HarvardX PH125.9x - My Own Project: Bike sharing prediction

Stefania_Ravazzini

2021/11/14

1. Executive Summary

The scope of the analysis is Bike-sharing prediction. The dataset used for the analysis comes from the following link of UCI Machine Learning Repository:
https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset
[License: Fanaee-T, Hadi, and Gama, Joao, "Event labeling combining ensemble detectors and background knowledge", Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg, doi:10.1007/s13748-013-0040-3.]

The dataset contains 2011 and 2012 bike sharing counts, aggregated on hourly basis, from Capital Bikeshare system, Washington D.C., USA. Counts of bike rentals are declined together with information on the type of day they occurred (working day / holiday etc), as well as the weather conditions that were registered in that specific day and hour.

The idea behind this analysis is that a regression task can be performed with this data. Historical logs can be analyzed to predict bike rental hourly count (= the dependent variable) based on the environmental and seasonal settings (= the independent or explanatory variables).

As already mentioned, the dataset has one single row for each hourly count of bike rentals. The columns - or "features" - are as follows:
- `instant` : record index
- `dteday` : date
- `season` : season (1: Winter, 2: Spring, 3: Summer, 4: Fall)
- `yr` : year (0: 2011, 1: 2012)
- `mnth` : month (1 to 12)
- `hr` : hour (0 to 23)
- `holiday` : weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
- `weekday` : day of the week
- `workingday` : if day is neither weekend nor holiday it is 1, otherwise it is 0
- `weathersit` : weather situation (1: Clear, Few clouds, Partly cloudy, 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist, 3: Light snow, Light rain + Thunderstorm + Scattered clouds, Light rain + Scattered clouds, 4: Heavy rain + Ice pallets + Thunderstorm + Mist, Snow + Fog)
- `temp` : normalized temperature in Celsius. The values are divided to 41 (max)
- `atemp`: normalized feeling temperature in Celsius. The values are divided to 50 (max)
- `hum`: normalized humidity. The values are divided to 100 (max)
- `windspeed`: normalized wind speed. The values are divided to 67 (max)
- `casual`: count of casual users
- `registered`: count of registered users
- `cnt`: count of total rental bikes including both casual and registered

After some steps of data cleaning and data exploration / visualization, the dataset is splitted into two separate datasets, i.e. the **training set** and the **test set**. According to machine learning standards, the development and training of the algorithms is made on the training set, while the final RMSE is evaluated

on the test set. The model with the lowest RMSE achievable is finally chosen as the best model. RMSE is a metric of "goodness of fit": the lowest the RMSE, the smallest the error of the model. RMSE is calculated as the square root of the average through all the observations of the difference squared between actual bike-rental counts and predicted bike-rental counts. In the last chapter of the analysis, some future work is suggested.

2. Analysis

2.a Load the dataset

The zip file containing the data is downloaded; the csv file contained in the zip file is extracted and loaded into R:

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip"
path <- file.path("~", "Bike-Sharing-Dataset.zip")
download.file(url, path)
dataset <- read.table(unzip(path, "hour.csv"),
                      header = TRUE,
                      sep = ",",
                      stringsAsFactors = FALSE)
```

2.b Data cleaning

Let's have a quick look at the first rows of the dataset and check info on its structure:

```
head(dataset)
```

```
##   instant     dteday season yr mnth hr holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1  0       0       6          0          1
## 2       2 2011-01-01      1  0    1  1       0       6          0          1
## 3       3 2011-01-01      1  0    1  2       0       6          0          1
## 4       4 2011-01-01      1  0    1  3       0       6          0          1
## 5       5 2011-01-01      1  0    1  4       0       6          0          1
## 6       6 2011-01-01      1  0    1  5       0       6          0          2
##   temp  atemp  hum windspeed casual registered cnt
## 1 0.24 0.2879 0.81    0.0000      3         13  16
## 2 0.22 0.2727 0.80    0.0000      8         32  40
## 3 0.22 0.2727 0.80    0.0000      5         27  32
## 4 0.24 0.2879 0.75    0.0000      3         10  13
## 5 0.24 0.2879 0.75    0.0000      0          1   1
## 6 0.24 0.2576 0.75    0.0896      0          1   1
```

```
str(dataset)
```

```
## 'data.frame':    17379 obs. of  17 variables:
##  $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ dteday    : chr  "2011-01-01" "2011-01-01" "2011-01-01" "2011-01-01" ...
##  $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hr        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday   : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weathersit: int  1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  0.24 0.22 0.22 0.24 0.24 0.24 0.22 0.2 0.24 0.32 ...
##  $ atemp     : num  0.288 0.273 0.273 0.288 0.288 ...
##  $ hum       : num  0.81 0.8 0.8 0.75 0.75 0.75 0.8 0.86 0.75 0.76 ...
```

```
##  $ windspeed : num  0 0 0 0 0 0.0896 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ cnt       : int  16 40 32 13 1 1 2 3 8 14 ...
```

First of all, we should see if some missing value needs to be treated:

```
sum(is.na(dataset))
```

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

No missing value is detected.
All the same, some adjustments need to be made:
1) We have some format issues to fix: by looking at the structure, we see that the date column `dteday` was set as "character", so we should convert it into "date". The `holiday` and `workingday` boolean columns were imported as "integers", so we should create new columns `holiday_log` and `workingday_log` where we convert them into "logical".
2) We have some additions to do, for better communication purpose in the next session of data exploration. The features `season`, `yr`, `mnth`, `weekday` and `weathersit` should be translated into dedicated columns containing the description of their modalities as factors, so we create new columns `season_fct`, `yr_fct`, `mnth_fct`, `weekday_fct` and `weathersit_fct` for this purpose. After this, we sort the factors with the appropriate order.

```
dataset <- dataset %>%
           mutate(dteday = as.Date(dteday),
                  season_fct = as.factor(case_when(season == 1 ~ "Winter",
                                                   season == 2 ~ "Spring",
                                                   season == 3 ~ "Summer",
                                                   season == 4 ~ "Fall"
                                                      )),
                  yr_fct = as.factor(ifelse(yr == 0, "2011", "2012")),
                  mnth_fct = as.factor(case_when(mnth == 1  ~ "Jan",
                                                 mnth == 2  ~ "Feb",
                                                 mnth == 3  ~ "Mar",
                                                 mnth == 4  ~ "Apr",
                                                 mnth == 5  ~ "May",
                                                 mnth == 6  ~ "Jun",
                                                 mnth == 7  ~ "Jul",
                                                 mnth == 8  ~ "Aug",
                                                 mnth == 9  ~ "Sep",
                                                 mnth == 10 ~ "Oct",
                                                 mnth == 11 ~ "Nov",
                                                 mnth == 12 ~ "Dec"
                                               )),
                  holiday_log = as.logical(holiday),
                  weekday_fct = as.factor(case_when(weekday == 0  ~ "Sun",
                                                    weekday == 1  ~ "Mon",
                                                    weekday == 2  ~ "Tue",
                                                    weekday == 3  ~ "Wed",
                                                    weekday == 4  ~ "Thu",
                                                    weekday == 5  ~ "Fri",
                                                    weekday == 6  ~ "Sat"
                                                  )),
```

```
                  workingday_log = as.logical(workingday),
                  weathersit_fct = as.factor(case_when(weathersit == 1  ~ "Clear",
                                                       weathersit == 2  ~ "Mist",
                                                       weathersit == 3  ~ "Light Rain",
                                                       weathersit == 4  ~ "Heavy Rain/Snow"
                                                ))
                  )

dataset <- dataset %>%
           mutate(season_fct = factor(season_fct, levels =
                                        c("Winter", "Spring", "Summer", "Fall")),
                  mnth_fct = factor(mnth_fct, levels =
                                      c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")),
                  weekday_fct = factor(weekday_fct, levels =
                                         c("Sun", "Mon", "Tue",
                                           "Wed", "Thu", "Fri", "Sat")),
                  weathersit_fct = factor(weathersit_fct, levels =
                                            c("Clear", "Mist", "Light Rain", "Heavy Rain/Snow")))
```

Some data consistency test is appropriate. We perform the following tests:
1) The number of rows where "the variable `yr_fct` is equal to the year of the variable `dteday`" should match the number of rows of the whole dataset;
2) The number of rows where "the variable `mnth` is equal to the month of the variable `dteday`" should match the number of rows of the whole dataset;
3) The number of rows where "the variable `weekday` is equal to the weekday of the variable `dteday` adjusted by 1" should match the number of rows of the whole dataset;
4) The number of rows where "the variable `season` contains the exact season where the `dteday` lies" should match the number of rows of the whole dataset;
5) The number of rows where "the variable `cnt` contains the sum of the two variables `casual` + `registered`" should match the number of rows of the whole dataset.
The tests are passed if the resulting test-table is all zero:

```
nrow <- nrow(dataset)

dataset_test <- dataset %>%
  summarize(yr_check = nrow - sum(ifelse(yr_fct == "2011", 2011, 2012) == year(dteday)),
            mnth_check = nrow - sum(mnth == month(dteday)),
            weekday_check = nrow - sum(weekday == wday(dteday)-1),
            season_check = sum(season - case_when(
                                        dteday <= ymd("2011-03-20") ~ 1,
                                        dteday > ymd("2011-12-20") &
                                        dteday <= ymd("2012-03-20") ~ 1,
                                        dteday > ymd("2012-12-20") ~ 1, # --> Winter
                                        dteday > ymd("2011-03-20") &
                                        dteday <= ymd("2011-06-20") ~ 2,
                                        dteday > ymd("2012-03-20") &
                                        dteday <= ymd("2012-06-20") ~ 2, # --> Spring
                                        dteday > ymd("2011-06-20") &
                                        dteday <= ymd("2011-09-22") ~ 3,
                                        dteday > ymd("2012-06-20") &
                                        dteday <= ymd("2012-09-22") ~ 3, # --> Summer
                                        TRUE ~ 4 # --> Fall
```

```
                                        )),
            cnt_check = sum(cnt - (casual + registered))
  )

dataset_test
```

```
##   yr_check mnth_check weekday_check season_check cnt_check
## 1        0          0             0            0         0
```

The data consistency tests are all passed.

2.c Data exploration

We will now explore the variables through quick statistics of range and counts.

