Analyzing Sales Patterns in UK E-commerce

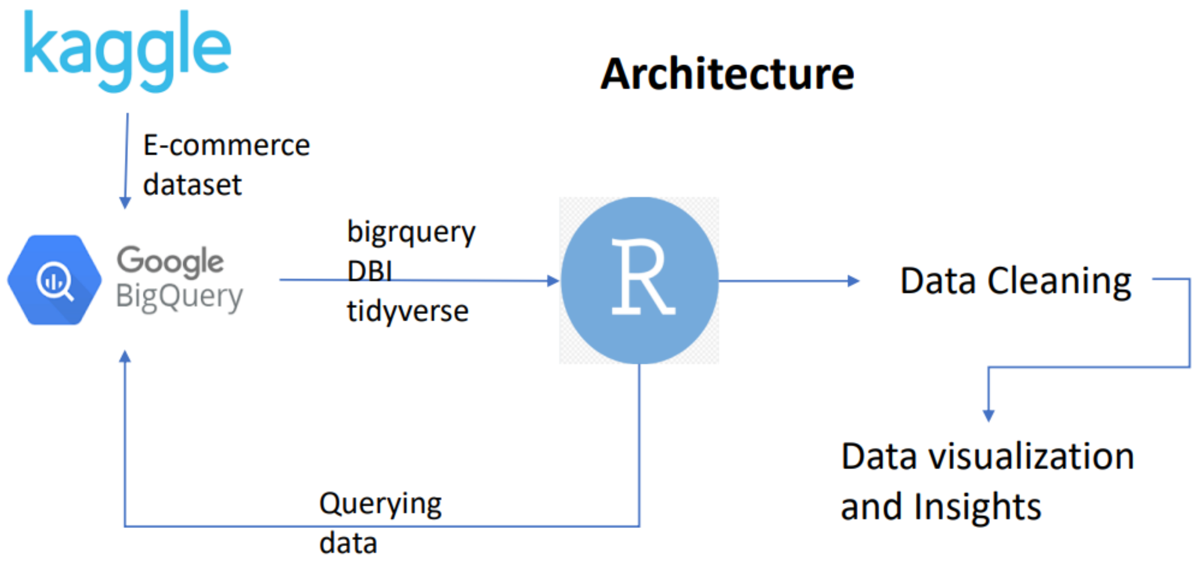
Adithya Praneeth Parupudi

Introduction

This is a transnational ecommerce data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

Architecture

The architecture of this project is structured around the utilization of Google Big Query, a powerful and scalable cloud-based big data analytics platform. The choice to use this service was motivated by its ability to process queries on large datasets rapidly, making it an ideal tool for the volume and complexity of e-commerce transaction data. To interface RStudio with Big Query, the bigrquery library was employed, leveraging its integration for direct data access and manipulation. This setup allows for efficient data handling and complex analyses, making it possible to distill large volumes of transactional data into actionable insights.



About the dataset

The dataset is sourced from "The UCI Machine Learning Repository" which can be found by typing "Online Retail" in the search bar.

This dataset has eight columns. They are:

- InvoiceNo : a 6-digit integral number uniquely assigned to each transaction.
- StockCode : a 5-digit integral number uniquely assigned to each distinct product
- Description: product name
- Quantity : the quantities of each product (item) per transaction
- InvoiceDate: the day and time when each transaction was generated
- UnitPrice : product price per unit (sterling)
- CustomerID : a 5-digit integral number uniquely assigned to each customer
- Country : the name of the country where each customer resides

Research Questions

Here are some research questions I have identified.

- How many distinct countries are there, and how many unique products are sold to each country?
- Is there a seasonal pattern in the sales data, indicating peak shopping periods?
- Which customers contribute the most to revenue, and what are their characteristics?
- What are the shopping trends of some specific keywords?

