the inventory data is structured in an R dataframe, we need to convert it into JSON format. JSON (JavaScript Object Notation) is widely used for storing and exchanging data, particularly in web applications and APIs. It is flexible and can represent nested data, making it ideal for handling product variations.

```
#Transforms the dataframe into JSON, using pretty = TRUE for readability.
json_data <- toJSON(inventory, pretty = TRUE)

# Save to file
write(json_data, "inventory.json")

# Load JSON back into R
inventory_json <- fromJSON("inventory.json")
head(inventory_json)
```

```
##          Category    Item_Name Item_ID      Brand   Price Variation_ID
## 1     Electronics   Smartphone     101  TechBrand  699.99        101-A
## 2     Electronics   Smartphone     101  TechBrand  699.99        101-B
## 3     Electronics       Laptop     102 CompuBrand 1099.99        102-A
## 4     Electronics       Laptop     102 CompuBrand 1099.99        102-B
## 5 Home Appliances Refrigerator     201   HomeCool  899.99        201-A
## 6 Home Appliances Refrigerator     201   HomeCool  899.99        201-B
##                             Variation_Details
## 1                Color: Black, Storage: 64GB
## 2               Color: White, Storage: 128GB
## 3              Color: Silver, Storage: 256GB
## 4          Color: Space Gray, Storage: 512GB
## 5 Color: Stainless Steel, Capacity: 20 cu ft
## 6           Color: White, Capacity: 18 cu ft
```

This transformation ensures that the inventory data is stored in a structured and lightweight format, making it easily shareable across different systems, including web applications and APIs. JSON's hierarchical structure also makes it efficient for handling complex product attributes and variations.

Next, we'll explore converting the inventory into HTML format, allowing us to present the data in a readable table format suitable for web-based applications.

Convert dataframe to HTML

Another common way to present structured data, especially for web-based applications, is through HTML tables. Converting our inventory dataframe to HTML allows us to create a formatted, human-readable table that can be embedded into webpages or shared with stakeholders without requiring additional tools to interpret the data.

To achieve this, we use the `kable` function from the knitr package, which formats the dataframe as an HTML table. The generated table is then saved as an HTML file, making it accessible through a web browser.

```
html_data <- kable(inventory, format = "html")

# Save as HTML file
writeLines(html_data, "inventory.html")

# Read HTML back
read_html("inventory.html")
```

```
## {html_document}
## <html>
## [1] <body><table>\n<thead><tr>\n<th style="text-align:left;"> Category </th>\ ...
```

This approach makes the inventory visually accessible and structured in a way that is easy to interpret. Unlike JSON, which is designed for machine readability, HTML is built for human readability, making it useful for reports, dashboards, and online catalogs.

Now, let's explore another widely used structured format XML. XML is similar to JSON but focuses more on defining and enforcing hierarchical relationships within data, making it a common choice for data exchange in enterprise applications.

Convert dataframe to XML

While HTML is designed for displaying data and JSON is optimized for lightweight data exchange, XML (Extensible Markup Language) provides a structured way to store hierarchical data. XML is widely used in industries such as finance, healthcare, and government for data storage and communication between systems.

In this step, we convert the inventory dataframe into an XML document, maintaining the structure where each item is represented as a node with relevant attributes like category, item name, price, and brand. We achieve this by:

