Project: Data Wrangling Exercise 1: Basic Data Manipulation

In this exercise, you'll work with a toy data set showing product purchases from an electronics store. While the data set is small and simple, it still illustrates many of the challenges you have to address in real-world data wrangling! The data set and exercise are inspired by this blog post.

Getting started

The data is in an Excel file here called *refine.xlsx.* Right away, you'll notice that the data set has a few issues:

- There are four brands: Philips, Akzo, Van Houten and Unilever. However, there are many different spellings and capitalizations of those names!
- The product code and number are combined in one column, separated by a hyphen.

Exercise

Using R, clean this data set to make it easier to visualize and analyze. Specifically, these are the tasks you need to do:

0: Load the data in RStudio

Save the data set as a CSV file called *refine_original.csv* and load it in RStudio into a data frame.

*// dataw = *refine_original* //*

```
> setwd("the path of data directory")
> dataw <- read.csv('refine_original.csv', header = TRUE, check.names=FALSE)
```

```
*// Install R packages : dplyr, tidyr, devtools, dummies//*

> library("dplyr", lib.loc="C:/RStudio/R-3.5.0/library")
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("tidyr", lib.loc="C:/RStudio/R-3.5.0/library")
> library(devtools)
dummies-1.5.6 provided by Decision Patterns
> View(dataw)
```

1: Clean up brand names

Clean up the 'company' column so all of the misspellings of the brand names are standardized. For example, you can transform the values in the column to be: *philips, akzo, van houten* and *unilever* (all lowercase).

```
> dataw$company <- sub(".*\\ps", replacement = "Philips", x = dataw$company)
> View(dataw)*// make sure data changed according to the code above//*
> dataw$company <- sub(".*\\pS", replacement = "Philips", x = dataw$company)
> dataw$company <- sub(pattern = "^ak.*", replacement = "Akzo", x= dataw$comp
any)
> dataw$company <- sub("^AK.*", replacement = "Akzo", x= dataw$company)
>
>
> dataw$company <- sub("^Van.*", replacement = "Van Houten", x= dataw$company
)
>
> dataw$company <- sub("^van.*", replacement = "Van Houten", x= dataw$company
)
> dataw$company <- sub("^uni.*", replacement = "Unilever", x= dataw$company)
> View(dataw)  *// make sure data changed according to the code above//*
```

2: Separate product code and number

Separate the product code and product number into separate columns i.e. add two new columns called *product_code* and *product_number*, containing the product code and number respectively

```
> dataw <- separate(dataw, "Product code / number" , c("product_code" ,"produ
ct_number"), sep = "-")

> View(dataw) *// make sure data changed according to the code above//*
```

3: Add product categories

You learn that the product codes actually represent the following product categories:

- p = Smartphone
- v = TV
- x = Laptop
- q = Tablet

In order to make the data more readable, add a column with the product category for each record.

```
> dataw$product_code <- sub("p", replacement = "Smartphone", x = dataw$produc
t_code)

> dataw$product_code <- sub("v", replacement = "TV", x = dataw$product_code)

> dataw$product_code <- sub("x", replacement = "Laptop", x = dataw$product_co
de)
> dataw$product_code <- sub("q", replacement = "Tablet", x = dataw$product_co
de)

> View(dataw) *// make sure data changed according to the code above//*
```

4: Add full address for geocoding

You'd like to view the customer information on a map. In order to do that, the addresses need to be in a form that can be easily geocoded. Create a new column *full_address* that concatenates the three address fields (*address, city, country*), separated by commas.

```
> dataw <- dataw %>% mutate(full_address = paste(address, city, country, sep
= ","))
> View(dataw)

*//After wxcuting above code , the new column “full_address will added at end
(after the “names” column). To move the column next to “product_number” , use
below code //*

> dataw %>% select(company,product_code,product_number,full_address,address,c
ity,country,name)----- *// this will only changing at console level not in th
e data frame file //*

> # moving the coulmn for: full_address, use below code

> dataw <- dataw [,c(1,2,3,8,4,5,6,7)]

> View(dataw)
```

5: Create dummy variables for company and product category

Both the company name and product category are categorical variables i.e. they take only a fixed set of values. In order to use them in further analysis you need to create dummy variables. Create dummy binary variables for each of them with the prefix *company_* and *product_* i.e.,

1. Add four binary (1 or 0) columns for company: *company_philips, company_akzo, company_van_houten* and *company_unilever.*

```
> cbind(dataw,dummy(dataw$company,sep = "-"))  *// tried cbind but the code
will only changing at console not in the data frame file //*

> dataw <- mutate(dataw, company_Philips = ifelse(company == "Philips", 1, 0)
)

> View(dataw)

> dataw <- mutate(dataw, company_Akzo = ifelse(company == "Akzo", 1, 0))

> dataw <- mutate(dataw, company_Van_Houten = ifelse(company == "Van_Houten",
1, 0))
> dataw <- mutate(dataw, company_Unilever = ifelse(company == "Unilever", 1,
0))

> View(dataw)
```

2. Add four binary (1 or 0) columns for product category: product_smartphone, product_tv, product_laptop and product_tablet.

```
 > dataw <- mutate(dataw,product_smartphone = ifelse(product_code == "Smartpho
ne", 1, 0))

> View(dataw)

> dataw <- mutate(dataw,product_tv = ifelse(product_code == "TV", 1, 0))
> dataw <- mutate(dataw,product_laptop  = ifelse(product_code == "Laptop", 1,
0))
> dataw <- mutate(dataw,product_tablet  = ifelse(product_code == "Tablet", 1,
0))
> View(dataw)
```

6: Submit the project on Github

Include your code, the original data as a CSV file *refine_original.csv*, and the cleaned up data as a CSV file called *refine_clean.csv*.

```
*// Use the 'write.csv( )' command to save the file: Ex- write.csv(healthstud
y,'healthstudy2.csv') //*

> write.csv(dataw, "refine_clean.csv.")

> View(refine_clean.csv)


Error in View : object 'refine_clean.csv' not found
```