The Code Part

Loading libraries

bigrquery makes it easier to work with Google's BigQuery for really big data. dbplyr helps translate data operations in R into SQL language. lubridate makes working with dates and times much simpler. scales helps make graphs look better. stringr makes it easy to work with text. purrr helps to do the same task over and over. And ggplot2 helps create detailed graphs.

```r
library(DBI) # Database interface in R
library(bigrquery) # Interface with Google BigQuery
library(tidyverse) # Data manipulation and visualization
library(dbplyr) # Database backend for dplyr
library(lubridate) # Date and time manipulation
library(scales) # Graphical scales mapping data to aesthetics
library(stringr) # String operations
library(purrr) # Functional programming tools
library(ggplot2) # Data visualization
```

Establishing connection

The 'bigrquery::bq_auth()' command, connects rstudio to Big Query. It is authenticated using our google account.After establishing a connection between Google Cloud and Rstudio, the below code declares connection string to uniquely identify the dataset, by their project name and dataset name. These are configured in the Big Query console.

```
# Uncomment this line for the authentication step. It prompts for a google account sig
# bigrquery::bq_auth()



# Establishing connection with big query
con <- dbConnect(bigquery(),
                 project = "ecommerce-data-407200",
                 dataset = "uk_ecommerce_data")
con
```

```
## <BigQueryConnection>
##   Dataset: ecommerce-data-407200.uk_ecommerce_data
##   Billing: ecommerce-data-407200
```

To test the connection, an SQL query is created to pull all the data in the dataset using the tbl() command which takes the connection string and the SQL query as parameters. The datatop 10 rows of the data can be seen using the head() command. We can immediately notice that there are a few empty rows in Description and CustomerID column. They provide a good starting point to identify null values for further analysis.

```
# Checking by running a query
my_db_pointer <-
  tbl(con,
      sql(
        "SELECT * FROM `ecommerce-data-407200.uk_ecommerce_data.ecommerce_data`"
      ))
```

```
## Warning: <BigQueryConnection> uses an old dbplyr interface
## i Please install a newer version of the package or contact the maintainer
## This warning is displayed once every 8 hours.

## ! Using an auto-discovered, cached token.

##   To suppress this message, modify your code or options to clearly consent to

##   the use of a cached token.

##   See gargle's "Non-interactive auth" vignette for more details:

##   <https://gargle.r-lib.org/articles/non-interactive-auth.html>

## i The bigrquery package is using a cached token for 'adityapraneeth@gmail.com'.

## Auto-refreshing stale OAuth token.
```

```
# pulling data to variable.
all_data <- collect(my_db_pointer)


# printing data
head(all_data)
```

```
## # A tibble: 6 x 8
##   InvoiceNo StockCode Description Quantity InvoiceDate         UnitPrice
##   <chr>     <chr>     <chr>          <int> <dttm>                  <dbl>
## 1 536414    22139     <NA>              56 2010-12-01 11:52:00         0
## 2 536545    21134     <NA>               1 2010-12-01 14:32:00         0
## 3 536546    22145     <NA>               1 2010-12-01 14:33:00         0
## 4 536547    37509     <NA>               1 2010-12-01 14:33:00         0
## 5 536549    85226A    <NA>                  1 2010-12-01 14:34:00         0
## 6 536550    85044     <NA>                  1 2010-12-01 14:34:00         0
## # i 2 more variables: CustomerID <int>, Country <chr>
```

```
str(all_data)
```

```
## tibble [541,909 x 8] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo  : chr [1:541909] "536414" "536545" "536546" "536547" ...
##  $ StockCode  : chr [1:541909] "22139" "21134" "22145" "37509" ...
##  $ Description: chr [1:541909] NA NA NA NA ...
##  $ Quantity   : int [1:541909] 56 1 1 1 1 1 1 3 23 -10 ...
##  $ InvoiceDate: POSIXct[1:541909], format: "2010-12-01 11:52:00" "2010-12-01 14:32:00
##  $ UnitPrice  : num [1:541909] 0 0 0 0 0 0 0 0 0 0 ...
##  $ CustomerID : int [1:541909] NA NA NA NA NA NA NA NA NA NA ...
##  $ Country    : chr [1:541909] "United Kingdom" "United Kingdom" "United Kingdom" "Un
```