```
kable(dataset %>%
  summarize(min_dteday = min(dteday),
            max_dteday = max(dteday)), caption = "Date - range")
```

Table 1: Date - range

min_dteday	max_dteday
2011-01-01	2012-12-31

Dates range from 2011-01-01 to 2012-12-31.

```
kable(table(dataset$yr_fct, dataset$season_fct), caption = "Season - row counts")
```

Table 2: Season - row counts

	Winter	Spring	Summer	Fall
2011	2068	2203	2240	2134
2012	2174	2206	2256	2098

Observations are equally distributed between seasons, by each year.

```
kable(table(dataset$yr_fct, dataset$mnth_fct), caption = "Month - row counts")
```

Table 3: Month - row counts

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2011	688	649	730	719	744	720	744	731	717	743	719	741
2012	741	692	743	718	744	720	744	744	720	708	718	742

Observations are quite equally distributed between months, by each year.

```
kable(table(dataset$yr_fct, dataset$weekday_fct), caption = "Weekday - row counts")
```

Table 4: Weekday - row counts

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
2011	1231	1235	1222	1229	1225	1239	1264
2012	1271	1244	1231	1246	1246	1248	1248

Observations are quite equally distributed between weekdays, by each year.

```
kable(table(dataset$yr_fct, dataset$holiday_log), caption = "Holiday - row counts")
```

Table 5: Holiday - row counts

	FALSE	TRUE
2011	8406	239
2012	8473	261

Less than 3% of observations are related to holidays.

```
kable(table(dataset$yr_fct, dataset$workingday_log), caption = "Workingday - row counts")
```

Table 6: Workingday - row counts

	FALSE	TRUE
2011	2734	5911
2012	2780	5954

More than 2/3 of observations are related to working days.

```
kable(table(dataset$yr_fct, dataset$hr), caption = "Hour - row counts")
```

Table 7: Hour - row counts

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2011	361	360	352	342	337	353	361	363	363	363	363	363	364	364	364	364	365	365	363	363	363	363	363	363
2012	365	364	363	355	360	364	364	364	364	364	364	364	364	365	365	365	365	365	365	365	365	365	365	365

Observations are quite equally distributed between hours, by each year.

```r
kable(table(dataset$yr_fct, dataset$weathersit_fct), caption = "Weather situation - row counts")
```

Table 8: Weather situation - row counts

	Clear	Mist	Light Rain	Heavy Rain/Snow
2011	5645	2218	781	1
2012	5768	2326	638	2

2011 and 2012 have quite the same number of observations, split more or less by the same weather condition.

```r
kable(dataset %>%
  summarize(min_temp = min(temp), max_temp = max(temp), mean_temp = mean(temp),
            sd_temp = sd(temp)), caption = "Temperature - stats")
```

Table 9: Temperature - stats

min_temp	max_temp	mean_temp	sd_temp
0.02	1	0.4969872	0.1925561

Temperatures range between 0.02 and 1, proof that they are normalized, as declared in source zip file.

```r
kable(dataset %>%
  summarize(min_atemp = min(atemp), max_atemp = max(atemp), mean_atemp = mean(atemp),
            sd_atemp = sd(atemp)), caption = "Feeling Temperature - stats")
```

Table 10: Feeling Temperature - stats

min_atemp	max_atemp	mean_atemp	sd_atemp
0	1	0.4757751	0.1718502

Feeling temperatures range between 0 and 1, proof that they are normalized, as declared in source zip file.

```r
kable(dataset %>%
  summarize(min_hum = min(hum), max_hum = max(hum), mean_hum = mean(hum),
            sd_hum = sd(hum)), caption = "Humidity - stats")
```

Table 11: Humidity - stats

min_hum	max_hum	mean_hum	sd_hum
0	1	0.6272288	0.1929298

Humidity range between 0 and 1, proof that it is normalized, as declared in source zip file.

```
kable(dataset %>%
  summarize(min_windspeed = min(windspeed), max_windspeed = max(windspeed),
            mean_windspeed = mean(windspeed),
            sd_windspeed = sd(windspeed)), caption = "Wind speed - stats")
```

Table 12: Wind speed - stats

min_windspeed	max_windspeed	mean_windspeed	sd_windspeed
0	0.8507	0.1900976	0.1223402

Wind speed range between 0 and 0.8507, proof that it is normalized, as declared in source zip file.

```
kable(dataset %>%
  summarize(min_cnt = min(cnt), max_cnt = max(cnt), mean_cnt = mean(cnt),
            sd_cnt = sd(cnt)), caption = "Bike rentals for All users - stats")
```

Table 13: Bike rentals for All users - stats

min_cnt	max_cnt	mean_cnt	sd_cnt
1	977	189.4631	181.3876

Hourly bike rental observations of users range between 1 and a peak of 977, mean value is 189 and variability is high.

```
kable(dataset %>%
  summarize(min_registered = min(registered), max_registered = max(registered),
            mean_registered = mean(registered),
            sd_registered = sd(registered)), caption = "Bike rentals for Registered users - stats")
```

Table 14: Bike rentals for Registered users - stats

min_registered	max_registered	mean_registered	sd_registered
0	886	153.7869	151.3573

Hourly bike rental observations of registered users range between 0 and a peak of 886, mean value is 154 and variability is high.

```
kable(dataset %>%
  summarize(min_casual = min(casual),  max_casual = max(casual),  mean_casual = mean(casual),
            sd_casual = sd(casual)), caption = "Bike rentals for Casual users - stats")
```

Table 15: Bike rentals for Casual users - stats

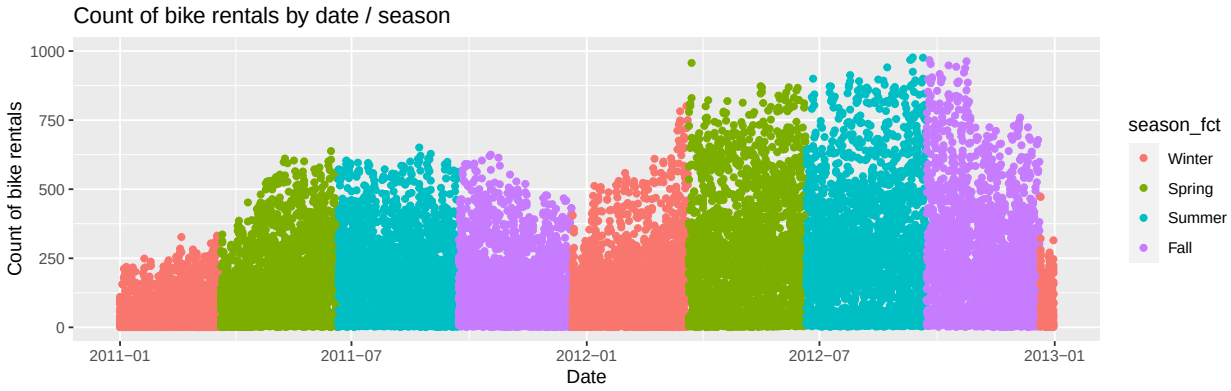
min_casual	max_casual	mean_casual	sd_casual
0	367	35.67622	49.30503

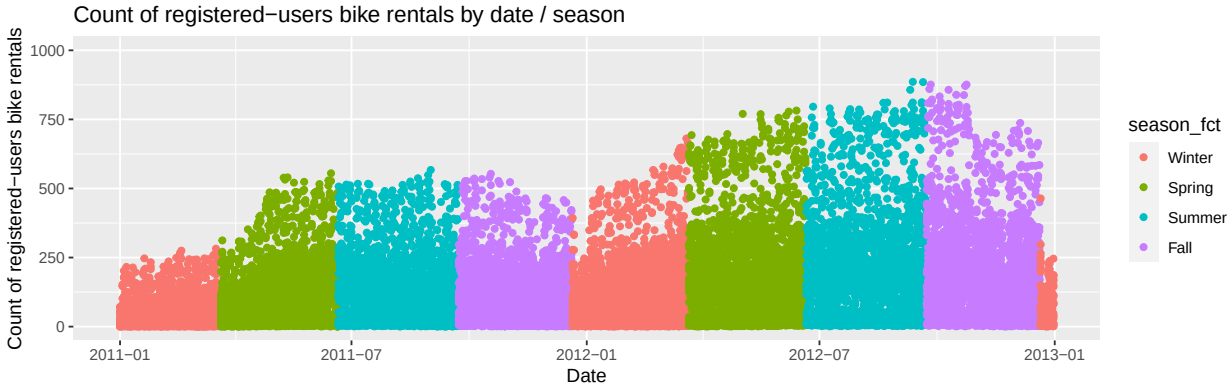
Hourly bike rental observations of casual users range between 0 and a peak of 367, mean value is 36 and variability is high.

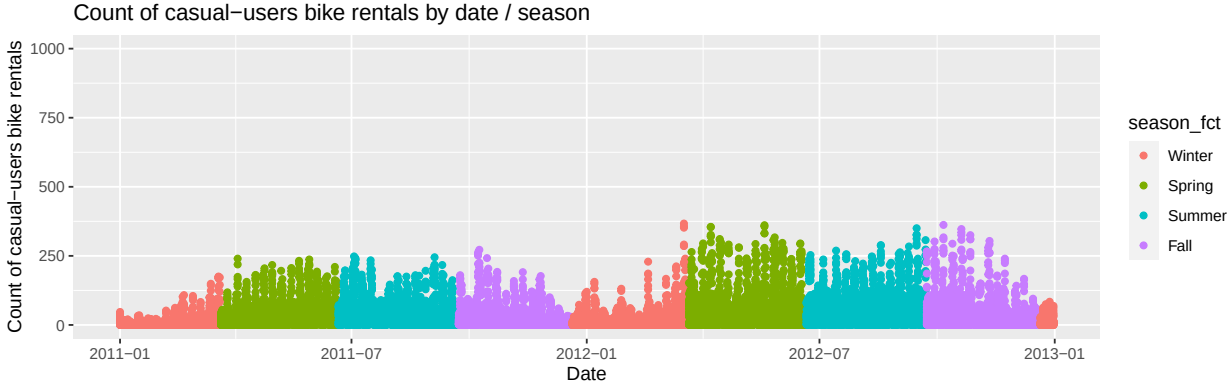
2.d Data visualization

In the following plots, we visualize bike rentals by seasonal and environmental settings. All plots are different ways of displaying the target dependent variable, `cnt`, as well as the two variables `registered` and `casual`, because - among other things - we want to explore how the bike-rental counts differ between the two types of users, registered or casual. `R code` will be displayed only for `cnt` plots and it will be hidden for `registered` and `casual`.

