initial_exploration

Librairies and data import

We start by importing the Hadleyverse and importing our data.

```
libs <- c('readr', 'lubridate', 'magrittr', 'tidyr', 'dplyr')
sapply(libs, require, character.only = TRUE)
train <- read_csv('~/Documents/kaggle_data/kaggle_rossmann/data/train.csv')
test <- read_csv('~/Documents/kaggle_data/kaggle_rossmann/data/test.csv')
stores <- read_csv('~/Documents/kaggle_data/kaggle_rossmann/data/store.csv')
```

We start by examining the train and stores data (the test data is structured as the train).

```
head(train, 3)
```

```
## Source: local data frame [3 x 9]
##
##   Store DayOfWeek       Date Sales Customers  Open Promo StateHoliday
##   (int)     (int)     (date) (int)     (int) (int) (int)        (int)
## 1     1         5 2015-07-31  5263       555     1     1            0
## 2     2         5 2015-07-31  6064       625     1     1            0
## 3     3         5 2015-07-31  8314       821     1     1            0
## Variables not shown: SchoolHoliday (int)
```

```
head(stores, 3)
```

```
## Source: local data frame [3 x 10]
##
##   Store StoreType Assortment CompetitionDistance CompetitionOpenSinceMonth
##   (int)     (chr)      (chr)               (int)                      (int)
## 1     1         c          a                1270                          9
## 2     2         a          a                 570                         11
## 3     3         a          a               14130                         12
## Variables not shown: CompetitionOpenSinceYear (int), Promo2 (int),
##   Promo2SinceWeek (int), Promo2SinceYear (int), PromoInterval (chr)
```

The train set contains information on **Sales**, which is the data to predict (the gross volume of cash made by a store during a day), while the stores data contains informations about the stores (distance to their competition for example).

We start by joining the datasets using the join method of dplyr.

```
train <- inner_join(train, stores, by = 'Store')
test <- inner_join(test, stores, by = 'Store')
head(train, 3)
```

```
## Source: local data frame [3 x 18]
##
```

```
##   Store DayOfWeek       Date Sales Customers  Open Promo StateHoliday
##   (int)     (int)     (date) (int)     (int) (int) (int)        (int)
## 1     1         5 2015-07-31  5263       555     1     1            0
## 2     2         5 2015-07-31  6064       625     1     1            0
## 3     3         5 2015-07-31  8314       821     1     1            0
## Variables not shown: SchoolHoliday (int), StoreType (chr), Assortment
##   (chr), CompetitionDistance (int), CompetitionOpenSinceMonth (int),
##   CompetitionOpenSinceYear (int), Promo2 (int), Promo2SinceWeek (int),
##   Promo2SinceYear (int), PromoInterval (chr)
```

NA filling

This sections explores basic strategies for NA filling. First we use the na_proportion function (defined in utilities) to show which columns have NAs and in which proportion.

```
na_proportion <- function(dataset, digits = 2) {
  # returns the proportion of missing values
  # in each column of dataset, for which there is NAs
  na_col <- names(which(apply(dataset, 2, function(x) any(is.na(x))) == TRUE))
  if (length(na_col) >= 1) {
    sapply(select(dataset, one_of(na_col)),
           function(x) round(sum(is.na(x)) / length(x), digits = digits))
  } else {
    cat("No missing values.")
  }
}
na_proportion(train)
```

```
##              StateHoliday       CompetitionDistance
##                      0.03                      0.00
## CompetitionOpenSinceMonth  CompetitionOpenSinceYear
##                      0.32                      0.32
##           Promo2SinceWeek           Promo2SinceYear
##                      0.50                      0.50
```

Competition Distance

We see a few missing values in the CompetitionDistance field, we can show explicitly those shops which lacks this information.

```
filter(stores, is.na(CompetitionDistance)) %>% select(Store, CompetitionDistance)
```

```
## Source: local data frame [3 x 2]
##
##   Store CompetitionDistance
##   (int)               (int)
## 1   291                  NA
## 2   622                  NA
## 3   879                  NA
```

It seems reasonable to imagine that those three stores do not have any competition?