Data Cleaning

The code uses summarise_all with is.na() to calculate the total number of NA (null) values in each column of the all_data dataframe. It provides an overview of how many null values are present. For the Description column, if a value is null or an empty string, it's replaced with the string "NA". For the CustomerID column, null values are replaced with 0.

```
# checking for null values
all_data %>% summarise_all( ~ sum(is.na(.)))
```

```
## # A tibble: 1 x 8
##   InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID
##       <int>     <int>       <int>    <int>       <int>     <int>      <int>
## 1         0         0        1454        0           0         0     135080
## # i 1 more variable: Country <int>
```

```
# handling null values
all_data <- all_data %>%
  mutate(
    Description = ifelse(is.na(Description) |
                           Description == "", "NA", Description),
    CustomerID = ifelse(is.na(CustomerID), 0, CustomerID)
  )



# checking for null values again
all_data %>% summarise_all( ~ sum(is.na(.)))
```

```
## # A tibble: 1 x 8
##   InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID
##       <int>     <int>       <int>    <int>       <int>     <int>      <int>
## 1         0         0           0        0           0         0          0
## # i 1 more variable: Country <int>
```

All the null values are handled now.

Answering Questions

1. How many distinct countries are there, and how many unique products are sold to each country?

From the output, it looks like UK, EIRE (or Ireland), Germany occupy the top 3 countries where the highest number of unique items are purchased. A stunning 4000+ unique items have been purchased by UK customers. And almost half of it (~2000+) are purchased by

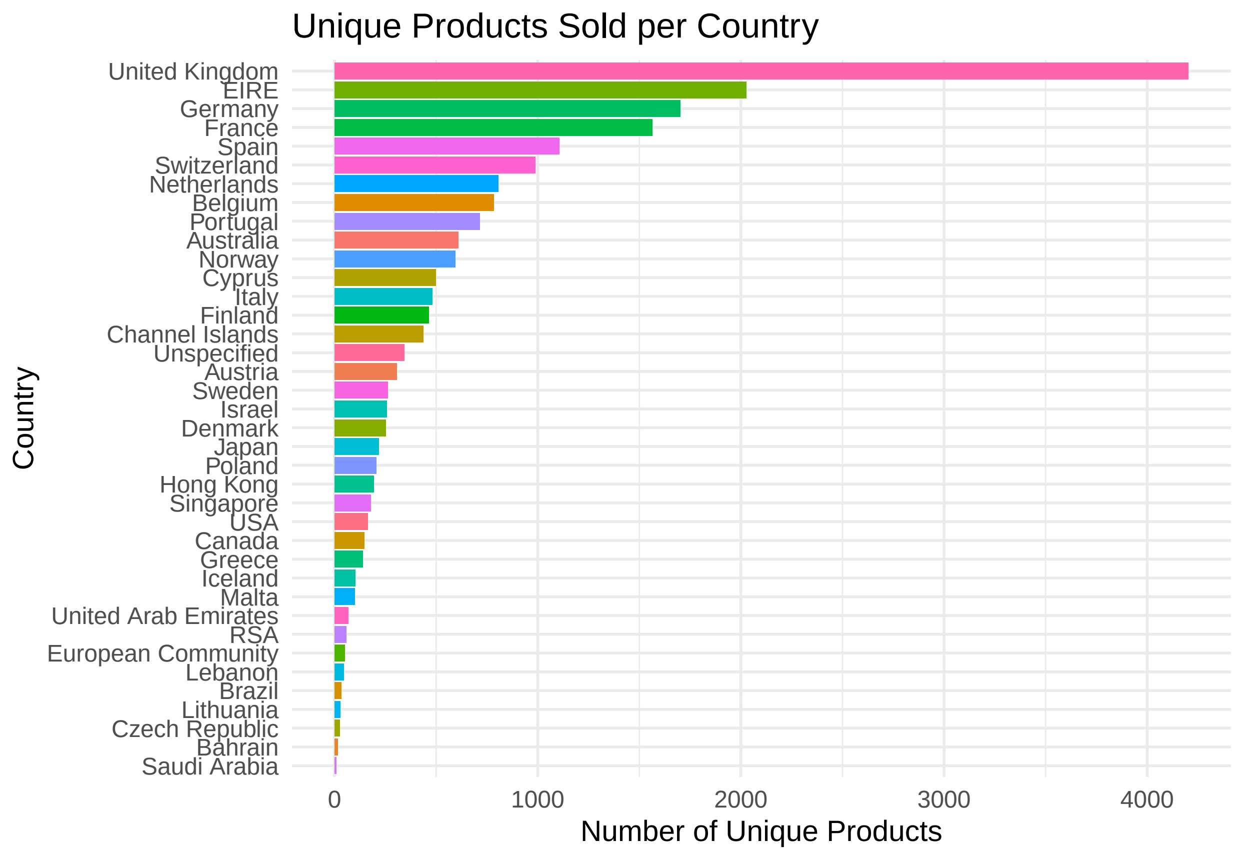
EIRE (or Ireland), and close to 1500+ products by Germany and France respectively. Looks like the least number of purchases have been made by Saudi Arabia, Bahrain and Czech Republic

```
# SQL query to count distinct countries and unique products per country
distinct_count_query <- "
SELECT
  Country,
  COUNT(DISTINCT Description) AS UniqueProducts
FROM
  `ecommerce-data-407200.uk_ecommerce_data.ecommerce_data`
GROUP BY
  Country
ORDER BY
  UniqueProducts DESC
"

# Get the data
distinct_counts <- dbGetQuery(con, distinct_count_query)



ggplot(distinct_counts,
       aes(
         x = reorder(Country, UniqueProducts),
         y = UniqueProducts,
         fill = Country
       )) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(x = "Country", y = "Number of Unique Products", title = "Unique Products Sold per
  theme_minimal() +
  theme(legend.position = "none")
```