```
ggplot(data = dataset, aes(x = dteday, y = cnt, color = season_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Date",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by date / season")
```

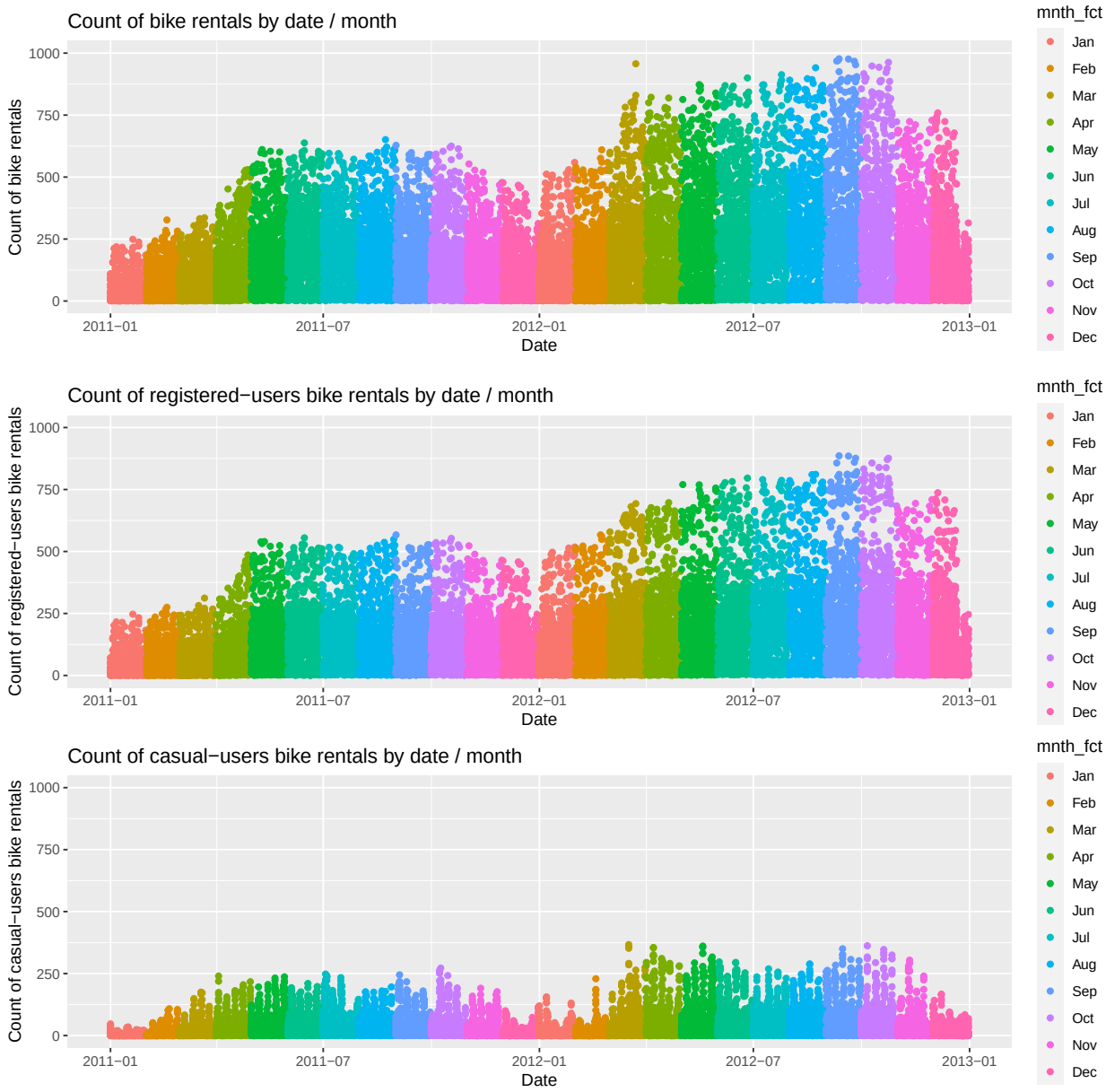






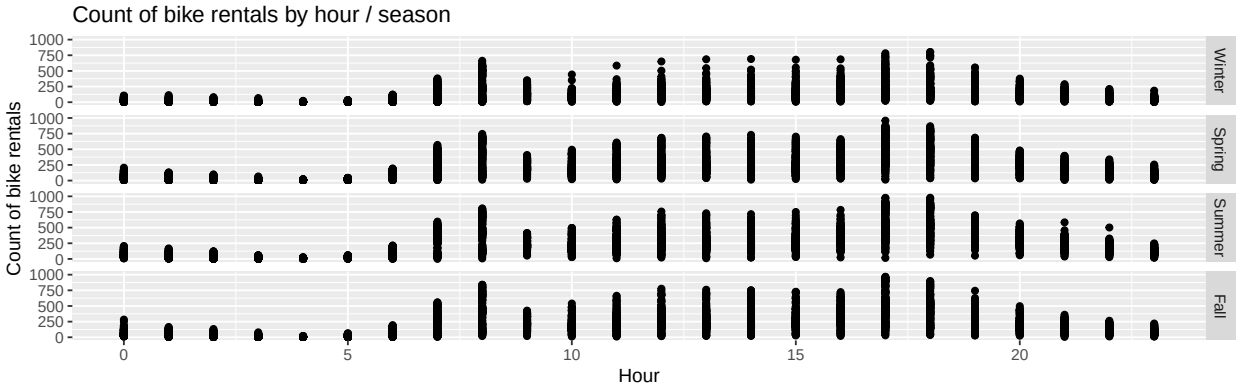
Count of bike-rentals increased in 2012 over 2011. As we would expect, counts in both years decrease during winter and increase when the season is hotter. Registered users explain the majority of the bike-rental counts and they are keen to using the bike even in cold months. Casual users use the bike rental service quite exclusively in hot seasons.

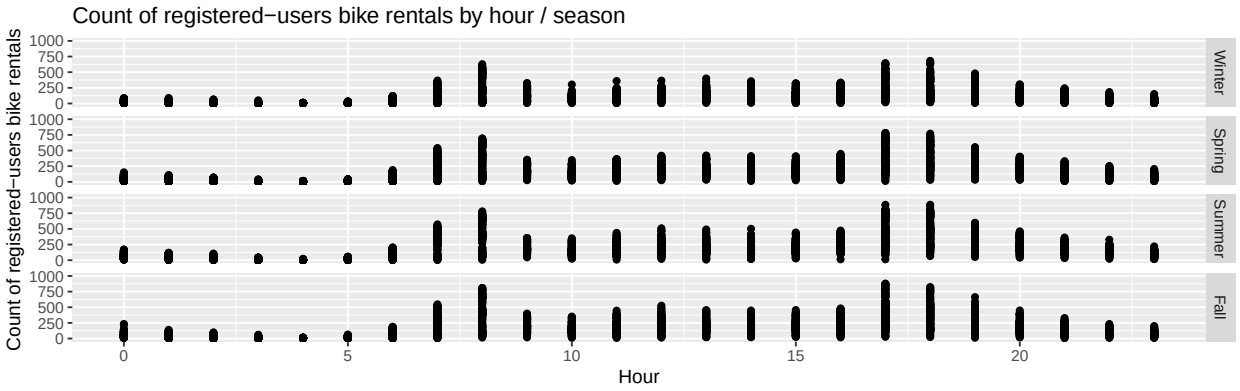
```
ggplot(data = dataset, aes(x = dteday, y = cnt, color = mnth_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Date",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by date / month")
```

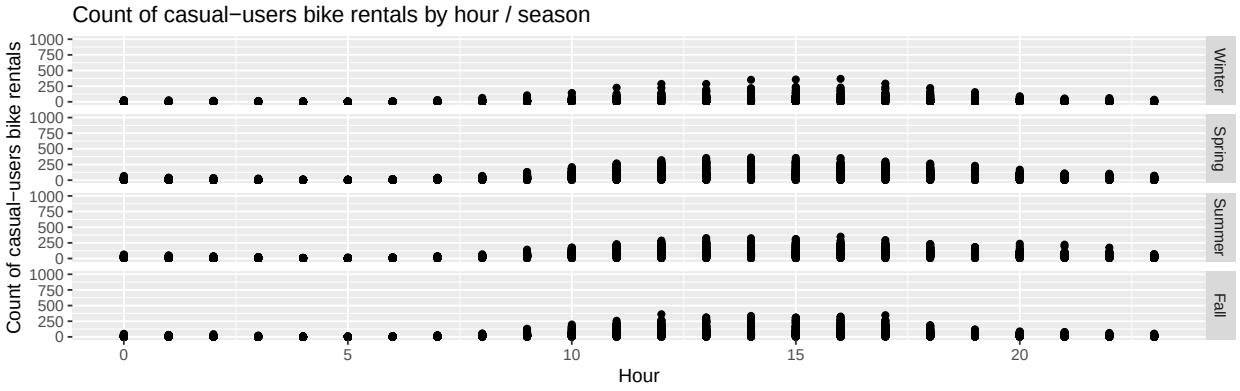


December and January are the months where the bike-rental service is less used, while September is the month where the counts reach the peak.

