Milestone: Data Story

Prediction of restaurant inspections in NYC

Contents

1 Introduction

Food service establishments (FEA's) in New York City receive an unannounced inspection at least once a year by the Health Department to monitor their compliance with food safety regulations. Recorded violations result in a score, which in turn determines the grade issued. A score from 0 to 13 is an A-Grade, 14 to 27 a B-grade, and scores above 27 result in a C-grade.

FEA's that receive an A-grade on their first inspection post it immediately. Restaurants that do not receive an A-Grade on their first inspection can improve their sanitary conditions and are re-inspected. If a restaurant does not receive an A-grade on the re-inspection, it must either post the letter-grade or 'grade pending'. Link

The goal of this project is to use data on these restaurant inspections to predict the duration until the next inspection occurs.

The data set contains information on inspections in New York City carried out by the Department of Health and Hygiene (DOHMH) between 2010 and 2017. It contains the following variables:

- CAMIS: a unique identifier for the restaurant
- DBA: name of (doing business as) the restaurant
- BORO: borough in which the restaurant is located

- BUILDING: building number for the restaurant
- STREET: street name at which the restaurant is located
- ZIPCODE: Zip code as per the (mailing) address of the restaurant
- PHONE: phone number
- CUISINE DESCRIPTION: cuisine of the restaurant
- INSPECTION DATE: date of inspection
- ACTION: action associated with the given inspection
- VIOLATION CODE: violation code associated with the given inspection
- VIOLATION DESCRIPTION: description that corresponds to violation code
- CRITICAL FLAG: indication if violation is critical or not
- SCORE: total score for a particular inspection
- GRADE: grade issued for the given inspection
- GRADE DATE: date when the current grade was issued to the restaurant
- RECORD DATE: The date when the extract was run to produce this data set
- INSPECTION TYPE: The type of inspection. A combination of the program and inspection type

```
df_raw <- read_excel( "New_York_City_Restaurants.xlsx",
                      col_types = c(rep("text", 8),
                                    "date",
                                    rep("text", 4),
                                    "numeric",
                                    "text",
                                    rep("date", 2),
                                    "text"))
```

For convenience, we rename the variables.

```
df_raw <- df_raw %>%
  clean_names() %>%
  rename(id = camis, rest_name = dba, cuisine_descr = cuisine_description,
         violation_descr = violation_description)
```

2 Exploratory Data Analysis

We will start the analysis by computing summary statistics for some of the variables in the data set.

2.1 Inspection Date

```r
df_raw <- df_raw %>%
  mutate(inspection_date = ymd(inspection_date))
df_raw %>%
  summarise(min = min(inspection_date),
            median = median(inspection_date),
            max = max(inspection_date))
```

```
## # A tibble: 1 x 3
##   min        median     max
##   <date>     <date>     <date>
## 1 1900-01-01 2016-08-10 2018-03-19
```

Apparently, there are some observations with an inspection date of "1900-01-01". Of course, these are faulty values. On closer inspection we see that values for all variables which are related to the inspection at hand are missing.

```r
df_raw %>%
  summarise(na_inspectionDate = round((sum(inspection_date == "1900-01-01")/n()), 4))
```

```
## # A tibble: 1 x 1
##   na_inspectionDate
##               <dbl>
## 1            0.0031
```

Those obervations amount to 0.31 % of all observations. Because it is more convenient, we will exclude them from the subsequent analysis.

```r
df_raw <- df_raw %>%
  filter(!inspection_date == "1900-01-01")
```

As we can see now, the remaining observations range from October 2011 to March 2018:

```r
df_raw %>%
  summarise(min = min(inspection_date),
            median = median(inspection_date),
            max = max(inspection_date))
```

```
## # A tibble: 1 x 3
##   min        median     max
##   <date>     <date>     <date>
## 1 2011-10-11 2016-08-12 2018-03-19
```

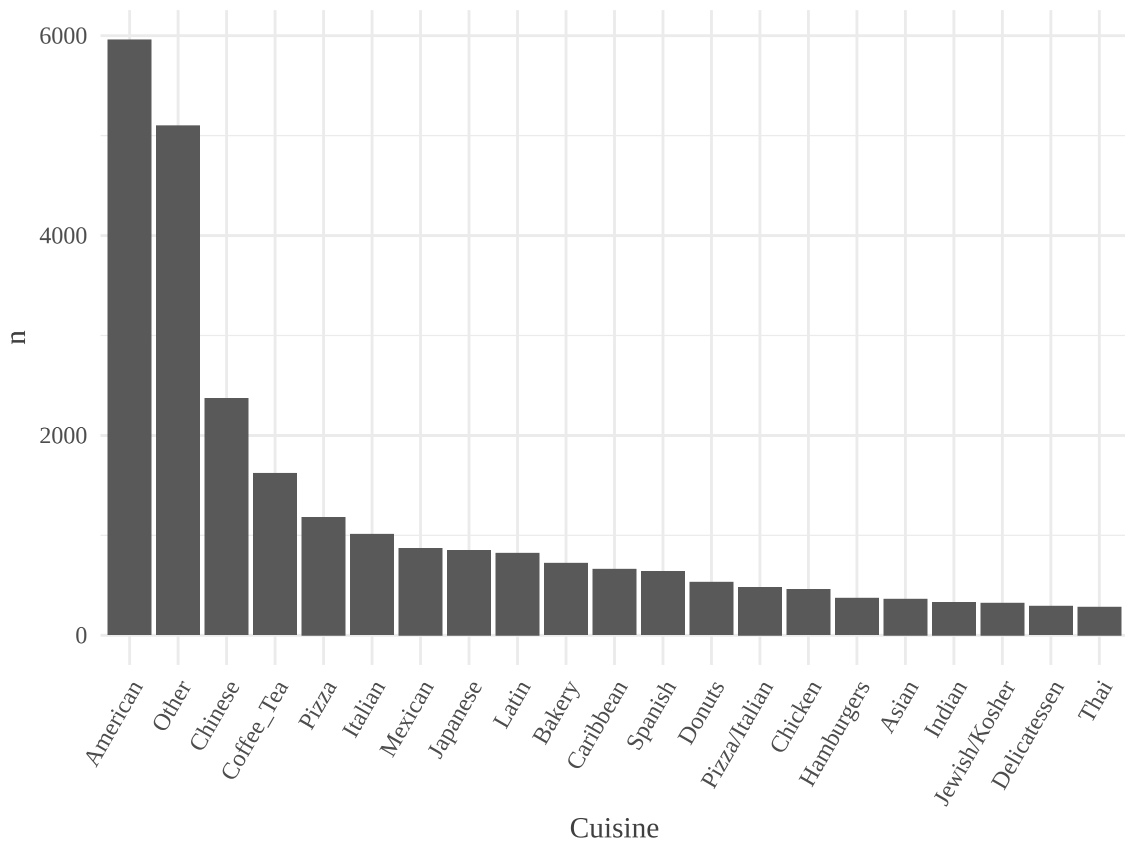
2.2 Cuisine description

```r
df_raw <- df_raw %>%
  mutate(cuisine_descr = as_factor(cuisine_descr)) %>%
  # Cuisine_description: Relabel level "CafÃƒÂ©/Coffee/Tea" and "Latin (... "
  # into labels that are more readable:
  mutate(cuisine_descr =
```

```
             fct_recode(cuisine_descr,
                        Coffee_Tea =
                          levels(cuisine_descr) %>%
                          str_subset(., "Coffee/Tea"),
                        Latin =
                          levels(cuisine_descr) %>%
                          str_subset(., "Latin"))) %>%
  # only keep the 20 most frequent cuisines
  mutate(cuisine_descr = fct_lump(f = cuisine_descr, n = 20))
```

Let us look at the distribution of 'cuisine_descr':

```
df_raw %>%
  group_by(cuisine_descr) %>%
  summarise(n = n_distinct(id)) %>%
  ungroup() %>%
  ggplot(aes(x = reorder(cuisine_descr, -n), y = n)) +
  geom_bar(stat = "identity") +
  xlab("Cuisine") +
  theme_minimal() +
  theme(
    text = element_text(
      family = "serif",
      color = "gray25"
    ),
    axis.text.x = element_text(angle = 60, hjust = 1)
  )
```



The most frequent cuisines are those labelled 'American', 'Other' and 'Chinese'.