2. Is there a seasonal pattern in the sales data, indicating peak shopping periods?

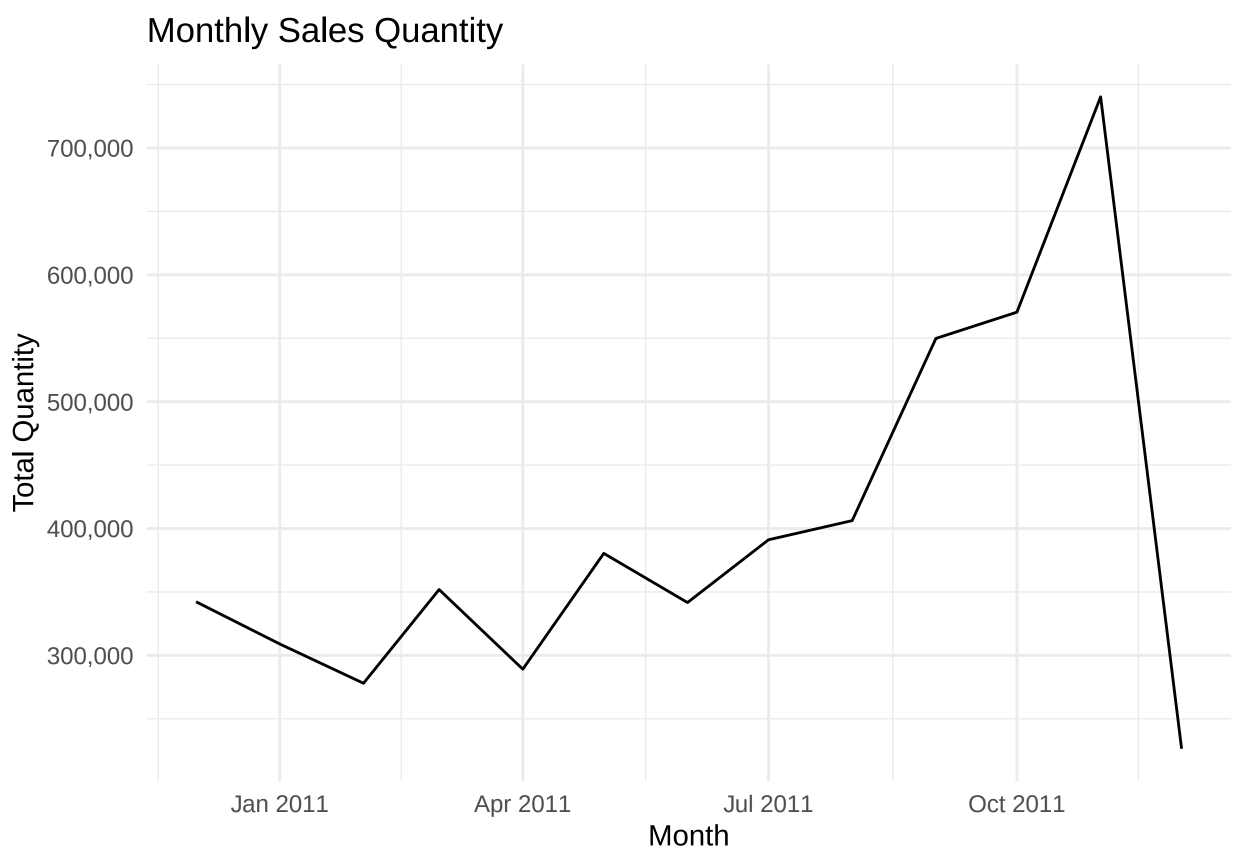
A line graph that shows the total quantity of products sold each month over a given period in 2011. The y-axis is labeled with the total quantity sold, formatted with commas for readability, and the x-axis shows the months from January to December. There's a noticeable peak towards the end of the year, which could indicate seasonal sales increases (commonly seen around holidays like Christmas). The sharp decline right after the peak could suggest a data cut-off or a steep drop in sales post-holiday season.

```r
all_data$InvoiceDate <- ymd_hms(all_data$InvoiceDate)

# Remove rows with NA in InvoiceDate
all_data <- all_data[!is.na(all_data$InvoiceDate),]

# Aggregate data by month
monthly_sales <- all_data %>%
  mutate(Month = floor_date(InvoiceDate, "month")) %>%
  group_by(Month) %>%
  summarise(MonthlyTotal = sum(Quantity))

# Visualize the seasonal sales data with formatted labels
ggplot(monthly_sales, aes(x = Month, y = MonthlyTotal)) +
  geom_line() +
  scale_y_continuous(labels = comma) +  # Format the y-axis labels
  labs(title = "Monthly Sales Quantity", x = "Month", y = "Total Quantity") +
  theme_minimal()
```