```r
ggplot(data = dataset, aes(x = hr, y = cnt)) +
  geom_point() +
  ylim(c(0, 1000)) +
  facet_grid(rows = vars(season_fct)) +
  labs(x = "Hour",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by hour / season")
```

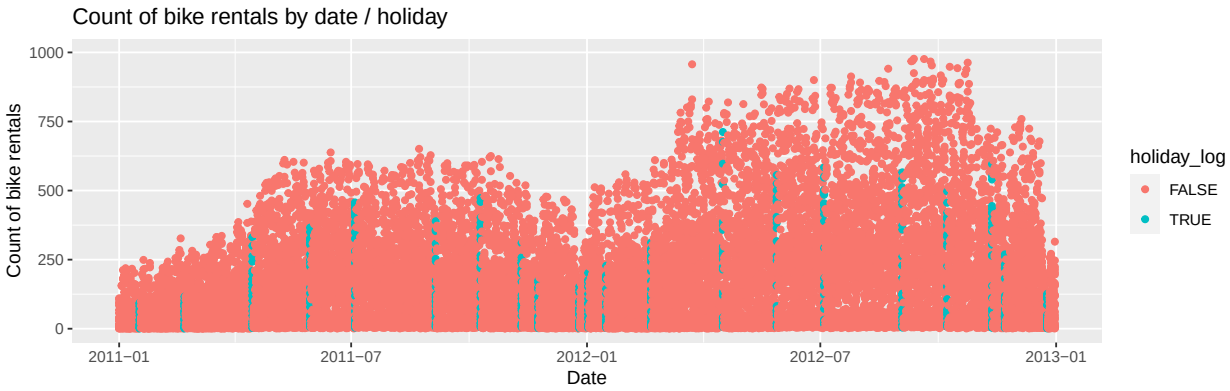


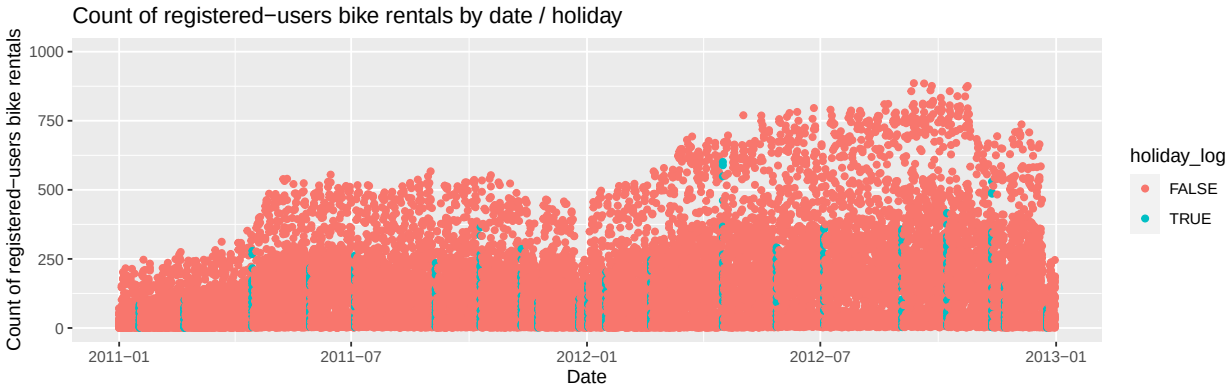


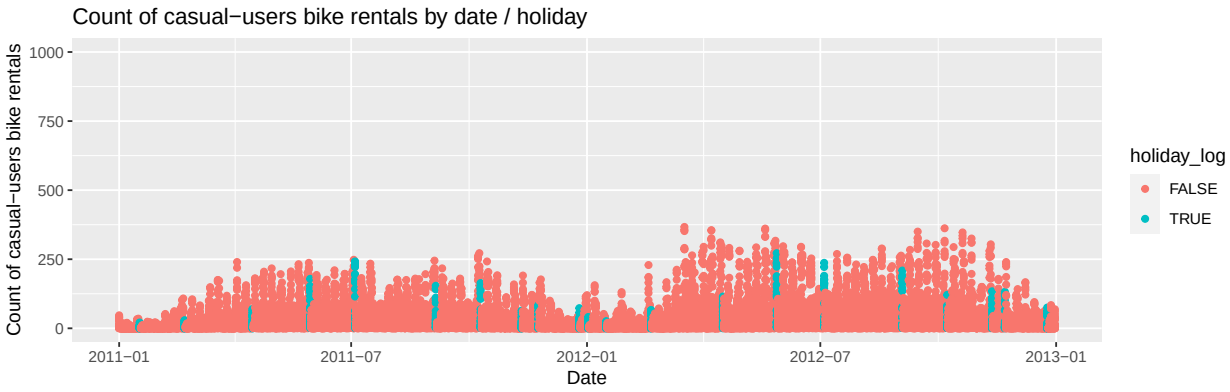


During night hours, bike-rentals are not very frequent. The peak of total counts is reached during 8 a.m., 5 p.m. and 6 p.m., this evidence is true across all seasons. Registered users display this trend very clearly, suggesting that they are commuters. Casual users prefer to rent a bike during the afternoon.

```r
ggplot(data = dataset, aes(x = dteday, y = cnt, color = holiday_log)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Date",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by date / holiday")
```

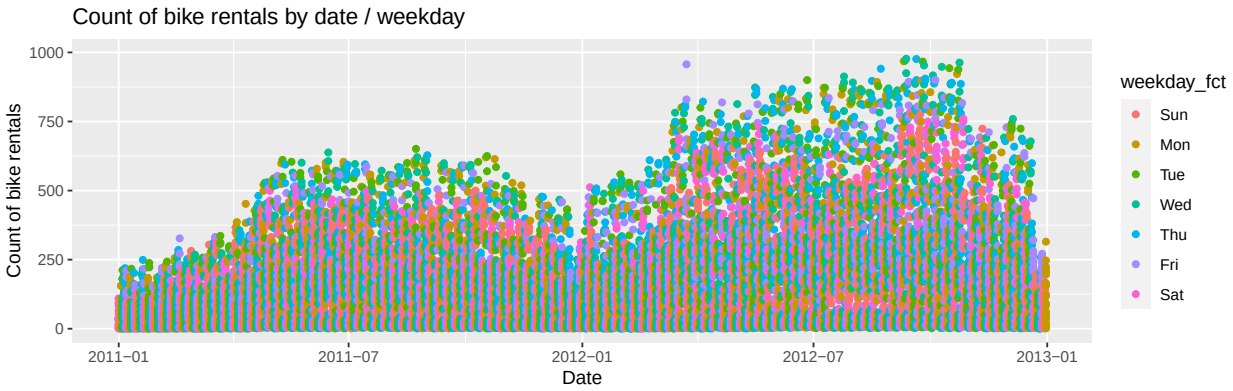


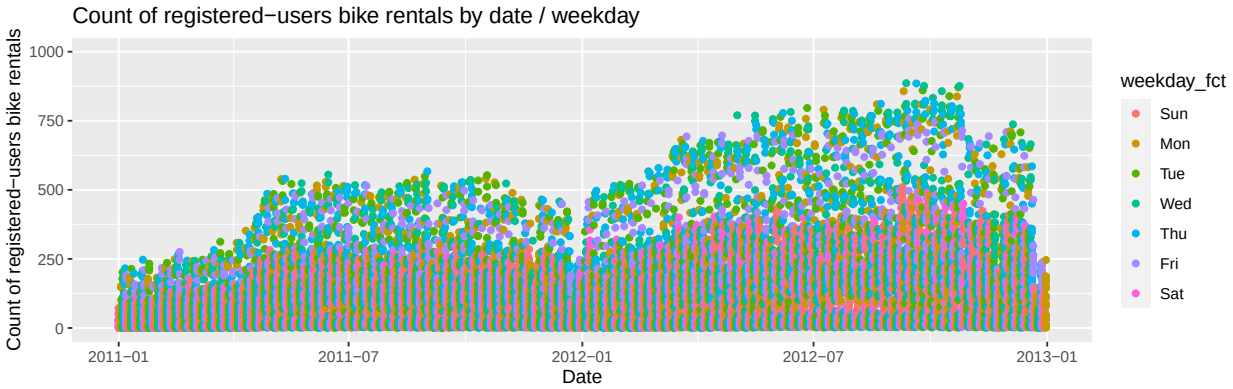


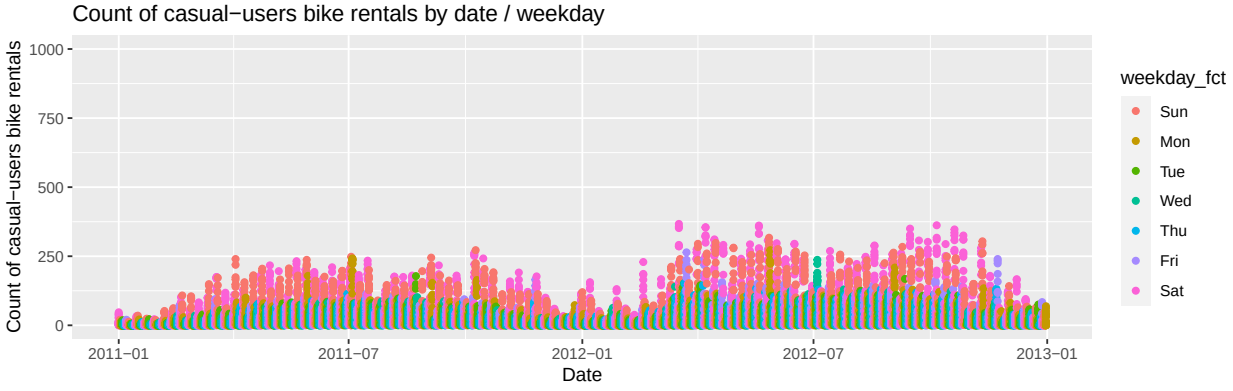


Holidays are very few if compared to the other days of the calendar, so this variable doesn't seem to have too much effect. During holidays, a peak on bike-rental counts of casual users can be observed, suggesting that during spare time these users like to ride the bycicle.