1. Creating an XML document (`doc`).
2. Defining a root node (`Inventory`).
3. Iterating through the dataframe and adding each row as an `Item` node with child elements for each column.
4. Saving the XML structure to a file (`inventory.xml`).
5. Reading the file back into R to ensure the structure is preserved.

```r
doc <- newXMLDoc()
root <- newXMLNode("Inventory", doc = doc)

apply(inventory, 1, function(row) {
  item <- newXMLNode("Item", parent = root)
  newXMLNode("Category", row["Category"], parent = item)
  newXMLNode("Item_Name", row["Item_Name"], parent = item)
  newXMLNode("Item_ID", row["Item_ID"], parent = item)
  newXMLNode("Brand", row["Brand"], parent = item)
  newXMLNode("Price", row["Price"], parent = item)
  newXMLNode("Variation_ID", row["Variation_ID"], parent = item)
  newXMLNode("Variation_Details", row["Variation_Details"], parent = item)
})
```

```
## [[1]]
## <Variation_Details>Color: Black, Storage: 64GB</Variation_Details>
##
## [[2]]
## <Variation_Details>Color: White, Storage: 128GB</Variation_Details>
##
## [[3]]
## <Variation_Details>Color: Silver, Storage: 256GB</Variation_Details>
##
## [[4]]
## <Variation_Details>Color: Space Gray, Storage: 512GB</Variation_Details>
##
## [[5]]
## <Variation_Details>Color: Stainless Steel, Capacity: 20 cu ft</Variation_Details>
##
## [[6]]
## <Variation_Details>Color: White, Capacity: 18 cu ft</Variation_Details>
##
## [[7]]
## <Variation_Details>Type: Front Load, Capacity: 4.5 cu ft</Variation_Details>
##
## [[8]]
## <Variation_Details>Type: Top Load, Capacity: 5.0 cu ft</Variation_Details>
##
## [[9]]
## <Variation_Details>Color: Blue, Size: S</Variation_Details>
##
## [[10]]
## <Variation_Details>Color: Red, Size: M</Variation_Details>
##
## [[11]]
## <Variation_Details>Color: Green, Size: L</Variation_Details>
##
## [[12]]
## <Variation_Details>Color: Dark Blue, Size: 32</Variation_Details>
##
## [[13]]
## <Variation_Details>Color: Light Blue, Size: 34</Variation_Details>
##
```

```
## [[14]]
## <Variation_Details>Format: Hardcover, Language: English</Variation_Details>
##
## [[15]]
## <Variation_Details>Format: Paperback, Language: Spanish</Variation_Details>
##
## [[16]]
## <Variation_Details>Format: eBook, Language: English</Variation_Details>
##
## [[17]]
## <Variation_Details>Format: Paperback, Language: French</Variation_Details>
##
## [[18]]
## <Variation_Details>Size: Size 7, Color: Orange</Variation_Details>
##
## [[19]]
## <Variation_Details>Material: Graphite, Color: Black</Variation_Details>
##
## [[20]]
## <Variation_Details>Material: Aluminum, Color: Silver</Variation_Details>
```

```
# Save XML to file
saveXML(doc, file = "inventory.xml")
```

```
## [1] "inventory.xml"
```

```
# Read XML back into R
inventory_xml <- xmlParse("inventory.xml")
xmlRoot(inventory_xml)
```

```
## <Inventory>
##   <Item>
##     <Category>Electronics</Category>
##     <Item_Name>Smartphone</Item_Name>
##     <Item_ID>101</Item_ID>
##     <Brand>TechBrand</Brand>
##     <Price> 699.99</Price>
##     <Variation_ID>101-A</Variation_ID>
##     <Variation_Details>Color: Black, Storage: 64GB</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Electronics</Category>
##     <Item_Name>Smartphone</Item_Name>
##     <Item_ID>101</Item_ID>
##     <Brand>TechBrand</Brand>
##     <Price> 699.99</Price>
##     <Variation_ID>101-B</Variation_ID>
##     <Variation_Details>Color: White, Storage: 128GB</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Electronics</Category>
##     <Item_Name>Laptop</Item_Name>
##     <Item_ID>102</Item_ID>
```

```
##     <Brand>CompuBrand</Brand>
##     <Price>1099.99</Price>
##     <Variation_ID>102-A</Variation_ID>
##     <Variation_Details>Color: Silver, Storage: 256GB</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Electronics</Category>
##     <Item_Name>Laptop</Item_Name>
##     <Item_ID>102</Item_ID>
##     <Brand>CompuBrand</Brand>
##     <Price>1099.99</Price>
##     <Variation_ID>102-B</Variation_ID>
##     <Variation_Details>Color: Space Gray, Storage: 512GB</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Home Appliances</Category>
##     <Item_Name>Refrigerator</Item_Name>
##     <Item_ID>201</Item_ID>
##     <Brand>HomeCool</Brand>
##     <Price> 899.99</Price>
##     <Variation_ID>201-A</Variation_ID>
##     <Variation_Details>Color: Stainless Steel, Capacity: 20 cu ft</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Home Appliances</Category>
##     <Item_Name>Refrigerator</Item_Name>
##     <Item_ID>201</Item_ID>
##     <Brand>HomeCool</Brand>
##     <Price> 899.99</Price>
##     <Variation_ID>201-B</Variation_ID>
##     <Variation_Details>Color: White, Capacity: 18 cu ft</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Home Appliances</Category>
##     <Item_Name>Washing Machine</Item_Name>
##     <Item_ID>202</Item_ID>
##     <Brand>CleanTech</Brand>
##     <Price> 499.99</Price>
##     <Variation_ID>202-A</Variation_ID>
##     <Variation_Details>Type: Front Load, Capacity: 4.5 cu ft</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Home Appliances</Category>
##     <Item_Name>Washing Machine</Item_Name>
##     <Item_ID>202</Item_ID>
##     <Brand>CleanTech</Brand>
##     <Price> 499.99</Price>
##     <Variation_ID>202-B</Variation_ID>
##     <Variation_Details>Type: Top Load, Capacity: 5.0 cu ft</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Clothing</Category>
##     <Item_Name>T-Shirt</Item_Name>
##     <Item_ID>301</Item_ID>
```

```
##     <Brand>FashionCo</Brand>
##     <Price>  19.99</Price>
##     <Variation_ID>301-A</Variation_ID>
##     <Variation_Details>Color: Blue, Size: S</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Clothing</Category>
##     <Item_Name>T-Shirt</Item_Name>
##     <Item_ID>301</Item_ID>
##     <Brand>FashionCo</Brand>
##     <Price>  19.99</Price>
##     <Variation_ID>301-B</Variation_ID>
##     <Variation_Details>Color: Red, Size: M</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Clothing</Category>
##     <Item_Name>T-Shirt</Item_Name>
##     <Item_ID>301</Item_ID>
##     <Brand>FashionCo</Brand>
##     <Price>  19.99</Price>
##     <Variation_ID>301-C</Variation_ID>
##     <Variation_Details>Color: Green, Size: L</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Clothing</Category>
##     <Item_Name>Jeans</Item_Name>
##     <Item_ID>302</Item_ID>
##     <Brand>DenimWorks</Brand>
##     <Price>  49.99</Price>
##     <Variation_ID>302-A</Variation_ID>
##     <Variation_Details>Color: Dark Blue, Size: 32</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Clothing</Category>
##     <Item_Name>Jeans</Item_Name>
##     <Item_ID>302</Item_ID>
##     <Brand>DenimWorks</Brand>
##     <Price>  49.99</Price>
##     <Variation_ID>302-B</Variation_ID>
##     <Variation_Details>Color: Light Blue, Size: 34</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Books</Category>
##     <Item_Name>Fiction Novel</Item_Name>
##     <Item_ID>401</Item_ID>
##     <Brand>-</Brand>
##     <Price>  14.99</Price>
##     <Variation_ID>401-A</Variation_ID>
##     <Variation_Details>Format: Hardcover, Language: English</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Books</Category>
##     <Item_Name>Fiction Novel</Item_Name>
##     <Item_ID>401</Item_ID>
```

```
##     <Brand>-</Brand>
##     <Price>  14.99</Price>
##     <Variation_ID>401-B</Variation_ID>
##     <Variation_Details>Format: Paperback, Language: Spanish</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Books</Category>
##     <Item_Name>Non-Fiction Guide</Item_Name>
##     <Item_ID>402</Item_ID>
##     <Brand>-</Brand>
##     <Price>  24.99</Price>
##     <Variation_ID>402-A</Variation_ID>
##     <Variation_Details>Format: eBook, Language: English</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Books</Category>
##     <Item_Name>Non-Fiction Guide</Item_Name>
##     <Item_ID>402</Item_ID>
##     <Brand>-</Brand>
##     <Price>  24.99</Price>
##     <Variation_ID>402-B</Variation_ID>
##     <Variation_Details>Format: Paperback, Language: French</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Sports Equipment</Category>
##     <Item_Name>Basketball</Item_Name>
##     <Item_ID>501</Item_ID>
##     <Brand>SportsGear</Brand>
##     <Price>  29.99</Price>
##     <Variation_ID>501-A</Variation_ID>
##     <Variation_Details>Size: Size 7, Color: Orange</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Sports Equipment</Category>
##     <Item_Name>Tennis Racket</Item_Name>
##     <Item_ID>502</Item_ID>
##     <Brand>RacketPro</Brand>
##     <Price>  89.99</Price>
##     <Variation_ID>502-A</Variation_ID>
##     <Variation_Details>Material: Graphite, Color: Black</Variation_Details>
##   </Item>
##   <Item>
##     <Category>Sports Equipment</Category>
##     <Item_Name>Tennis Racket</Item_Name>
##     <Item_ID>502</Item_ID>
##     <Brand>RacketPro</Brand>
##     <Price>  89.99</Price>
##     <Variation_ID>502-B</Variation_ID>
##     <Variation_Details>Material: Aluminum, Color: Silver</Variation_Details>
##   </Item>
## </Inventory>
```