3. What are the shopping trends of some specific keywords?

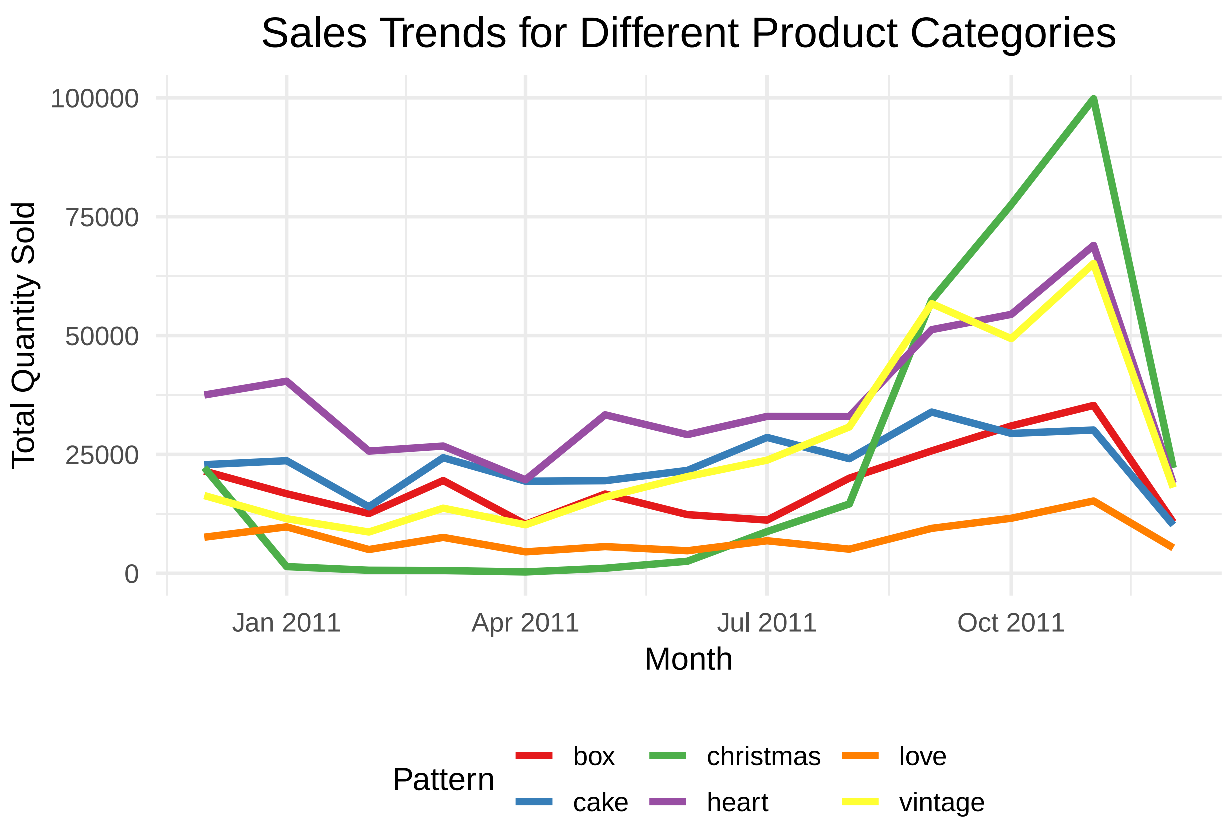
The graph illustrates shopping trends for various product themes based on keyword analysis throughout 2011. It's evident that products related to 'christmas' spiked dramatically in sales towards the end of the year, which aligns with the holiday season. Similarly, 'heart' themed items show an increase around February, likely coinciding with Valentine's Day. 'Vintage' products have a steady climb over the year, suggesting a consistent interest that peaks towards the end of the year. This type of analysis helps identify seasonal trends and customer preferences, which can guide inventory and marketing strategies.

```r
# Define a function to filter and aggregate sales for a given pattern
analyze_pattern_sales <- function(pattern) {
  all_data %>%
    filter(str_detect(tolower(Description), tolower(pattern))) %>%
    mutate(Month = floor_date(ymd_hms(InvoiceDate), "month")) %>%
    group_by(Month) %>%
    summarise(TotalQuantity = sum(Quantity, na.rm = TRUE),
              Pattern = pattern)
}

# Define a vector of patterns to search in the product descriptions
patterns <-
  c("heart", "christmas", "love", "vintage", "cake", "box")

# Use map_df from purrr to apply the function to each pattern and combine the results
pattern_sales_monthly <- map_df(patterns, analyze_pattern_sales)

ggplot(pattern_sales_monthly,
       aes(x = Month, y = TotalQuantity, color = Pattern)) +
  geom_line(linewidth = 1.3) +  # Increase line thickness
  scale_color_brewer(palette = "Set1") +  # Use a more colorful palette
  labs(title = "Sales Trends for Different Product Categories",
       x = "Month",
       y = "Total Quantity Sold") +
  theme_minimal(base_size = 14) +  # Increase the base text size for clarity