2.3 Boros

Next, let us take a look at the distribution of 'Boro'.

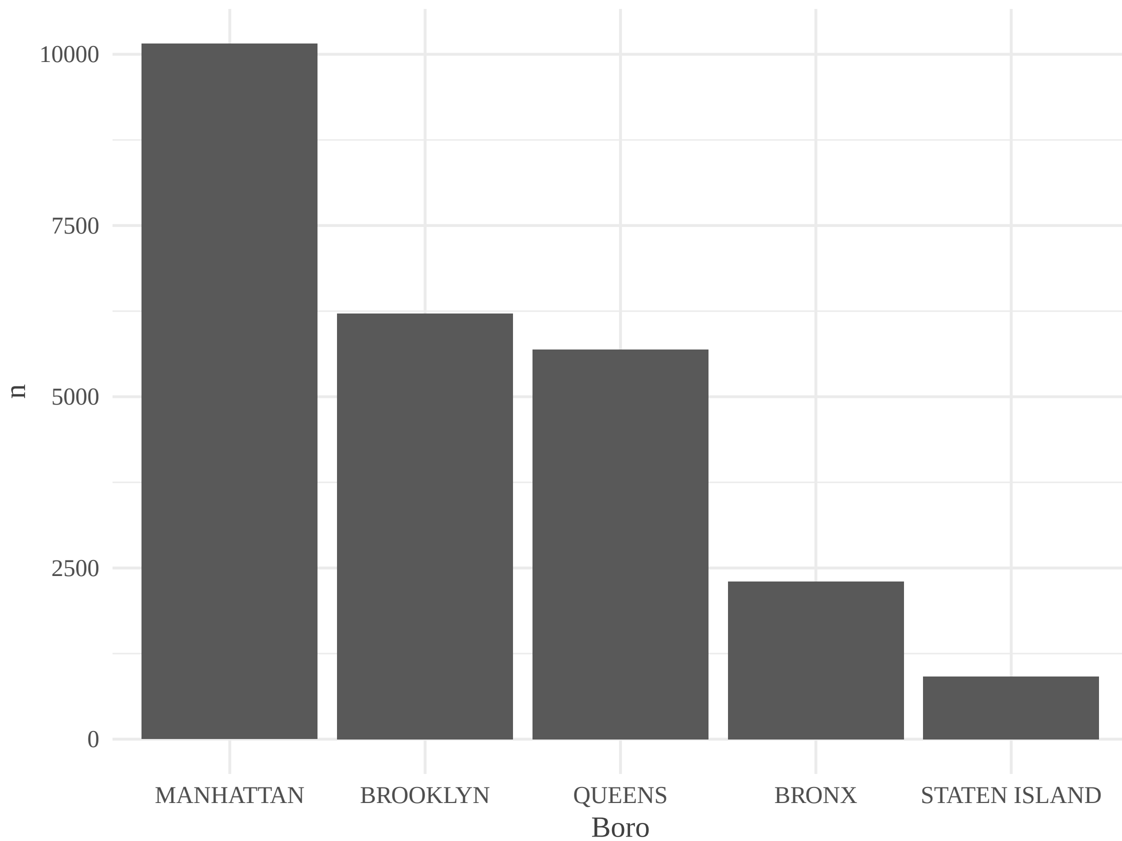
```
df_raw <- df_raw %>%
  mutate(boro = na_if(boro, "Missing")) %>%
  mutate(boro = as_factor(boro))

df_raw %>%
  count(boro)
```

```
## # A tibble: 5 x 2
##   boro              n
##   <fct>         <int>
## 1 MANHATTAN    149821
## 2 BROOKLYN      94772
## 3 QUEENS        85191
## 4 BRONX         33105
## 5 STATEN ISLAND  12900
```

```
df_raw %>%
  group_by(boro) %>%
  summarize(n = n_distinct(id)) %>%
```

```
ggplot(aes(x = reorder(boro, -n), y = n)) +
geom_bar(stat = "identity") +
theme(axis.text.x = element_text(angle=60, hjust=1)) +
xlab("Boro") +
theme_minimal() +
theme(
  text = element_text(
    family = "serif",
    color = "gray25"
  )
)
```



Most of the food service establishments are located in Manhattan, followed by Brooklyn and Queens.

2.4 Actions taken

Next, we look at the variable 'actions taken', which describes the action carried out by the DOHMH. The variable takes on the following values:

```
df_raw <- df_raw %>%
  mutate(action = as_factor(action))

df_raw %>%
  distinct(action)
```

```
## # A tibble: 5 x 1
##   action
##   <fct>
## 1 Violations were cited in the following area(s).
## 2 Establishment re-opened by DOHMH
## 3 Establishment Closed by DOHMH.  Violations were cited in the following a~
## 4 No violations were recorded at the time of this inspection.
## 5 Establishment re-closed by DOHMH
```

We rename its levels for convenience:

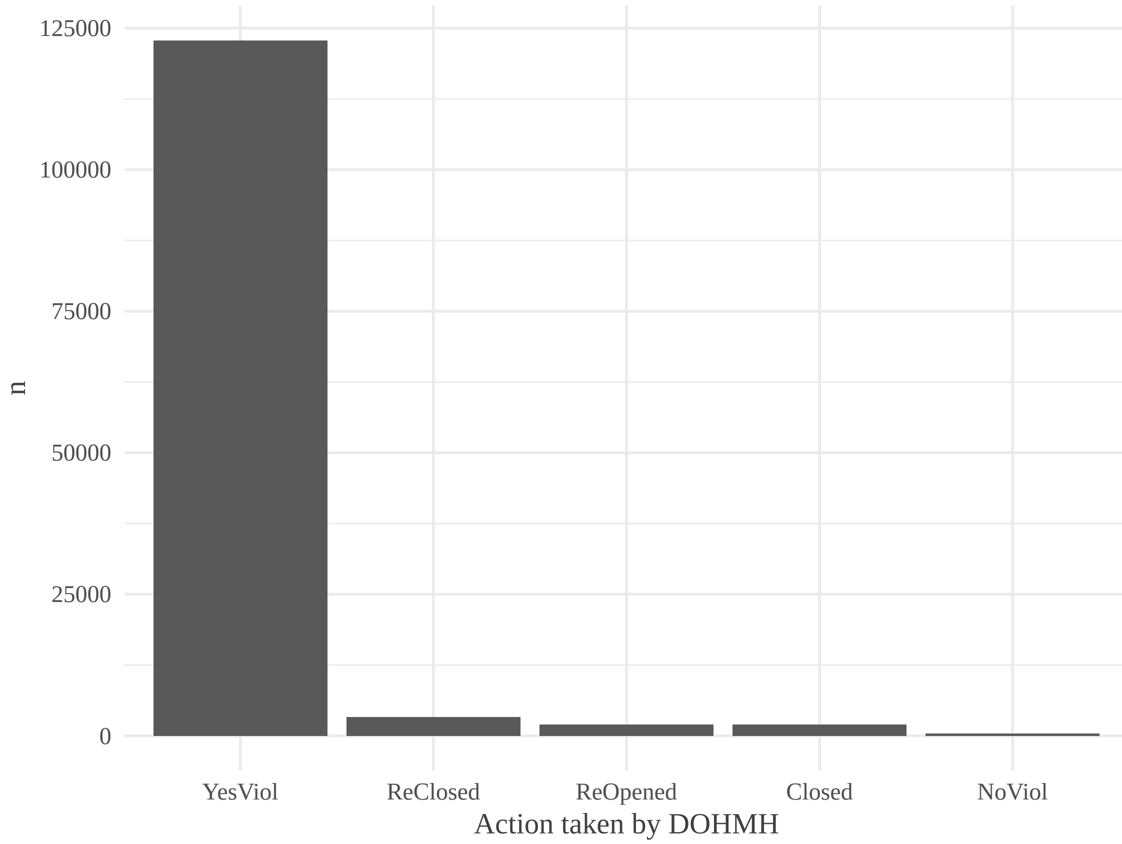
```
df_raw <- df_raw %>%
    mutate(action = fct_recode(action,
                               "YesViol" = levels(action)[1],
                               "ReOpened" = levels(action)[2],
                               "Closed" = levels(action)[3],
                               "ReClosed" = levels(action)[4],
                               "NoViol" = levels(action)[5]))

df_raw %>%
  pull(action) %>%
  levels
```

```
## [1] "YesViol"  "ReOpened" "Closed"   "ReClosed" "NoViol"
```