```
ggplot(data = dataset, aes(x = dteday, y = cnt, color = weekday_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Date",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by date / weekday")
```

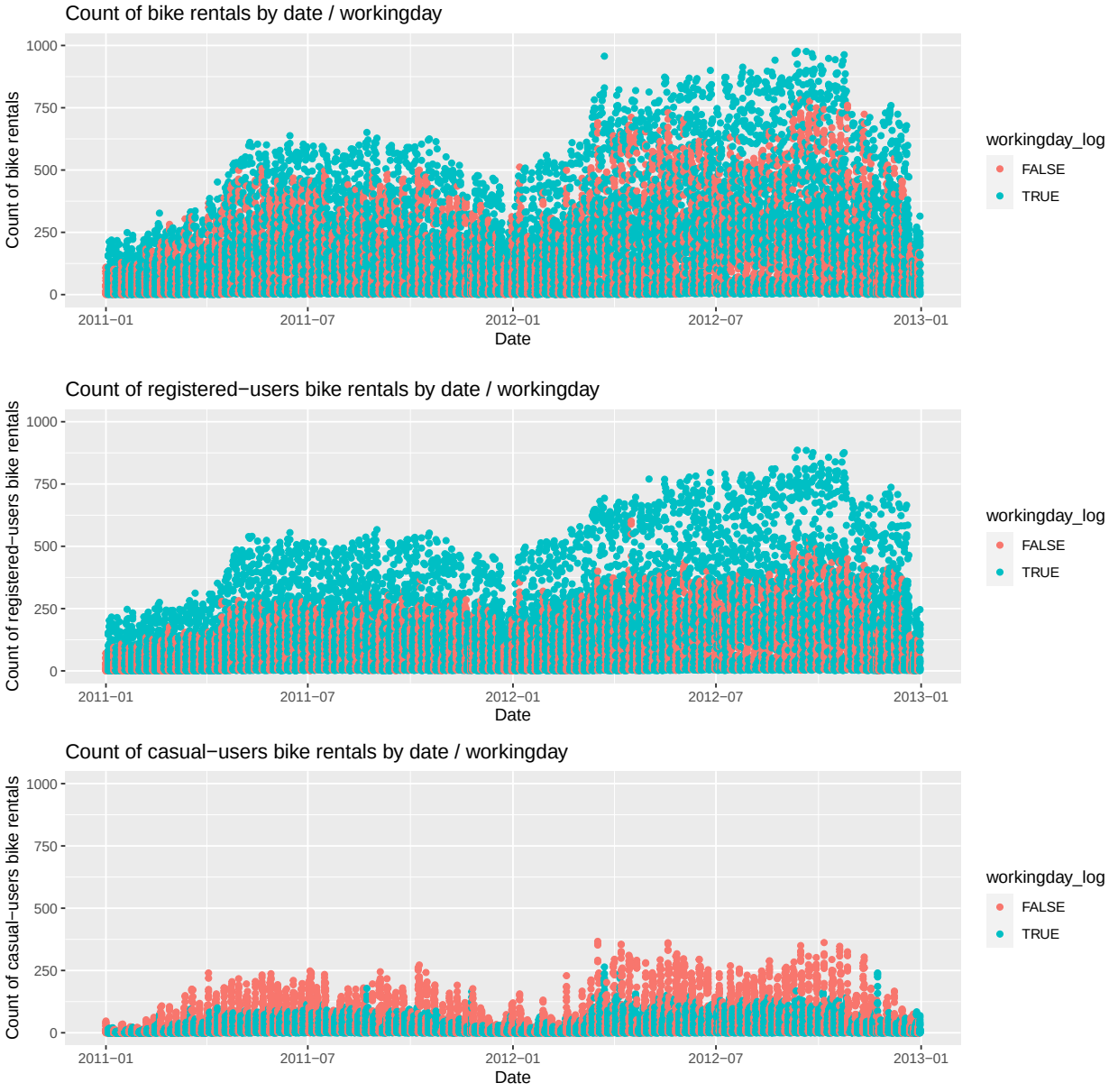






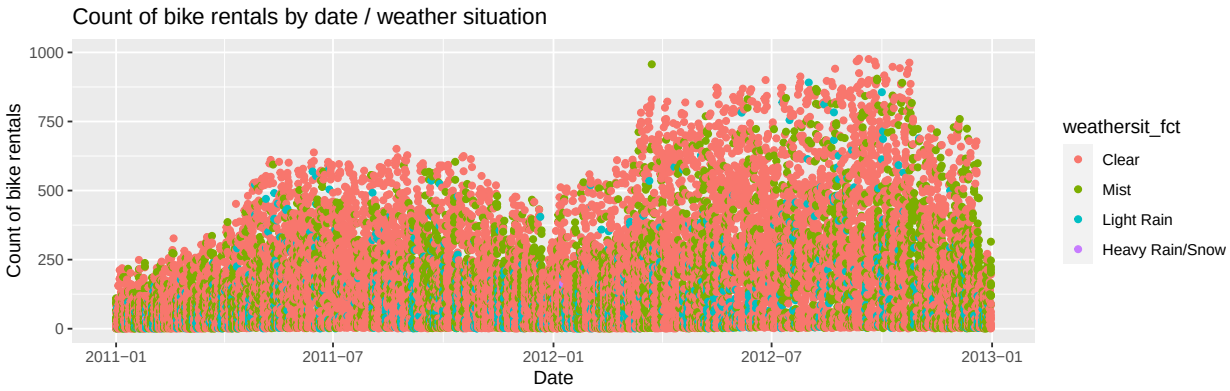
Week-days from Monday to Friday are the most busy for bike-rental service, this is really clear by looking at the "total count" and the "registered users" plots. Casual users prefer to ride the bike on Saturday and Sunday.

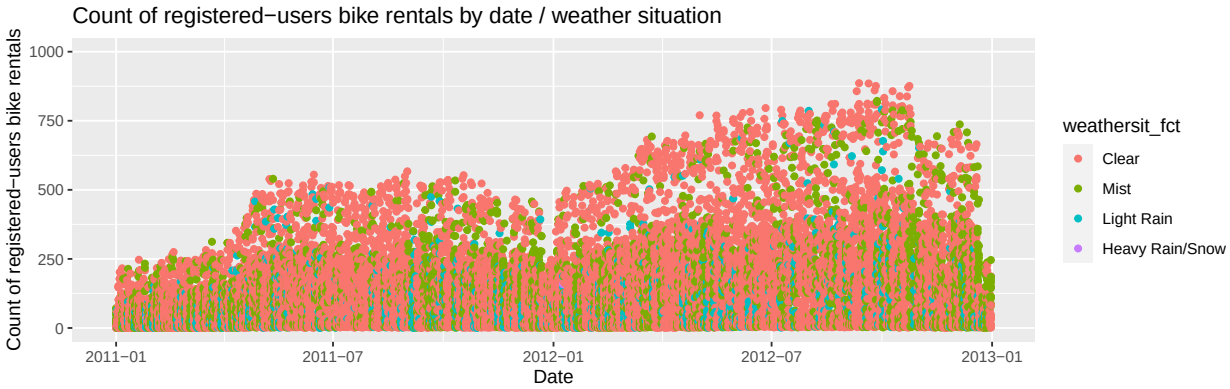
```
ggplot(data = dataset, aes(x = dteday, y = cnt, color = workingday_log)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Date",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by date / workingday")
```

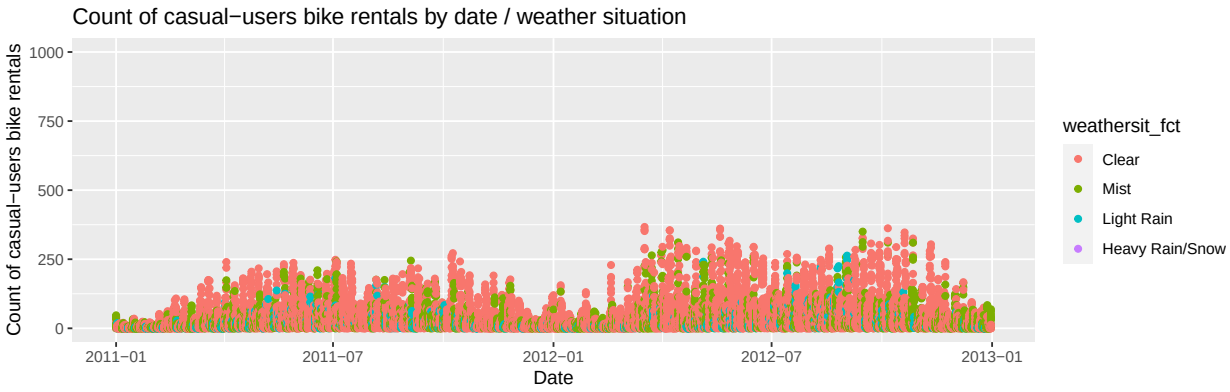


The highest bike-rental counts can be observed during working days. Registered users - which explain the most of the counts - indeed use the service at most during working days, this is another clue that most of them are commuters. Casual users prefer to apply for bike-rentals during spare time (i.e. where `workingday_log` is `FALSE`).