This structure makes the inventory both human-readable and machine-readable while ensuring data integrity through strict hierarchical relationships. XML files are commonly used in web services, APIs, and data feeds

where structured information needs to be exchanged between different systems.

Although XML is highly structured, it can be verbose and inefficient for large-scale datasets. This is where Parquet, a highly optimized columnar storage format, comes into play. Next, we will convert the inventory data to Parquet, which is particularly useful for big data processing and analytics.

Save dataframe as Parquet

We now turn to Parquet, a highly efficient format designed for big data processing and analytics. Unlike row-based formats like CSV or JSON, Parquet stores data in a columnar format, making it particularly well-suited for:

- **Fast query performance**: Since column-based storage enables selective reading of data, it significantly speeds up query execution in analytics.
- **Efficient compression**: Parquet applies compression at the column level, reducing storage size and optimizing read speeds.
- **Scalability**: It integrates well with distributed computing frameworks like Apache Spark and Hadoop, making it the preferred choice for handling large datasets.

To store our inventory data as a Parquet file, we follow these steps:

1. Write the dataframe to a Parquet file (`inventory.parquet`).
2. Read the Parquet file back into R to verify that the structure is preserved.

```
write_parquet(inventory, "inventory.parquet")

# Read Parquet back into R
inventory_parquet <- read_parquet("inventory.parquet")

head(inventory_parquet)
```

```
## # A tibble: 6 x 7
##   Category        Item_Name   Item_ID Brand Price Variation_ID Variation_Details
##   <chr>           <chr>         <dbl> <chr> <dbl> <chr>        <chr>
## 1 Electronics     Smartphone      101 Tech~  700. 101-A        Color: Black, St~
## 2 Electronics     Smartphone      101 Tech~  700. 101-B        Color: White, St~
## 3 Electronics     Laptop          102 Comp~ 1100. 102-A        Color: Silver, S~
## 4 Electronics     Laptop          102 Comp~ 1100. 102-B        Color: Space Gra~
## 5 Home Appliances Refrigerat~     201 Home~  900. 201-A        Color: Stainless~
## 6 Home Appliances Refrigerat~     201 Home~  900. 201-B        Color: White, Ca~
```

Since Parquet is not human-readable in a text editor, it is mainly used for machine-level processing in data pipelines. This format is widely used in data engineering, cloud storage, and analytics applications where performance and storage optimization are critical.

With our inventory dataset now available in multiple formats (JSON, HTML, XML, and Parquet), we can compare their efficiency, usability, and storage benefits.

Comparison of Data Formats

Each data format has unique characteristics that impact its efficiency, readability, and suitability for different applications. Below is a comparison based on key criteria:

1. Storage Efficiency

(a) **Parquet** is the most efficient format, as it compresses data and optimizes column storage, making it ideal for large datasets.
(b) **JSON and XML** are text-based and tend to be larger in size due to extra metadata and nested structures.
(c) **HTML** is the least efficient for storage, as it includes styling elements and markup that increase file size.

2. Readability

(a) **JSON** is the easiest to read, with key-value pairs that make it intuitive.
(b) **XML** is structured but more verbose due to opening/closing tags.
(c) **HTML** is readable but not designed for data storage.
(d) **Parquet** is not human-readable, as it is a binary format optimized for processing.

3. Processing Speed

(a) **Parquet** is the fastest for large-scale data operations, as it is designed for fast querying and analytics.
(b) **JSON and XML** are slower due to parsing overhead but are still widely used in applications.
(c) **HTML** is the slowest in terms of data extraction since it requires additional parsing steps.

4. Suitability for Different Applications

(a) **JSON** is ideal for web APIs and application data exchange.
(b) **HTML** is useful for displaying data in browsers.
(c) **XML** is commonly used in structured document storage and enterprise applications.
(d) **Parquet** is best for big data storage, analytics, and machine learning workflows.

Recommended Format for This Dataset

For inventory management and analysis, Parquet is the best choice due to its compression, speed, and efficiency. However:

- **JSON** is a strong alternative if integration with web applications and APIs is needed.
- **XML** might be useful for document-based applications.
- **HTML** is useful for visualization but not recommended for storage.

Conclusion

Working with multiple file formats has demonstrated how data storage and representation can vary significantly depending on the intended use. JSON provided a lightweight and intuitive way to structure and exchange data, making it ideal for web-based applications. HTML, while useful for displaying data in a readable format, was more suited for presentation rather than computational analysis. XML offered a well-structured approach but required additional formatting, making it less efficient for large datasets. On the other hand, Parquet proved to be the most efficient for storage and analysis, optimizing performance through compression and columnar storage, though it lacked human readability.

This assignment reinforced the importance of selecting the right format based on specific needs. If the goal is data interoperability, JSON is the best choice. For web display, HTML works well, while XML remains relevant for structured document storage. However, when handling large-scale datasets and analytics, Parquet stands out as the superior option. The ability to transform and manipulate data in different formats is a critical skill, ensuring flexibility in data management across various applications.