Let us have a look at the frequencies of the levels of 'action'.

```
df_raw %>%
  group_by(id, inspection_date) %>%
  summarise(action_taken = first(action)) %>%
  group_by(action_taken) %>%
  summarise(n = n()) %>%
  ggplot(aes(x = reorder(action_taken, -n), y = n)) +
  geom_bar(stat = "identity") +
  xlab("Action taken by DOHMH") +
  theme_minimal() +
  theme(
    text = element_text(
      family = "serif",
      color = "gray25"
    )
  )
```



The most frequent category is 'YesViol', i.e., in the majority of inspections there were violations.

2.5 Violation types

Next, we will consider the different types of violations recorded. The most common violation codes are listed below.

```
df_raw %>%
  mutate(violation_code =
      as_factor(violation_code)) %>%
  group_by(violation_code) %>%
  summarise(number = n()) %>%
  mutate(rank = rank(desc(number))) %>%
  filter(rank <= 10) %>%
  arrange(rank)
```

```
## # A tibble: 10 x 3
##    violation_code number  rank
##    <fct>           <int> <dbl>
##  1 10F             53154     1
##  2 08A             38706     2
##  3 04L             26768     3
##  4 06C             25521     4
##  5 06D             25215     5
##  6 02G             23866     6
```

```
##   7 10B             21819     7
##   8 02B             19023     8
##   9 04N             18882     9
## 10 04H              8174    10
```

10F (general violation pertaining to non-food contact surfaces) is the most common violation type. followed by 08A (facility not vermin proof), 04L (evidence of mice).

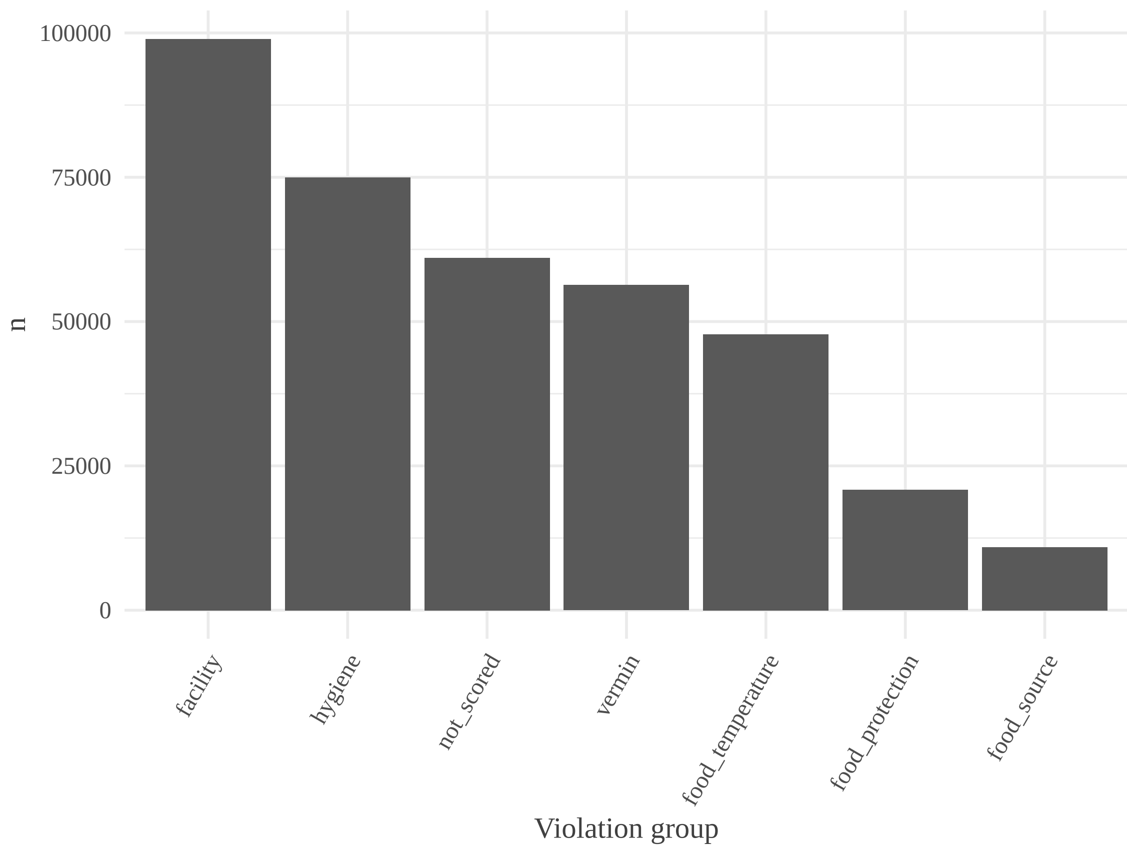
Because some of the violation types occur at low frequencies, we lump them into broader violation groups. Violations are either scored or unscored. Among scored violations, we have to distinguish between so called critical and general violations. Critical violations are deemed more severe than general violations and accordingly, are awarded more points than general violations. We end up with the self-annotaed scored categories 'vermin', 'facility', 'hygiene', 'food protection', 'food temperature', and 'food source'. Furthermore there are 'other_scored' and 'not_scored', denoting categories that encompass other scored categories and unscored violations, respectively.

```
# New variable violation_group:
df_raw <-
  df_raw %>%
  mutate(
    violation_group =
      case_when(
        violation_code %in%
          str_c("02", LETTERS[1:10]) ~ "food_temperature",
        violation_code %in%
          c(str_c("03", LETTERS[1:7]),
            str_c("09", LETTERS[1:3])) ~ "food_source",
        violation_code %in%
          str_c("04", LETTERS[1:10]) ~ "food_protection",
        violation_code %in%
          c(str_c("05", LETTERS[1:9]),
            str_c("10", LETTERS[1:10])) ~ "facility",
        violation_code %in%
            str_c("06", LETTERS[1:9]) ~ "hygiene",
        violation_code %in%
            str_c("04", LETTERS[11:15]) ~ "vermin",
        violation_code %in%
          c("07A", "99B") ~ "other_scored",
        !is.na(violation_code) ~ "not_scored"
  )
)
```

Here is a plot displaying the distribution. We will exclude the category *other_scored* since it contains very few observations.

```
df_raw %>%
  group_by(violation_group) %>%
  filter(!is.na(violation_group),
         violation_group != "other_scored") %>%
  summarize(n = n()) %>%
  ggplot(aes(reorder(violation_group, -n), n)) +
  geom_bar(stat = "identity") +
  xlab("Violation group") +
  theme_minimal() +
```