```
ggplot(data = dataset, aes(x = dteday, y = cnt, color = weathersit_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Date",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by date / weather situation")
```

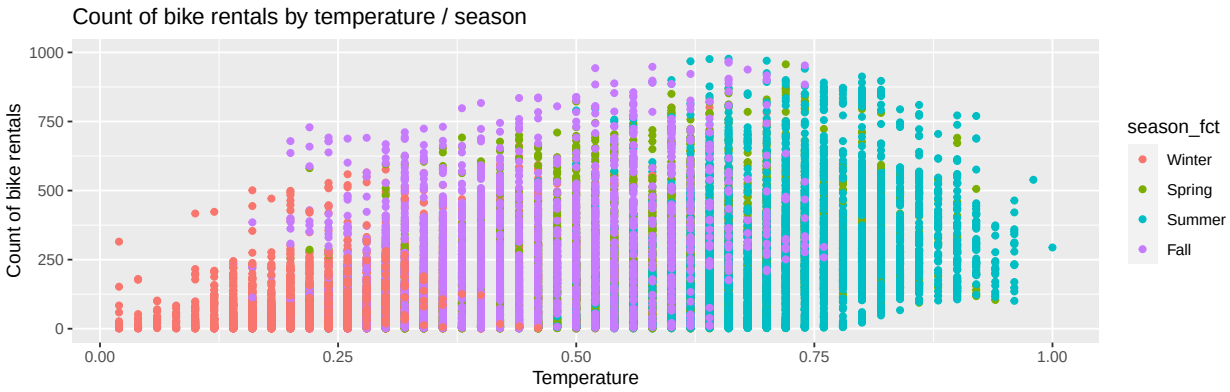


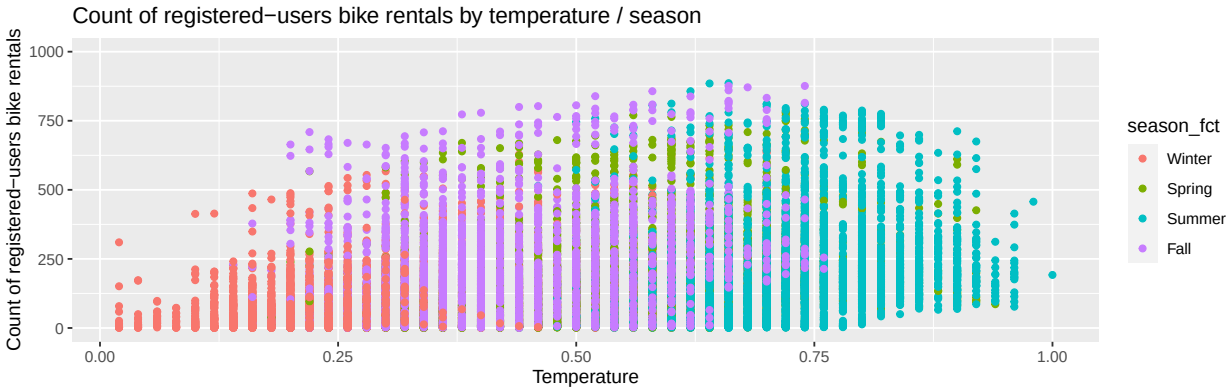


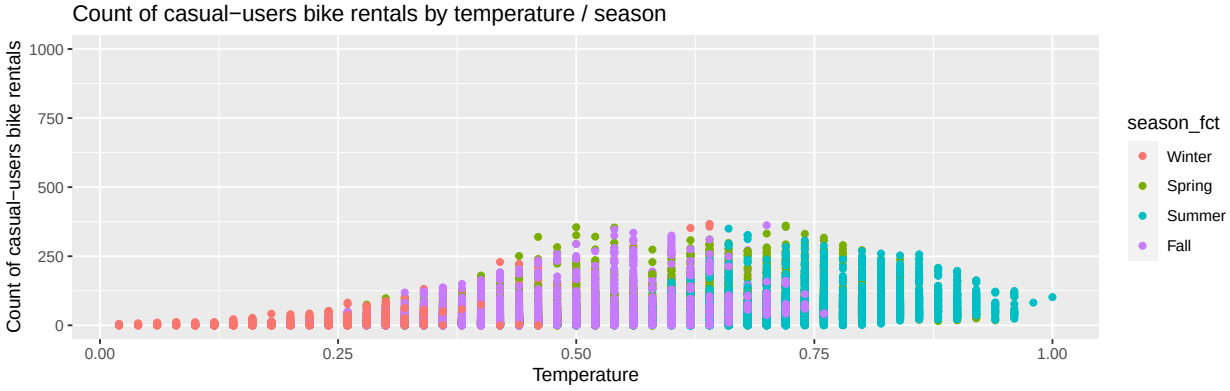


Most of the days in the dataset have "Clear" weather. It's interesting to see that registered users don't give up on bike-rental even when weather conditions are misty or when light rain is falling. On the other hand, casual users rarely choose this service when weather situation is poor.

```r
ggplot(data = dataset, aes(x = temp, y = cnt, color = season_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Temperature",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by temperature / season")
```

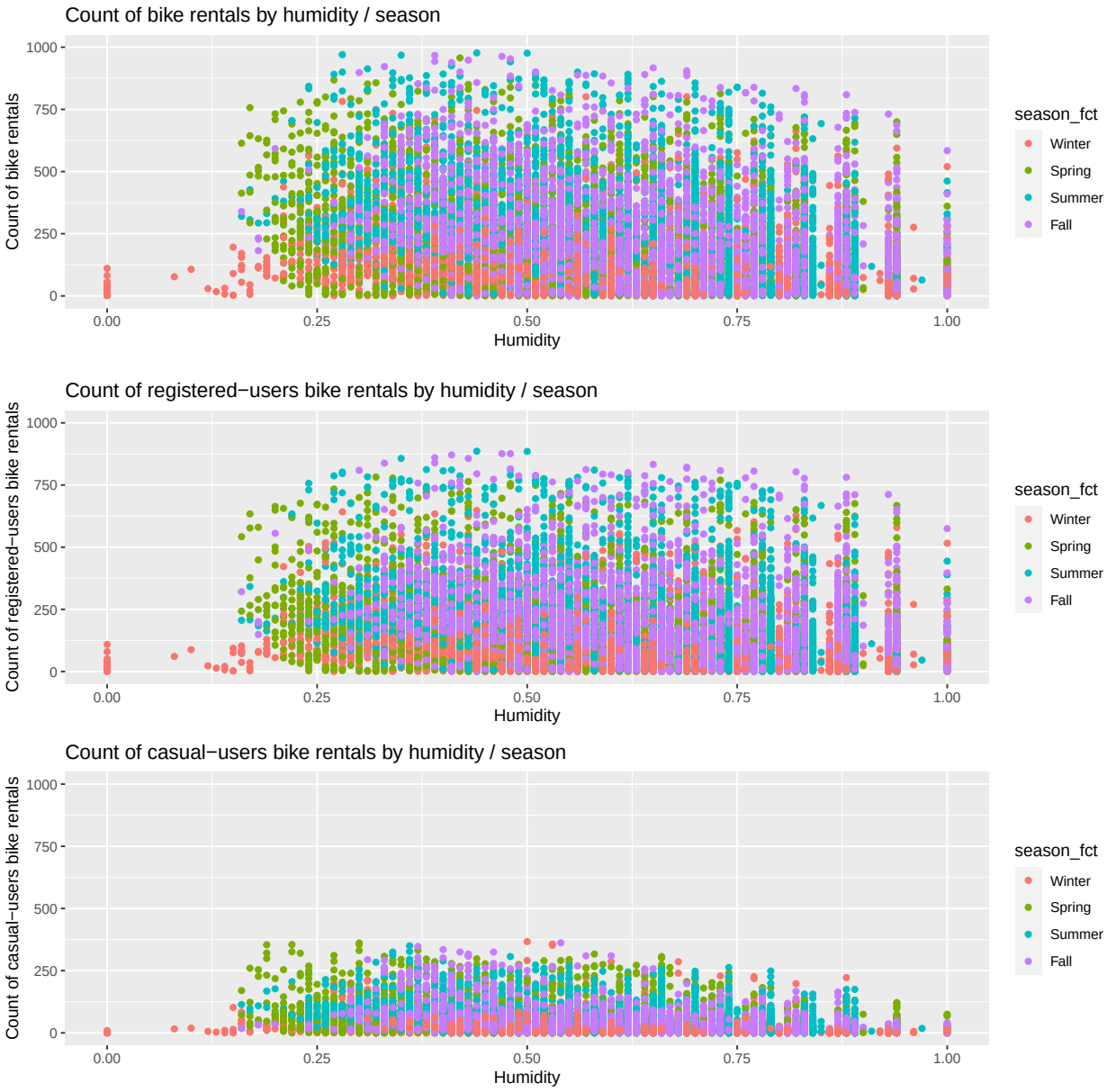






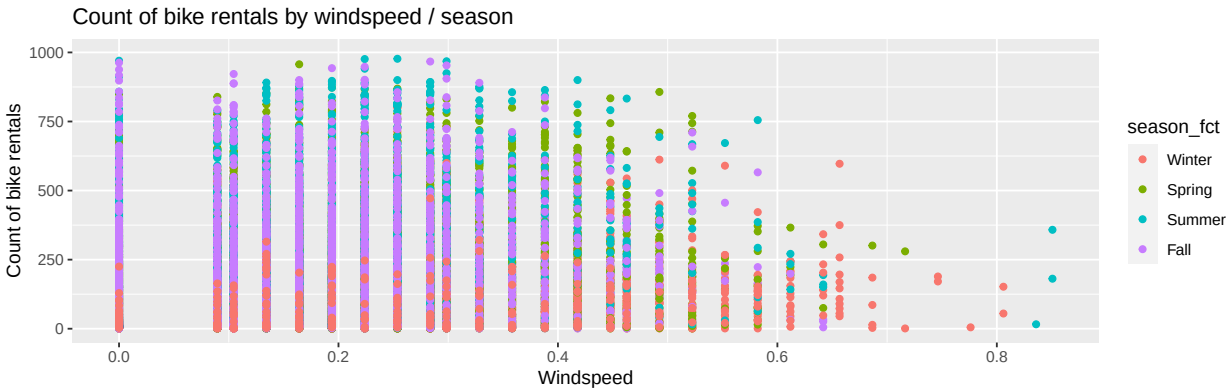
Temperature is an important driver when choosing the bike-rental service. The relationship between temperature and count is quite linear starting from the lowest degrees on, until temperatures between 21° C and 31° C are reached; then the hottest temperatures are negatively associated with the count of bike rentals. Plots of "counts by feeling temperature" show quite the same information, therefore they will not be displayed.

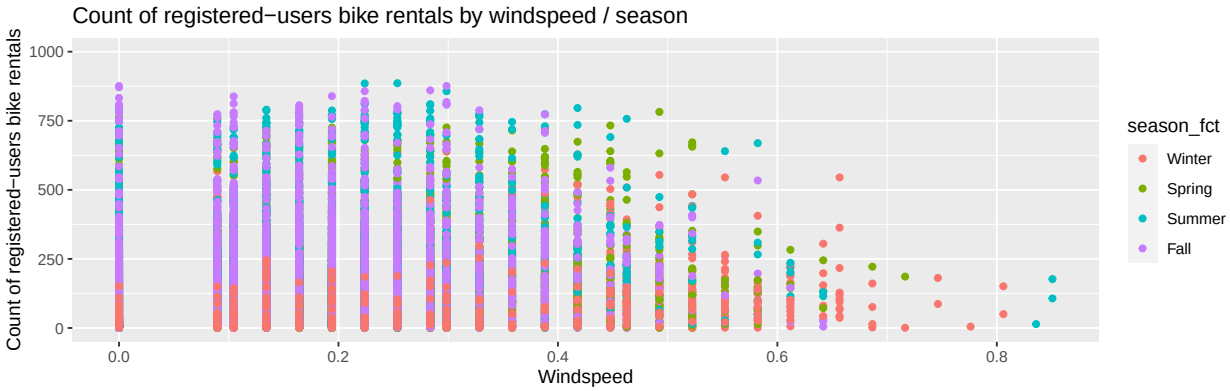
```
ggplot(data = dataset, aes(x = hum, y = cnt, color = season_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Humidity",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by humidity / season")
```

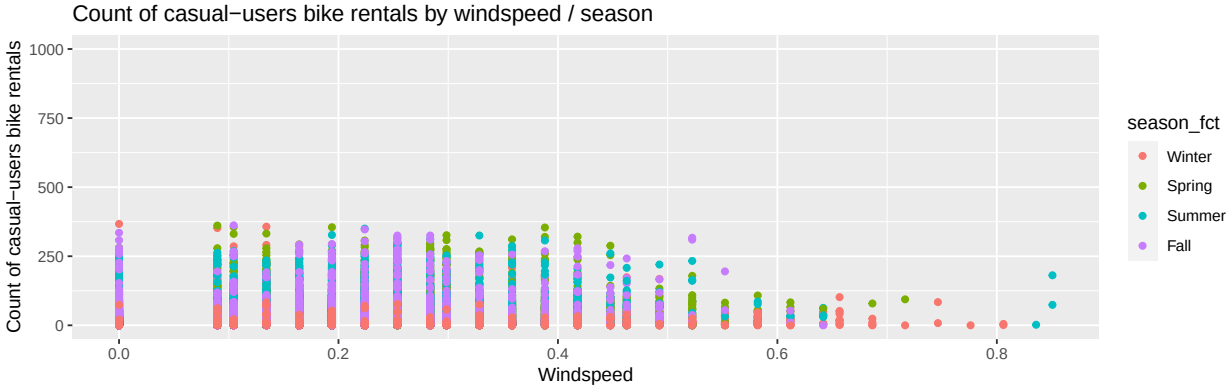


Humidity doesn't seem to be an important driver when choosing the bike-rental service. Registered users have quite the same count of bike rentals across the most various humidity values. On the opposite side, counts of bike-rentals for casual users decrease as humidity reaches its highest values.