We make the disctuable assumption to fill those values by the maximum distance to the competition observed.

```
max_comp_dist <- max(stores$CompetitionDistance, na.rm = TRUE)
train[is.na(train$CompetitionDistance), 'CompetitionDistance'] <- max_comp_dist
test[is.na(test$CompetitionDistance), 'CompetitionDistance'] <- max_comp_dist
```

Some shop in the test set are closed

It should be noticed that some Sales to be predicted are for closed shops.

```
test %>% filter(Open == 0) %>% head(3)
```

```
## Source: local data frame [3 x 17]
##
##      Id Store DayOfWeek       Date  Open Promo StateHoliday SchoolHoliday
##   (int) (int)     (int)     (date) (int) (int)        (int)         (int)
## 1   544   703         4 2015-09-17     0     1            0             0
## 2   677   879         4 2015-09-17     0     1            0             0
## 3   841  1097         4 2015-09-17     0     1            0             0
## Variables not shown: StoreType (chr), Assortment (chr),
##   CompetitionDistance (int), CompetitionOpenSinceMonth (int),
##   CompetitionOpenSinceYear (int), Promo2 (int), Promo2SinceWeek (int),
##   Promo2SinceYear (int), PromoInterval (chr)
```

We need to guarantee that for those days our prediction are 0. To do so we define a function in utilities which should **always** be applied to any prediction vector.

```
set_pred_closed <- function(pred_vec) {
  # this function set to zero the Sales predictions
  # anytime the shop is closed in the test set
  pred_vec[which(test$Open == 0)] <- 0
}
```

Some shop have missing Open value

```
test %>% filter(is.na(Open)) %>% head(3)
```

```
## Source: local data frame [3 x 17]
##
##      Id Store DayOfWeek       Date  Open Promo StateHoliday SchoolHoliday
##   (int) (int)     (int)     (date) (int) (int)        (int)         (int)
## 1   480   622         4 2015-09-17    NA     1            0             0
## 2  1336   622         3 2015-09-16    NA     1            0             0
## 3  2192   622         2 2015-09-15    NA     1            0             0
## Variables not shown: StoreType (chr), Assortment (chr),
##   CompetitionDistance (int), CompetitionOpenSinceMonth (int),
##   CompetitionOpenSinceYear (int), Promo2 (int), Promo2SinceWeek (int),
##   Promo2SinceYear (int), PromoInterval (chr)
```

To see what the filling should be we need to inspect the corresponding dates. We can notice already that the Promo takes the value 1 for some of those days: so even the shop is open and there was a bug, or the shop was in promo and closed !

Let us do saome date aprsing (using lubridate and the fact that this is a german retailer).

```
train$Date <- ymd(train$Date, tz = "Europe/Berlin")
test$Date <- ymd(test$Date, tz = "Europe/Berlin")
```

Now we can explore the days where we see that anomaly.

```
test_open_anomaly <- filter(test, is.na(Open))
wday(test_open_anomaly$Date, label = TRUE)
```

```
##  [1] Thurs Wed   Tues  Mon   Sat   Fri   Thurs Wed   Tues  Mon   Sat
## Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat
```

No sunday ! When is this shop usually closed ?

```
store_622 <- filter(test, Store == 622)
store_622_closed <- filter(store_622, Open == 0, StateHoliday == 0)
unique(wday(store_622_closed$Date, label = TRUE))
```

```
## [1] Sun
## Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat
```

It is only closed on Sunday ! Then we decide to fill those values with 1's (i.e. the shop was probably open).

```
test[is.na(test$Open), 'Open'] <- 1
```

The state holiday problem

StateHoliday never takes the value 1!

```
sum(train$StateHoliday == 0, na.rm = TRUE)
```

```
## [1] 986159
```

```
sum(train$StateHoliday == 1, na.rm = TRUE) # jamais 1!
```

```
## [1] 0
```

```
sum(test$StateHoliday == 0, na.rm = TRUE)
```

```
## [1] 40908
```

```
sum(test$StateHoliday == 1, na.rm = TRUE) # jamais 1!
```

```
## [1] 0
```

Let us look at some of those rows where StateHoliday is missing.

```
filter(train, is.na(StateHoliday)) %>% select(Date)
```

```
## Source: local data frame [31,050 x 1]
##
##          Date
##        (time)
## 1  2015-06-04
## 2  2015-06-04
## 3  2015-06-04
## 4  2015-06-04
## 5  2015-06-04
## 6  2015-06-04
## 7  2015-06-04
## 8  2015-06-04
## 9  2015-06-04
## 10 2015-06-04
## ..        ...
```

Those dates are probably holidays, so we decide to fill by 1 all those StateHoliday missing values.

```
train[is.na(train$StateHoliday), 'StateHoliday'] <- 1
test[is.na(test$StateHoliday), 'StateHoliday'] <- 1
```