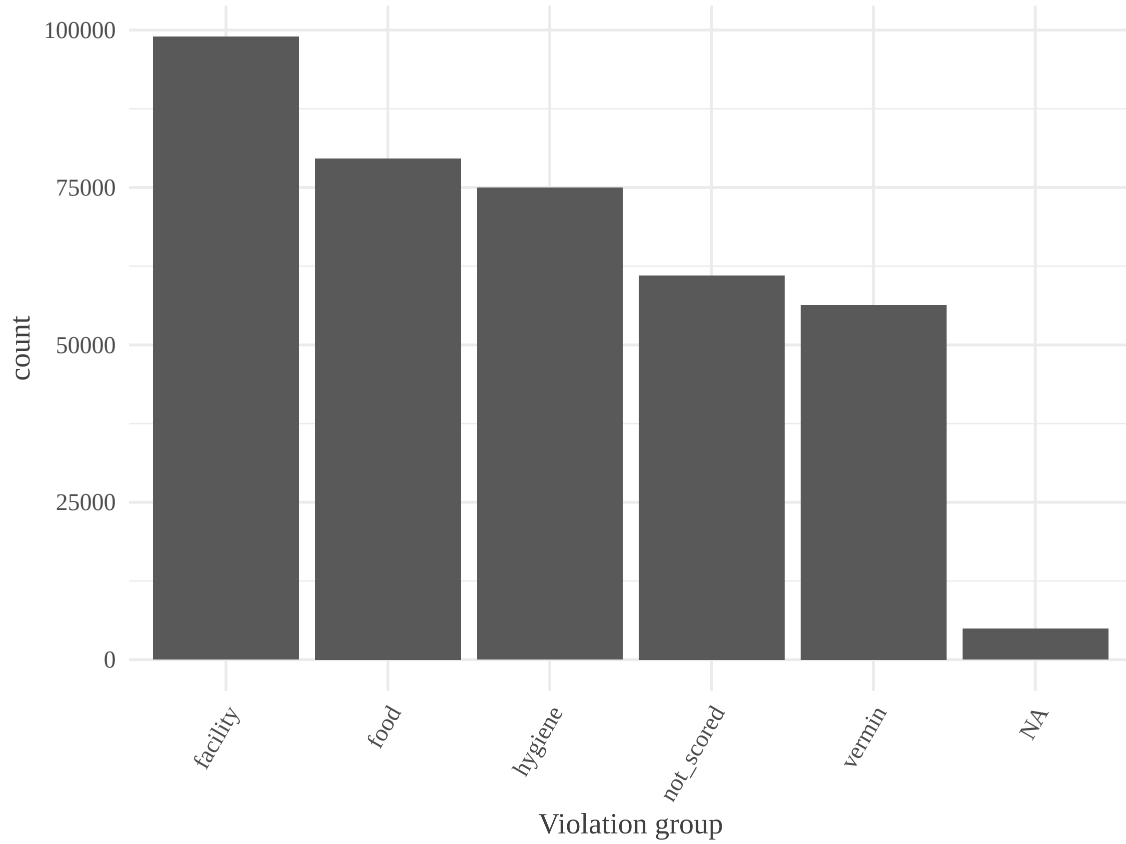
```
  theme(
    text = element_text(
      family = "serif",
      color = "gray25"
    ),
    axis.text.x = element_text(angle = 60, hjust = 1)
  )
```



We see that the most frequent category is 'facility', followed by 'hygiene' and 'not_scored'. We will lump the *food_**-categories categories *food_protection* and *other_scored* into a category labelled *food*.

```
df_raw <- df_raw %>%
  mutate(violation_group = as_factor(violation_group),
         violation_group = fct_collapse(violation_group,
                                        food = c("food_temperature",
                                                 "food_protection",
                                                 "food_source",
                                                 "other_scored"))
  )
```

Here is a display of the updated *violation_group*:



For later use, we construct indicator variables for the different violation groups.

```
df_raw <- df_raw %>%
  # Create indicator variables for levels of 'violation_group'
  mutate(
    viol_vermin =
      case_when(violation_group == "vermin" ~ 1,
                TRUE ~ 0),
    viol_facility =
      case_when(violation_group == "facility" ~ 1,
                TRUE ~ 0),
    viol_food =
      case_when(violation_group == "food" ~ 1,
                TRUE ~ 0),
    viol_hygiene =
      case_when(violation_group == "hygiene" ~ 1,
              TRUE ~ 0),
    viol_not_scored =
      case_when(violation_group == "not_scored"  ~ 1,
              TRUE ~ 0)
    )
```

2.6 Inspection types

The variable inspection type takes on 34 distinct values. Here we show ten of those values.

```r
df_raw %>%
  distinct(inspection_type) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 1
##    inspection_type
##    <chr>
##  1 Pre-permit (Operational) / Re-inspection
##  2 Administrative Miscellaneous / Initial Inspection
##  3 Pre-permit (Operational) / Initial Inspection
##  4 Cycle Inspection / Re-inspection
##  5 Cycle Inspection / Initial Inspection
##  6 Administrative Miscellaneous / Re-inspection
##  7 Trans Fat / Initial Inspection
##  8 Pre-permit (Operational) / Compliance Inspection
##  9 Cycle Inspection / Reopening Inspection
## 10 Smoke-Free Air Act / Re-inspection
```

As with the violation type, we lump some of these values together. We obtain the following values:

```r
df_raw <- df_raw %>%
  separate(
    inspection_type,
    into = c("before_slash", "after_slash"),
    sep = " / ") %>%
  mutate(
    after_slash =
      str_replace_all(after_slash, fixed(" "), fixed(""))) %>%
  mutate(
    inspection_type2 =
      str_replace_all(after_slash, fixed("-i"), fixed("I")),
    inspection_type2 = as_factor(inspection_type2))
```

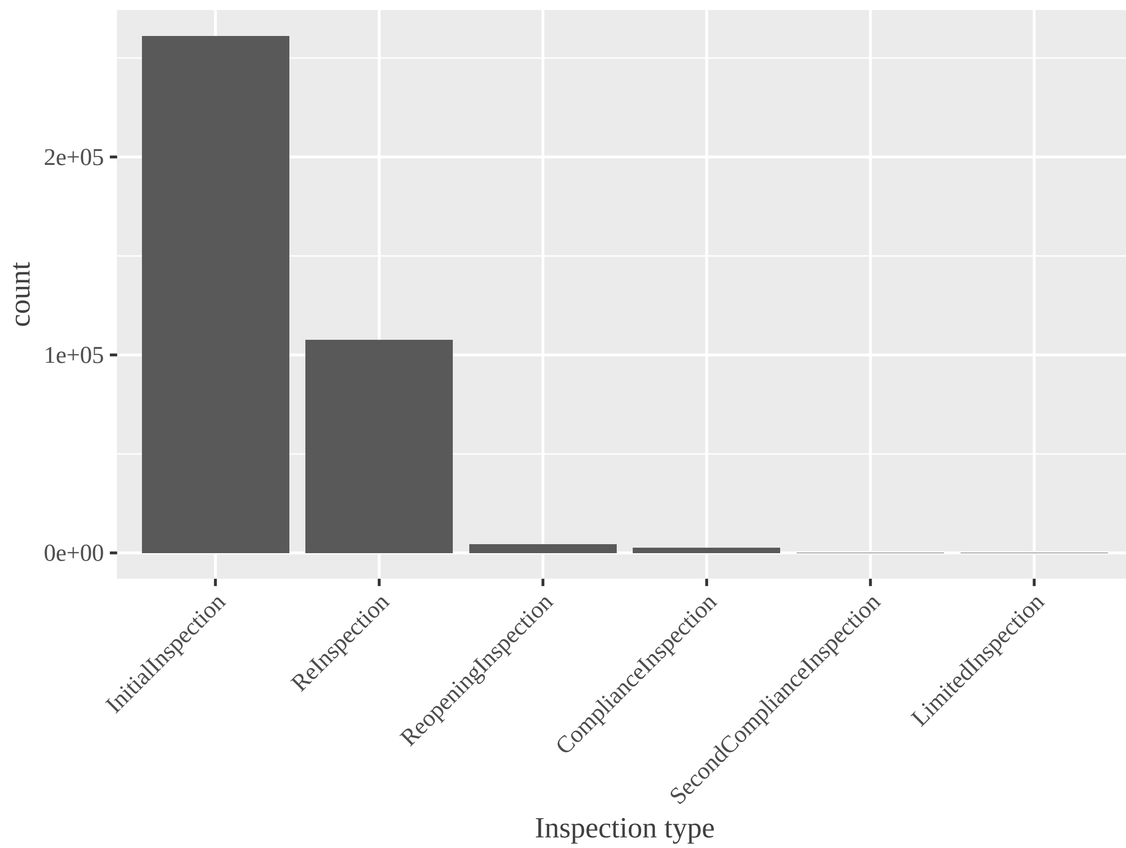
The variable has the following levels:

```r
df_raw %>%
  pull(inspection_type2) %>%
  levels()
```

```
## [1] "ReInspection"                "InitialInspection"
## [3] "ComplianceInspection"        "ReopeningInspection"
## [5] "SecondComplianceInspection" "LimitedInspection"
```