```
ggplot(data = dataset, aes(x = windspeed, y = cnt, color = season_fct)) +
  geom_point() +
  ylim(c(0, 1000)) +
  labs(x = "Windspeed",
       y = "Count of bike rentals") +
  ggtitle("Count of bike rentals by windspeed / season")
```







As long as wind-speed remains within a normalized value of 0.4, it doesn't seem to affect bike-rental counts. Unfavourable wind-speed conditions instead can be a key element to decrease the bike-rental counts, for both registered and casual users. At the same time, the highest wind speeds can be observed during winter, so maybe a combination of these two situations can result in a low count of bike-rentals.

Wrap up: the bike-rental service is at most used by registered users. These users seem to be commuters, because they reveal a peak in usage during the specific hours of office-commuting and use the bike service especially during working days. These registered users prefer not to use the bike during winter and / or if

the weather conditions are extreme (a lot of humidity / too-windy days). Casual users are a small portion of total users and they use the bike service more frequently during holidays, Saturday or Sunday, in the afternoon and when the weather is clear and season is hot.

2.e Data refinement

The data exploration and visualization steps revealed that some features are dependent one from the other. For machine learning purpose, columns with dependencies should be excluded from the dataset. The following features will then be dropped:
1) `instant`: this is the id of the observations, it will not be used for machine learning purpose.
2) `dteday`: this will be dropped because it contains quite the same information as `yr` and `mnth` features.
3) `season`: this will be dropped because it contains quite the same information as `mnth` feature.
4) `atemp`: this will be dropped because it is a function of `temp` feature.
5) `casual` and `registered` column: these are also dependent variables, but in this project we develop a model only for `cnt` dependent variable, so these will be dropped.
6) all variables like `*_fct`, `*_log`: these are duplicates of existing variables, so these will be dropped.

```
dataset_selection <- dataset %>%
                     select(yr, mnth, hr, holiday, weekday, workingday, weathersit, temp,
                            hum, windspeed, cnt)
```

2.f Machine learning models

Now that features are all validated for usage, it's time to select some machine learning models that can be applied to the task of regression `cnt` = f (`independent variables`). Here is a list of the three model that we will test:
1) Poisson model could fit the task, because this regression problem is related to "counts" and "counts" are well explained by poisson models. Theory says that if the mean of `cnt` is close to its variance, then poisson model is ok. On the other hand, if the mean of `cnt` is much different from its variance, then the quasipoisson model should be used instead.

```
kable(dataset_selection %>%
  summarize(mean_cnt = mean(cnt),
            var_cnt = var(cnt)), caption = "Count of bike rentals - stats")
```

Table 16: Count of bike rentals - stats

mean_cnt	var_cnt
189.4631	32901.46

Data suggest that the quasipoisson model should be used, because the variance of `cnt` is much greater than its mean.
2) A decision tree model could fit the task, because independent variables can be "cut" into groups (e.g.: `mnth` equal to `1` or `windspeed` lower than `0.4`) and the dependent variable can assume higher or lower values depending on the groups of the independent variables that are defined.
3) A random forest model could fit the task, because it is basically an ensemble of many decision trees models.

The model selection should always come before the split into training set and test set, because, depending on the models that we want to test, a specific percentage of split is suitable. Usually a 80% - 20% split

between training and test is recommended; at the same time, given that we chose a random forest model which is keen to over-fitting, a 70% - 30% split is safer, so this is the split that will be applied.

```
set.seed(1, sample.kind = "Rounding")
index<-createDataPartition(dataset$instant,times=1,p=0.7,list=FALSE)
dataset_selection_train <- dataset_selection[index,]
dataset_selection_test <- dataset_selection[-index,]
```

Before fitting the models, we need to perform some data consistency checks. We want to be sure that the number of rows is consistent with the split applied:

```
row_check_train <- nrow(dataset_selection_train) - 0.7 * nrow
row_check_train
```

```
## [1] 1.7
```

```
row_check_test <- nrow(dataset_selection_test) - 0.3 * nrow
row_check_test
```

```
## [1] -1.7
```

The checks are close to zero, so data consistency is respected.

We now fit the quasipoisson model on the training set and determine the predictions on the test set:

```
qp_model <- glm(cnt ~ yr + mnth + hr + holiday + weekday + workingday + weathersit +
                      temp + hum + windspeed,
                data = dataset_selection_train,
                family = quasipoisson)

y_hat_qp_model <- predict(qp_model,
                          newdata = dataset_selection_test,
                          type = "response")
```

We now fit the decision tree model on the training set and determine the predictions on the test set.

```
dt_model <- rpart(cnt ~ yr + mnth + hr + holiday + weekday + workingday + weathersit +
                  temp + hum + windspeed,
                data = dataset_selection_train)

y_hat_dt_model <- predict(dt_model,
                          newdata = dataset_selection_test)
```

We finally fit the random forest model on the training set and determine the predictions on the test set.

```
rf_model <- randomForest(cnt ~ yr + mnth + hr + holiday + weekday + workingday + weathersit +
                               temp + hum + windspeed,
                         data = dataset_selection_train)

y_hat_rf_model <- predict(rf_model,
                          newdata = dataset_selection_test)
```

3. Model results and model performance

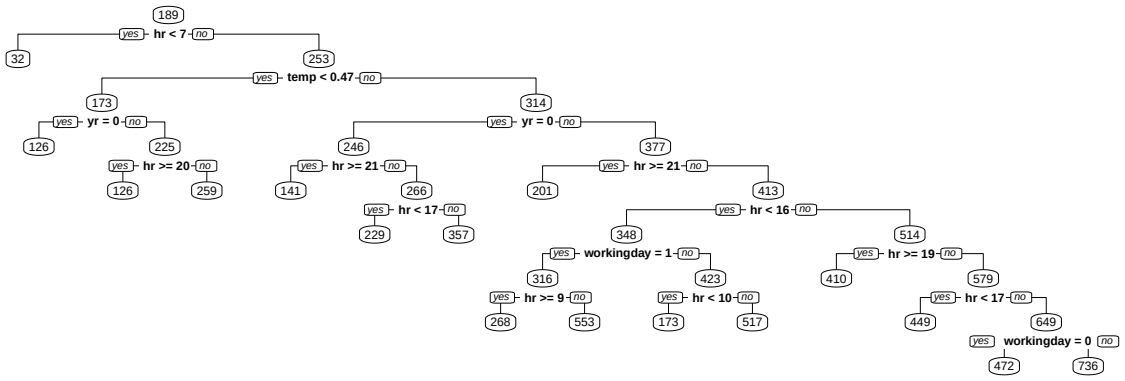
Let's have a look at the model results.
We start from the quasipoisson model:

```
qp_model
```

```
##
## Call:  glm(formula = cnt ~ yr + mnth + hr + holiday + weekday + workingday +
##     weathersit + temp + hum + windspeed, family = quasipoisson,
##     data = dataset_selection_train)
##
## Coefficients:
## (Intercept)           yr         mnth           hr      holiday      weekday
##    3.862987     0.425837     0.039733     0.045959    -0.169334     0.005894
##  workingday   weathersit         temp          hum    windspeed
##    0.007446    -0.021140     1.558130    -0.985237     0.220172
##
## Degrees of Freedom: 12166 Total (i.e. Null);  12156 Residual
## Null Deviance:       2022000
## Residual Deviance: 1192000   AIC: NA
```

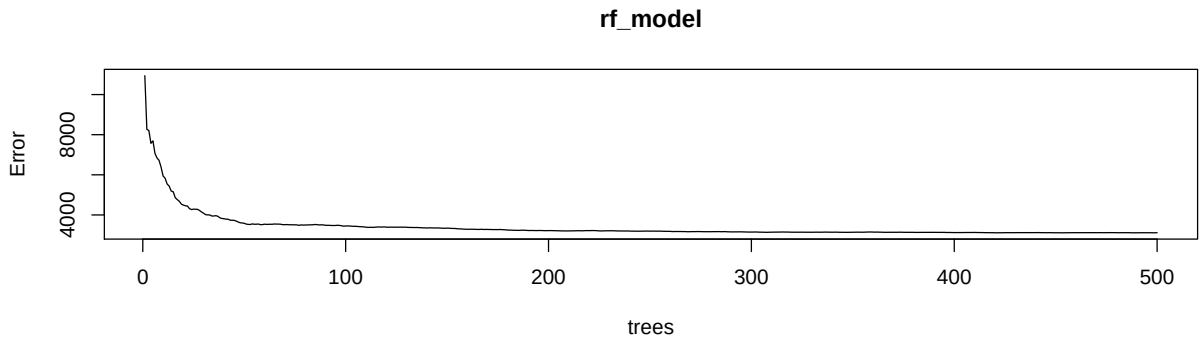
The quasipoisson model shows associations that we already spotted in data viz section: positive association with variables like `temp` (= temperature) and `yr` (= year) and negative association with `hum` (= humidity) and `holiday`.
The decision tree that we fitted can be well interpreted by looking at the specific plot; the same conclusions that we got in our data exploration section are declined in formulas by the model:

```
prp(dt_model, type = 2, branch=1, varlen=0, yesno=2)
```



The random forest model is not easy to display, in contrast to what we saw for the single decision tree, given that the random forest is an ensemble of many trees. Usually, random forests perform better than a single decision tree, at a cost of being less explicable. The following plot helps to see at what number of trees the "error" of the model doesn't decrease anymore - this is the minimum number of trees that is needed for the best performance of the model, and we can see this number is around 100:

```r
plot(rf_model)
```



In order to choose which model performs better, we should look at the RMSE of each model. Of course this measure will be evaluated on the test set, because this dataset is guaranteed to be independent from the observations of the training set, where the algorithms were trained. As already said in the executive summary, in order to calculate the RMSE, we follow these steps:
1) we take the residuals of each model - calculated on the test set;
2) then we square each of these;
3) we take the mean and then the square root of this value.
The lowest the RMSE, the better performance we have from the model.

```r
rmses <- dataset_selection_test %>%
  mutate(residual_qp = y_hat_qp_model - cnt,
         residual_dt = y_hat_dt_model - cnt,
         residual_rf = y_hat_rf_model - cnt) %>%
  summarize(rmse_qp_model = sqrt(mean(residual_qp^2)),
            rmse_dt_model = sqrt(mean(residual_dt^2)),
            rmse_rf_model = sqrt(mean(residual_rf^2)),
            sd_cnt = sd(cnt))
kable(rmses, caption = "RMSEs of the models")
```

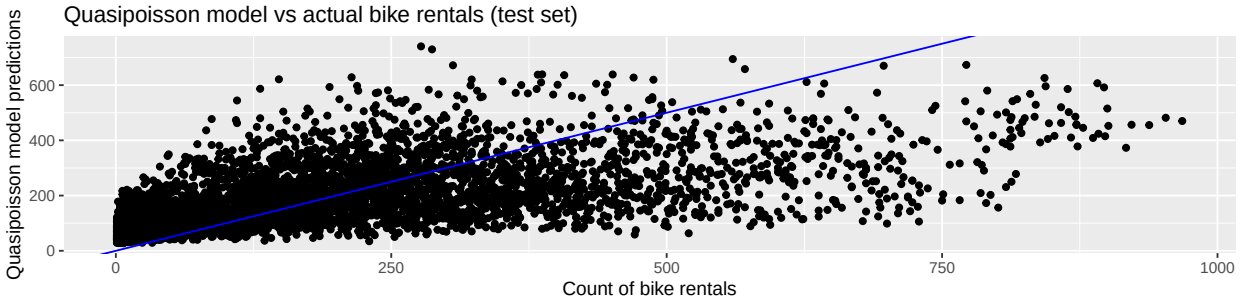
Table 17: RMSEs of the models

rmse_qp_model	rmse_dt_model	rmse_rf_model	sd_cnt
143.8837	97.72575	53.87928	182.3813

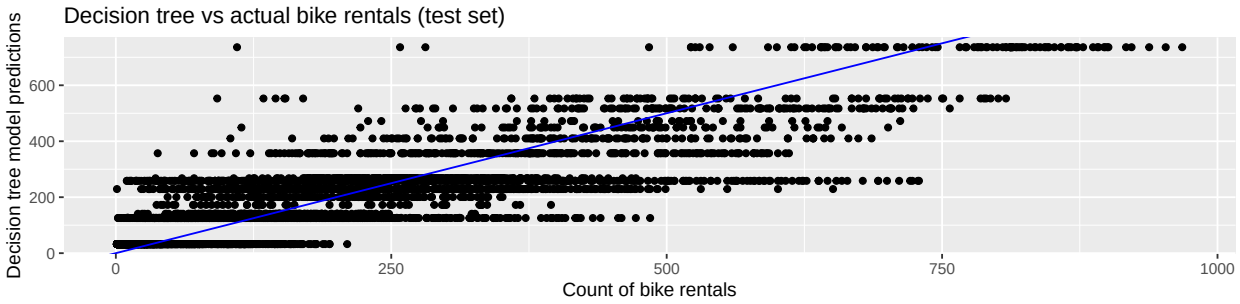
Quasipoisson model has an RMSE lower than standard deviation of the `cnt` variable, but it doesn't seem to be too explanatory. Decision tree performs better than quasipoisson. Random forest model performs way better than the quasipoisson and the decision tree models: its RMSE is the lowest.
These results can be better understood graphically: we plot, for each model, the actual `cnt` values on the test set versus the expected `cnt` values evaluated by the model. The closest the points lie on the identity line, the higher the model performance is.

```r
dataset_selection_test <- dataset_selection_test %>%
                          mutate(y_hat_qp_model = y_hat_qp_model,
                                 y_hat_dt_model = y_hat_dt_model,
                                 y_hat_rf_model = y_hat_rf_model)
```

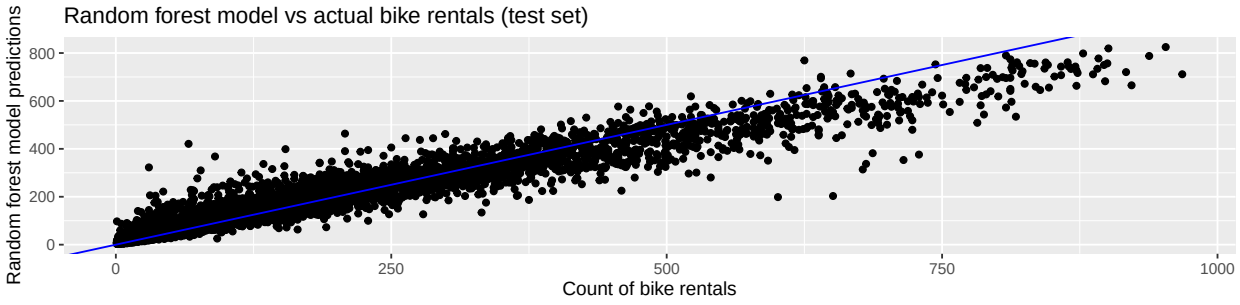
```
ggplot(data = dataset_selection_test, aes(x = cnt, y = y_hat_qp_model)) +
  geom_point() +  geom_abline(color = "blue") +
  labs(x = "Count of bike rentals",  y = "Quasipoisson model predictions") +
  ggtitle("Quasipoisson model vs actual bike rentals (test set)")
```




```
ggplot(data = dataset_selection_test, aes(x = cnt, y = y_hat_dt_model)) +
  geom_point() +  geom_abline(color = "blue") +
  labs(x = "Count of bike rentals", y = "Decision tree model predictions") +
  ggtitle("Decision tree vs actual bike rentals (test set)")
```




```
ggplot(data = dataset_selection_test, aes(x = cnt, y = y_hat_rf_model)) +
  geom_point() +  geom_abline(color = "blue") +
  labs(x = "Count of bike rentals", y = "Random forest model predictions") +
  ggtitle("Random forest model vs actual bike rentals (test set)")
```



We can clearly see that the random forest model has the best estimates for the actual `cnt` values.

4. Conclusion

In this analysis we saw that commuters seem to be the typical customer for bike-rental market segment. Their habits and needs influence a lot the bike-rental counts - for example, these counts are more likely to increase in working days and during "traffic peak" hours. At the same time, the choice for bike-rentals is really dependent on weather conditions, as one would expect from such a means of transportation: when the season is cold and wind speed or humidity are too high, bike-rentals counts go down.
A good machine learning model for bike count predictions is the random forest model: it is really well performing but is not easy to interpret. From the perspective of a Sales Manager of the bike-rental industry, the low explainability of such a model is its real limitation. The Sales department would indeed like to have help from the model in making the right decisions, but random forest models are really hard to summarize in action. Discussions on a decision tree model, which is less accurate but more explainable and actionable, could help the Sales team to support the bike-rentals strategy: dedicated marketing campaigns could be decided based on the model suggestions - e.g.: discount during "traffic peak" hours could increase the number of commuters deciding to join the service.
For future work, other models could be tested; at the same time, it could be interesting to see if further data collection could be feasible, because two years of time-series could be too few to infer good marketing decisions.