  theme(
    legend.position = "bottom",
    legend.title = element_text(size = 12),
    # Increase legend title text size
    legend.text = element_text(size = 10),
    # Increase legend text size
    axis.title = element_text(size = 12),
    # Increase axis title text size
    axis.text = element_text(size = 10),
    # Increase axis text size
    plot.title = element_text(size = 16, hjust = 0.5)
  )  # Increase and center the plot title size
```



Conclusions

After a thorough examination of e-commerce transactions, this project identified a wide range of national customers, with the UK, Germany, and EIRE accounting for the majority of unique product purchases. Seasonal trends demonstrated a notable increase in sales towards the end of the year, in line with international purchasing patterns throughout the holiday season. The data from the UK showed a range of interests, from historical to practical objects. ‘Christmas’ and ‘love’ themed merchandise peaked during their respective celebrating periods, highlighting the further impact of seasonal events on purchase patterns.

Future Scope

This project could be expanded by incorporating predictive analytics to forecast sales trends and inventory demands. Integrating customer sentiment analysis through product reviews and social media data could offer deeper insights into consumer behavior. Additionally, exploring the impact of marketing campaigns on sales volumes would provide a more holistic view of e-commerce dynamics.

References

- UCI Machine Learning Repository - https://archive.ics.uci.edu/dataset/352/online+retail
- Google big query - https://cloud.google.com/bigquery/docs
- bigrquery package - https://bigrquery.r-dbi.org/
- purrr package - https://purrr.tidyverse.org/
- scales package - https://scales.r-lib.org/