Here is a display of the distribution:

```r
df_raw %>%
  ggplot(aes(x = fct_infreq(inspection_type2))) +
  geom_bar() +
  xlab("Inspection type") +
  theme(text = element_text(
      family = "serif",
      color = "gray25"),
      axis.text.x = element_text(angle = 45, hjust = 1))
```

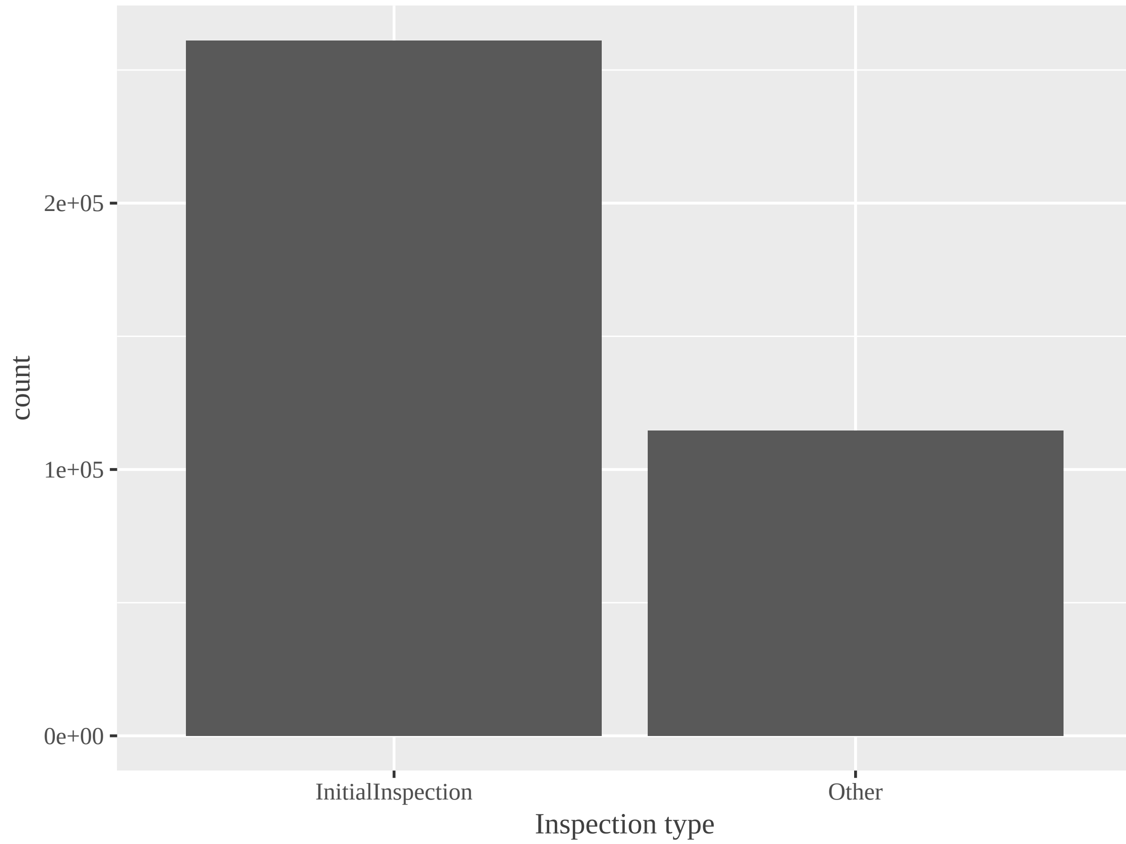


Again, there a are categories that are 'almost empty'. The vast majority of observations are initial and re-inspections: 86306 and 40831, respectively. Hence, we will lump all categories except *Initial Inspection* together:

```
df_raw  <- df_raw %>%
  mutate(inspection_type2 = fct_other(inspection_type2,
                                      keep = "InitialInspection"))
```

Let us check what *inspection_type* looks like now:

```
df_raw  %>%
  ggplot(aes(x = inspection_type2)) +
  geom_bar() +
  xlab("Inspection type") +
  theme(text = element_text(
      family = "serif",
      color = "gray25")
  )
```



2.7 Grade

Next, we investigate the variable ‘grade’.

```
df_raw %>%
  group_by(grade) %>%
  summarise(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 7 x 2
##   grade              n
##   <chr>          <int>
## 1 <NA>          187702
## 2 A             149897
## 3 B              24696
## 4 C               6318
## 5 Z               3510
## 6 Not Yet Graded   1915
## 7 P               1751
```

There is a massive amount of NA’s. We are going to investigate how the number of NAs for ‘grade’ relates to ‘score’ and ‘inspection type.’ More precisely, we know that on initial inspections no grade is issued, if the corresponding score is above 14. It seems plausible that this would lead to a value of NA for ‘grade’.

```
df_raw %>%
  select(inspection_type2, score, grade) %>%
  filter(inspection_type2 == "InitialInspection", score > 14) %>%
  group_by(grade) %>%
  summarise(n = n())
```

```
## # A tibble: 2 x 2
##   grade               n
##   <chr>           <int>
## 1 Not Yet Graded   1548
## 2 <NA>           137410
```

So about 73 % of the missing values for grade are a consequence of a score of 14 or higher. Furthermore, out of the 1915 observations with value 'Not yet graded', 1548 are the result of a high score (above 13) in the initial inspection. We will recode these NA's to "Not Yet Graded" to reduce the number of missing values for 'grade'.

```
# Recode NA's with inspection type = 'InitialInspection' and 'score' > 14 to "Not Yet Graded":
df_raw <- df_raw %>%
  mutate(grade = case_when(
    is.na(.$grade) &
      .$inspection_type2 == "InitialInspection" &
      .$score > 14 ~ "Not Yet Graded",
    TRUE ~ grade
  ))
```

