Clean Epicurious Data

Leonard Armstrong

11/5/2018

This script is responsible for preparing the raw Epicurious data for further statistical analysis. The raw data file is assumed to be located in the current working directory.

# LIBRARIES

Load any required libraries

library("dplyr")

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library("stringi")

## READ RECIPE DATA

Read Epicurious recipe data in from a CSV file, [downloaded from Kaggle](https://www.kaggle.com/hugodarwood/epirecipes).

# Local location of recipe CSV file from Kaggle  
recipe.csv.filename <- "epi\_r.csv"  
  
# Specifying "stringsAsFactors = FALSE" will prevent the title from being  
# coerced into factors.   
recipe.data <- read.csv(recipe.csv.filename,  
 header = TRUE,   
 stringsAsFactors = FALSE)  
  
# Record original data volume.  
volume.data = c(original = nrow(recipe.data))

## REMOVE RECORDS WITH NA NUMERIC DATA

Remove records that have no data (NA) in one or more of the five quantitative categories.

# Remove records with any numeric value set to NA.  
recipe.data <- recipe.data %>%  
 filter(!is.na(recipe.data$rating) &  
 !is.na(recipe.data$calories) &  
 !is.na(recipe.data$protein) &  
 !is.na(recipe.data$fat) &  
 !is.na(recipe.data$sodium))  
  
# Capture volume after removal of NAs  
volume.data <- c(volume.data, na.removal = nrow(recipe.data))

## REMOVE RECORDS WITH IMPOSSIBLE DATA VALUES

Remove records that have “impossible” data values 1. Ratings of 0. Ratings start with 1 “fork” so a 0 rating is not possible. 2. Calories of 0. Unless its a recipe for water, recipes have to have more than 0 calories.

# Since ratings start with 1 "fork" we remove any recipe with a 0 rating  
recipe.data <- recipe.data %>%  
 filter(recipe.data$rating > 0)   
  
# And remove any 0 calorie recipes  
recipe.data <- recipe.data %>%  
 filter(recipe.data$calories > 0)   
  
# Capture volume after removal of NAs  
volume.data <- c(volume.data, impossible.values.removal = nrow(recipe.data))

# REMOVE OUTLIER DATA

We also eliminate any outliers. This will essentially get rid of ridiculously high numbers that were clearly in error (e.g., 200,000 calories) as well as eliminate some recipes that have been rated on an entire recipe basis rather than on a per-serving basis.

# Define names for the 6 numeric attributes (variables).  
var.names = c("Rating", "Calories", "Protien", "Fat", "Sodium")  
  
# We will use the boxplot function to determine outliers. Boxplot, by default,  
# generates upper and lower limits based on no more than 1.5 x the inter-  
# quartile range (IQR). Any data point below min or above max will be am  
# outlier.  
recipe.bp <-   
 boxplot(recipe.data$rating,   
 recipe.data$calories,   
 recipe.data$protein,   
 recipe.data$fat,   
 recipe.data$sodium,  
 names = var.names,   
 plot = FALSE) # Prevent onscreen plot. Only capture the data.  
  
## Define symobolic names to make indexing the results more readable.  
# Result row names  
min.index <- 1  
max.index <- 5  
# Result column names  
calories.index <- 2  
protein.index <- 3  
fat.index <- 4  
sodium.index <- 5  
  
# Capture key result values into symbolic names  
calories.low <- recipe.bp$stats[min.index, calories.index]  
calories.hi <- recipe.bp$stats[max.index, calories.index]  
protein.low <- recipe.bp$stats[min.index, protein.index]  
protein.hi <- recipe.bp$stats[max.index, protein.index]  
fat.low <- recipe.bp$stats[min.index, fat.index]  
fat.hi <- recipe.bp$stats[max.index, fat.index]  
sodium.low <- recipe.bp$stats[min.index, sodium.index]  
sodium.hi <- recipe.bp$stats[max.index, sodium.index]  
  
# Remove records with calorie outliers  
recipe.data <- recipe.data %>%  
 filter(between(recipe.data$calories, calories.low, calories.hi))  
  
# Remove records with protein outliers  
recipe.data <- recipe.data %>%  
 filter(between(recipe.data$protein, protein.low, protein.hi))  
  
# Remove records with fat outliers  
recipe.data <- recipe.data %>%  
 filter(between(recipe.data$fat, fat.low, fat.hi))  
  
# Remove records with sodium outliers.  
recipe.data <- recipe.data %>%  
 filter(between(recipe.data$sodium, sodium.low, sodium.hi))  
  
# Capture volume after removal of outliers  
volume.data <- c(volume.data, outliers = nrow(recipe.data))

# REMOVE DUPLICATE RECORDS

Remove all but the first instance of duplicate records. Records are determined to be duplicates if **all** of the following are true. 1. The titles are identical 2. The calories are identical 3. The sodium grams are identical 4. The fat grams are identical 5. The protein grams are identical

# Remove duplicates. Records have their title, calories, sodium, fat, and  
# protein values concatenated into one string to simplify the comparison.  
recipe.data <-   
 recipe.data[  
 !duplicated(  
 stri\_join(  
 recipe.data$title,  
 recipe.data$calories,   
 recipe.data$sodium,   
 recipe.data$fat,   
 recipe.data$protein)), ]  
  
# Capture volume after removal of duplicates  
volume.data <- c(volume.data, duplicates = nrow(recipe.data))

# This section will run some tests to create a new data frame from the JSON

# data.

library(“rjson”)

jsonField.to.vector <- function (jsonData, fieldName) { unlist( lapply( jsonData, function(x) { if (is.null(x[[fieldName]])) { NA } else { x[[fieldName]] } # if } # anonymous function ) # lapply ) # unlist } # jsonField.to.vector

recipe.json.filename <- “full\_format\_recipes.json” recipe.extendedData <- fromJSON(file = recipe.json.filename)

title <- trimws(jsonField.to.vector(recipe.extendedData, “title”), which = “both”) rating <- jsonField.to.vector(recipe.extendedData, “rating”) fat <- trimws(jsonField.to.vector(recipe.extendedData, “fat”), which = “both”) calories <- trimws(jsonField.to.vector(recipe.extendedData, “calories”), which = “both”)