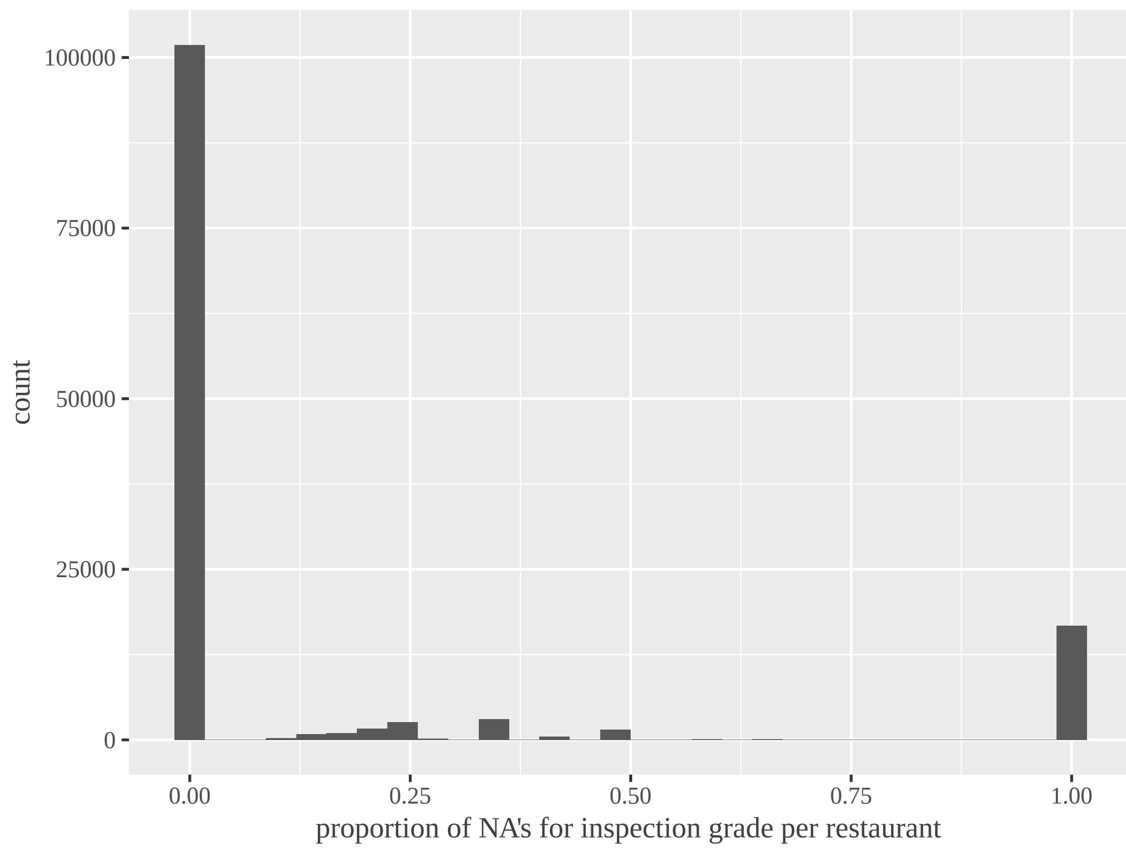
This reduces the number of NA's to 13 %:

```
df_raw %>%
  summarise(prop_na = round(mean(is.na(grade)), 2),
            n_not_graded = sum(grade == "Not Yet Graded", na.rm = T))
```

```
## # A tibble: 1 x 2
##   prop_na n_not_graded
##     <dbl>        <int>
## 1    0.13       139325
```

Here is the distribution of the proportion of inspections with an NA for grade per restaurant:

```
df_raw %>%
  group_by(id, inspection_date) %>%
  summarise(prop_gradeNA = mean(is.na(grade))) %>%
  ggplot(aes(x = prop_gradeNA)) +
  geom_histogram(bins = 30) +
  xlab("proportion of NA's for inspection grade per restaurant") +
  theme(
    text = element_text(
      family = "serif",
      color = "gray25"
    )
  )
```



It seems that for most restaurants there is no inspection with a missing value for grade. On the other hand, there are some restaurants for whom all values for grade are NA.

```
df_raw %>%
  group_by(id, inspection_date) %>%
  summarise(prop_gradeNA = round(mean(is.na(grade)), 3)) %>%
  ungroup() %>%
  group_by(prop_gradeNA) %>%
  summarise(proportion_of_restaurants = round(n()/nrow(.), 3)) %>%
  arrange(desc(proportion_of_restaurants)) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 2
##    prop_gradeNA proportion_of_restaurants
##           <dbl>                     <dbl>
##  1        0                         0.781
##  2        1                         0.128
##  3        0.333                     0.024
##  4        0.25                      0.02
##  5        0.2                       0.012
##  6        0.5                       0.011
##  7        0.167                     0.008
##  8        0.143                     0.004
##  9        0.4                       0.003
## 10        0.125                     0.002
```

2.8 Score

Let us turn to 'score'.

```
df_raw %>%
  pull(score) %>%
  summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   -2.00   11.00   14.00   18.93   24.00  151.00   19514
```

There are 19514 NA's. So let us look at the proportion of inspections that exhibit an NA for 'score'.

```
df_raw %>%
  group_by(id, inspection_date) %>%
  summarise(n_na = sum(is.na(score))) %>%
  ungroup() %>%
  summarise(n_na = round(sum(n_na)/n(), 3))
```

```
## # A tibble: 1 x 1
##    n_na
##   <dbl>
## 1  0.15
```

15 % of all inspections have a missing value for 'score'.

We know that there are 61028 observations that refer to violations that are not scored:

```
df_raw %>%
  group_by(violation_group) %>%
  summarise(n = n()) %>%
  filter(violation_group == "not_scored")
```

```
## # A tibble: 1 x 2
##   violation_group     n
##   <fct>           <int>
## 1 not_scored      61028
```

These observations are expected to have an NA for 'score':

```
df_raw %>%
  select(violation_group, score) %>%
  filter(violation_group == "not_scored") %>%
  summarise(n = n(), n_na_score = sum(is.na(score)), prop_na_score = round(n_na_score/n, 2))
```

```
## # A tibble: 1 x 3
##       n n_na_score prop_na_score
##   <int>      <int>         <dbl>
## 1 61028      16086          0.26
```

Hence, these 16086 observations account for 82 % of all of the 19514 NA's for 'score'. What about the remaining 3428 missing values for 'score'? Let us consider the relationship with violation group:

```r
df_raw %>%
  select(violation_group, score) %>%
  filter(is.na(violation_group)) %>%
  summarise(n = n(), n_na_score = sum(is.na(score)))
```

```
## # A tibble: 1 x 2
##        n n_na_score
##    <int>      <int>
## 1  4921       3428
```

Hence, the rest of the NA's for score are associated with NA's on 'violation_group'. So let us turn to the NA's for the variable 'violation_group'. One cause for an NA here may be that on a given inspection, there was no violation at all. So what is the number of NA's for violation group among observations that do not exhibit a violation?

```r
df_raw %>%
  filter(action == "NoViol") %>%
  summarise(viol_group_NA = sum(is.na(violation_group)))
```

```
## # A tibble: 1 x 1
##   viol_group_NA
##           <int>
## 1             1
```

3 Predicting time until the next inspection

Finally, we turn to the question posed in the beginning, namely, whether it is possible to predict the number of days until the next inspection, using the variables described above.

3.1 Split into training and test data

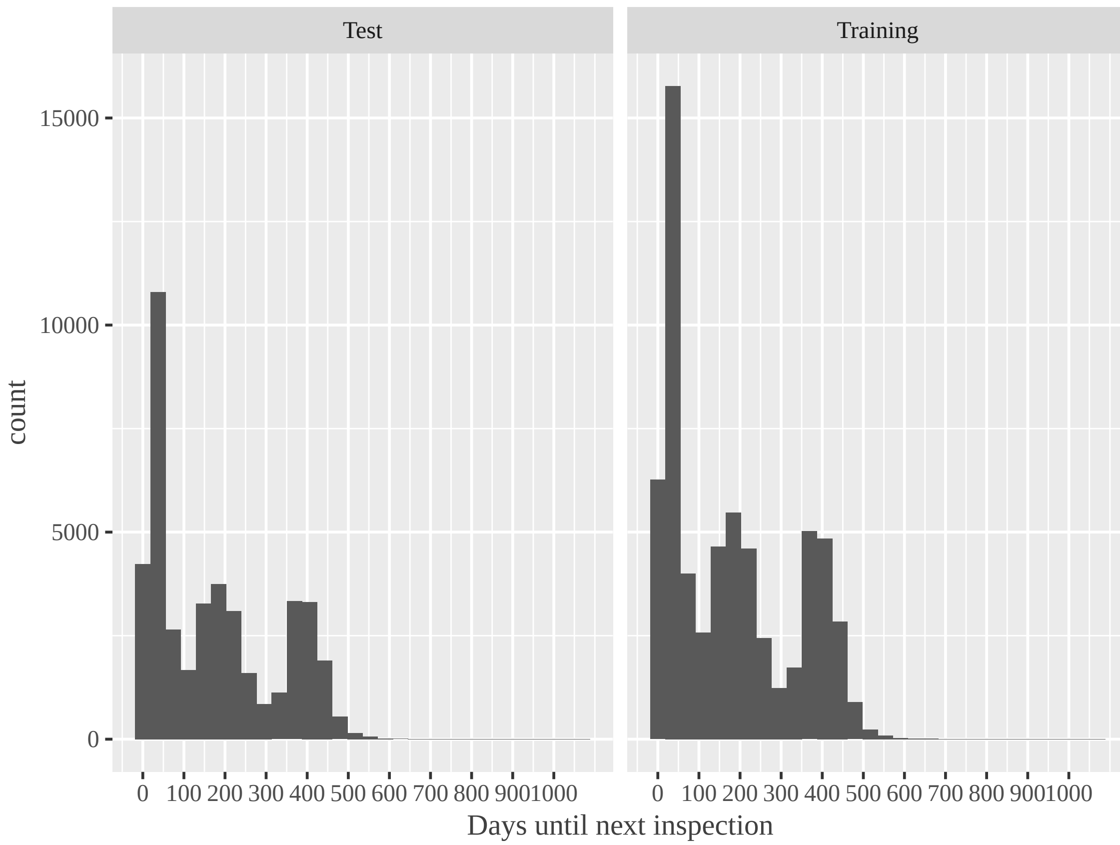
As a first step, we will divide our data set into a training and a test data set, putting 2/3 of the restaurants into the training set.

```r
set.seed(1)
train_ids <- sample(unique(df_raw$id), 0.6*length(unique(df_raw$id)))
test_ids <- setdiff(unique(df_raw$id), train_ids)
```

Next, we have to make sure that the distributions of the target feature in training and test data are reasonably similar:

```r
df_raw %>%
  select(id, inspection_date) %>%
  group_by(id, inspection_date) %>%
  arrange(inspection_date) %>%
  summarise() %>%
  mutate(difference = lead(inspection_date) - inspection_date) %>%
  filter(!is.na(difference)) %>%
  mutate(group = ifelse(id %in% train_ids, "Training", "Test")) %>%
```

```r
ggplot(aes(as.numeric(difference))) +
geom_histogram(bins = 30) +
scale_x_continuous(breaks = seq(0, 1000, by = 100)) +
xlab("Days until next inspection") +
facet_grid(. ~ group) +
theme(
  text = element_text(
    family = "serif",
    color = "gray25"
  )
)
```



The distribution of the target seems reasonably similar across training and test data.

3.2 Features

Next, we will construct features using the variables in the data set. The result of that construction will be a table, where each row corresponds to an inspection for a given restaurant, containing

- the grade of the inspection (*grade*),
- an indicator whether the row corresponds to an initial inspection(*inspection_type2*),
- the score of the inspection (*score*),
- the cuisine of the restaurant (*cuisine_descr*),
- the number of violations in the different violation_groups (*viol_*), and
- the number of critical flags,

- the number of days until the next inspection (*days_until_next*) which is to be used only in the training data

```r
## Create data set with features for inspection type, score, grade and number of violations:

## Make feature set:

make_features_raw <- function ( df ) {

  df %>%
    select ( id, inspection_date, inspection_type2, score,
             grade, cuisine_descr ) %>%
    group_by ( id, inspection_date ) %>%
    arrange ( id, inspection_date ) %>%
    summarise_at ( vars(c("grade","inspection_type2", "score", "cuisine_descr")),
                   .funs = list(first) ) %>%
    left_join (
      df %>%
        select ( id, inspection_date, starts_with("viol_"), starts_with("critical") ) %>%
        mutate( critical_flag = case_when( critical_flag == "Critical" ~ 1,
                                           TRUE ~ 0 ) ) %>%
        group_by ( id, inspection_date ) %>%
        arrange ( id, inspection_date ) %>%
        summarise_at ( vars(starts_with( "viol" ), starts_with( "critical" ) ),
                       ~ sum( ., na.rm = TRUE ) ),
      by = c ( "id", "inspection_date" ) ) %>%
    ungroup()

}

# Add target feature

add_target_feature <- function(df) {

  df %>%
    group_by(id) %>%
    arrange(id, inspection_date) %>%
    mutate (
      days_until_next = lead ( inspection_date ) - inspection_date
      ) %>%
    filter(!is.na(days_until_next)) %>%
    ungroup() %>%
    mutate (
      days_until_next_categ =
        case_when (
          between (
            as.numeric ( days_until_next ), 0, 100
            ) ~ "within 3 months",
          between (
            as.numeric ( days_until_next ), 101, 300
            ) ~ "within 3 to 10 months",
          TRUE ~ "in more than 10 months" ),
      days_until_next_categ =
        as_factor ( days_until_next_categ ) ) %>%
```

```
    ungroup()

}
```

As we have seen in the exploration, there are a number of NA's for the features *grade* and *score*. Because the prediction algorithms that we will employ require that there be no missing values, we will impute these values. To impute *score* (*grade*), we will train a regression tree (classification tree) to predict *score* (*grade*) using all the other features on all rows where *score* (*grade*) is not missing.

```
# Impute features

impute_features <- function ( df ) {

  df %>%
    filter(is.na(score)) %>%
    mutate(score =
             predict(df %>%
                       filter ( !is.na ( score ) ) %>%
                       select (
                         score, grade, inspection_type2, cuisine_descr,
                         starts_with ( "viol_" ), critical_flag ) %>%
                       rpart ( score ~ ., data = ., method = "anova" ), .)
           ) %>%
    bind_rows(df %>% filter(!is.na(score))) %>%
    filter(is.na(grade)) %>%
    mutate(grade =
             predict(df %>%
                       filter ( !is.na ( grade ) ) %>%
                       select (
                         score, grade, inspection_type2, cuisine_descr,
                         starts_with ( "viol_" ), critical_flag ) %>%
                       rpart ( grade ~ ., data = ., method = "class" ), .,
                     type = "class"),
           grade =
             as.character(grade)) %>%
    bind_rows(df %>%
                filter(!is.na(grade)))

}
```

Furthermore, we add in *days_until_next*.

```
df_features <- df_raw %>%
  make_features_raw() %>%
  impute_features() %>%
  add_target_feature()

df_train <- df_features %>%
  filter(id %in% train_ids)

df_test <- df_features %>%
  filter(id %in% test_ids)
```

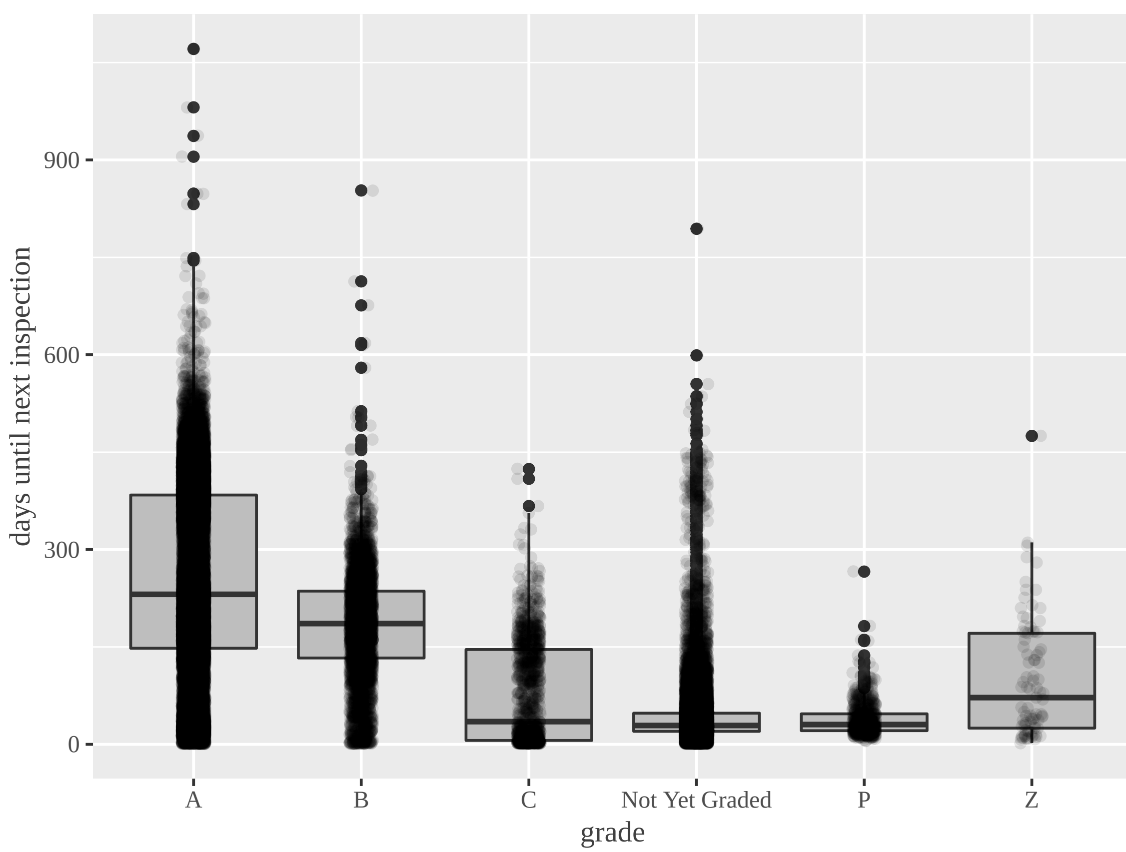
For now, the test data will be removed from the session. Later, we will use it to evaluate the final model.

```
write_rds(df_test, "df_test.rds")
rm(df_test)
```

3.3 Feature Exploration

Let us first explore a couple of bivariate relationships between the predictors and the target feature. We will begin with the relationship between *grade* and *days until the next inspection*.

```
df_train %>%
  ggplot ( aes ( x = grade, y = as.numeric ( days_until_next ) ) ) +
  geom_boxplot ( fill = "grey") +
  geom_jitter(alpha = .1, width = .07) +
  xlab("grade") +
  ylab("days until next inspection") +
  theme(
    text = element_text(
      family = "serif",
      color = "gray25"
    )
  )
```



As we can see, the median number of days until the next inspection decreases when moving from A to C. This is not suprising, since a good grade on the initial inspection reduces the likelihood of being subjected to a re-inspection the same inspection cycle.

Let us move on to the relationship between violation types and the target feature:

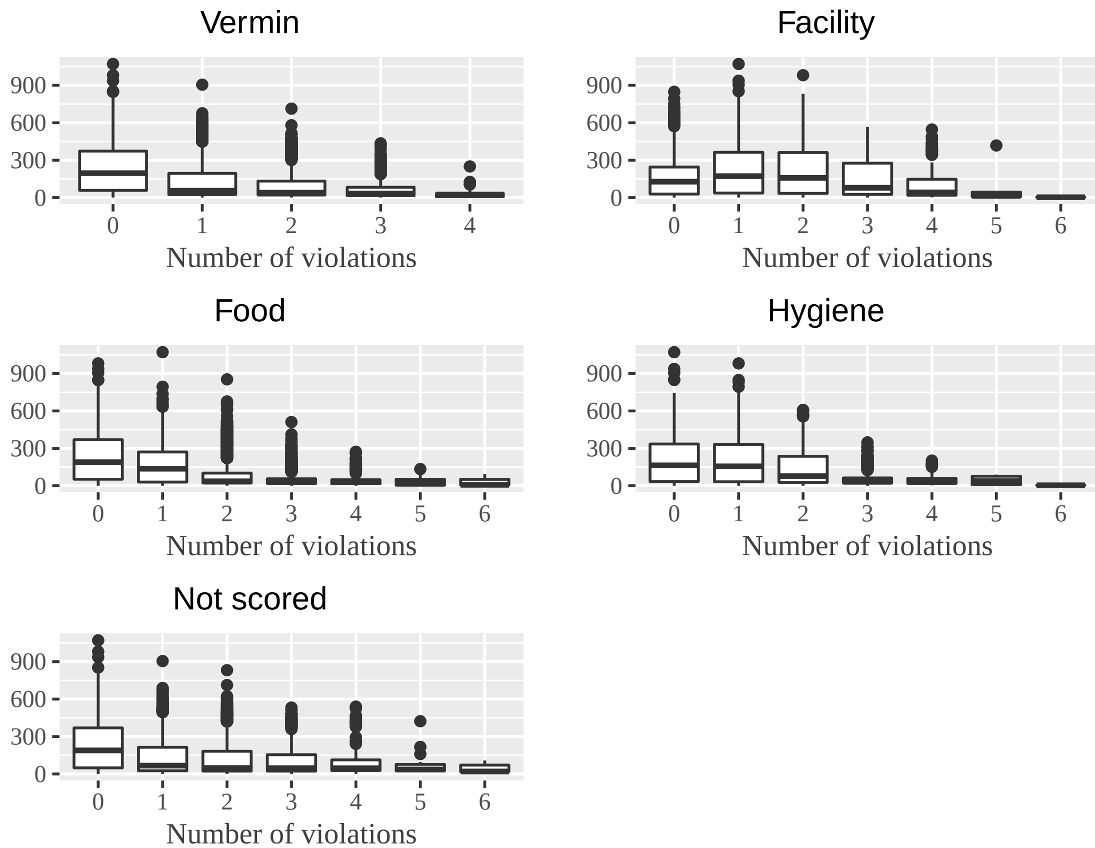
```
# Plotting function
plot_vs_days <- function(z) {

  df_train %>%
    ggplot(aes(x = as.factor(z), y = as.numeric(days_until_next))) +
    geom_boxplot() +
    xlab("Number of violations") +
    ylab("") +
    theme(
      text = element_text(
        family = "serif",
        color = "gray25"
        )
      )

}

# Plot objects
plot_lst <- df_train %>%
  select(starts_with("viol_")) %>%
  map(plot_vs_days)

# Arrange plots in grid
grid.arrange(
  arrangeGrob(plot_lst[[1]], top = "Vermin") ,
  arrangeGrob(plot_lst[[2]], top = "Facility"),
  arrangeGrob(plot_lst[[3]], top = "Food"),
  arrangeGrob(plot_lst[[4]], top = "Hygiene"),
  arrangeGrob(plot_lst[[5]], top = "Not scored"),
  ncol = 2)
```




```r
# Remove auxiliary objects
rm(plot_lst)
rm(plot_vs_days)
```

Here also, as expected, the median number of days until the next inspection decreases as the number of violations per inspection increases.

Next, we shall see whether it is possible to predict the time until the next inspection.