Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

PROJECTE O TESINA D'ESPECIALITAT

Títol

DEMAND FORECAST MODEL FOR A BICYCLE SHARING SERVICE

Autor

ORIOL COSP ARQUÉ

Tutora

MARIA ISABEL ORTEGO MARTÍNEZ

Departament

MA III – DEPARTAMENT DE MATEMÀTICA APLICADA III

Intensificació

MATEMÀTIQUES

Data

JUNE 2015

Abstract

Demand forecast model for a bicycle sharing service

Author: Oriol Cosp Arqué

Advisor: Maribel Ortego Martínez

Key words: bicycle sharing, transportation, forecast, predictive model, random forest, machine learning.

The present document is aimed to study the demand of bicycle sharing systems through weather variables. The main objective has been to create a data based predictive model for the demand of Capital Bikeshare, one of the U.S.A.'s larges bicycle sharing systems.

After having done some research on the problem and the methodologies that have been used to try to solve it before, the first step was to acquire the data on which the model was later built. This required downloading data from the Capital Bikeshare website, Washington D.C.'s local government website and designing a web scrapping tool to acquire weather data from an internet weather service website. This data was later aggregated into a more manageable single table.

Once the data was gathered, the R programming language was used to visualize and explore the data. An initial random forest model was built and tested on the data through cross validation. Using visualization techniques, a major problem in the model was discovered; it failed at capturing a temporal increasing trend on bicycle rentals. Since this was due to one of the random forest limitations, the model was improved by combining de-trending techniques with the random forest algorithm to significantly reduce error rates. Finally, the variable importance measures built in the random forest were used to draw conclusions about the variables driving the demand.

In this project a demand forecast model was developed, which can be easily trained and deployed. It can be used by bike sharing services to try to solve one of their major causes of customer loss (the lack of bicycles or parking spots at a given station) by either doing dynamic repositioning or designing an incentive based system.

Abstract

Model de demanda per a un servei de bicicletes públiques

Autor: Oriol Cosp Arqué

Tutora: Maribel Ortego Martínez

Paraules clau: servei de bicicletes públiques, transports, predicció, random forest, machine learning.

El present document té com a fita l'estudi de la demanda dels serveis de bicicletes públics a partir de variables meteorològiques. El principal objectiu ha estat el de crear un model predictiu basat en dades per a la demanda de Capital Bikeshare, un del majors serveis de bicicletes públiques dels EUA.

Després d'haver fet recerca sobre el problema i les metodologies que han estat utilitzades per a solucionar-lo amb anterioritat, el primer pas era el d'adquirir les dades sobre les que es construiria el model. Això va requerir descarregar dades de la web de Capital Bikeshare, del govern local de Washington D.C. i dissenyar una eina de web scrapping per tal d'adquirir dades de meteorologia d'una pàgina web de serveis meteorològics. Aquestes dades es van agregar en una sola taula més manejable.

Un cop reunides les dades, es va utilitzar el llenguatge de programació R per a visualitzar i explorar les dades. Es va construir un model inicial de random forest i va ser testejat amb les dades mitjançant validació creuada. Emprant tècniques de visualització, es va descobrir un greu problema; el model no aconseguia capturar una tendència temporal creixent en el lloguer de bicicletes. Com que això es devia a una de les limitacions del random forest, el model va ser millorat combinant tècniques per a remoure la tendència amb l'algorisme del random forest, reduint l'error de manera significativa. Finalment, les mesures d'importància implementades en el random forest van ser utilitzades per a extreure conclusions sobre les variables conductores de la demanda.

En aquest projecte es va desenvolupar un model de predicció de la demanda que pot ser fàcilment entrenat i implementat. Pot ser utilitzat pels serveis de bicicletes públiques per a intentar solucionar una de les majors causes de pèrdua de clients que pateixen (la manca de bicicletes o de llocs per aparcar en una estació) mitjançant el reposicionament dinàmic o el disseny d'un sistema basat en incentius.

Abstract

Modelo de demanda para un servicio de bicicletas públicas

Autor: Oriol Cosp Arqué

Tutora: Maribel Ortego Martínez

Palabras clave: servicio de bicicletas públicas, transportes, predicción, random forest, machine learning.

El presente documento tiene como meta el estudio de la demanda de los servicios de bicicletas públicos a partir de variables meteorológicas. El principal objetivo ha sido el de crear un modelo predictivo basado en datos para la demanda de Capital Bikeshare, uno de los mayores servicios de bicicletas públicas de EEUU.

Después de haber hecho investigación sobre el problema y las metodologías que han sido utilizadas para solucionarlo con anterioridad, el primer paso era el de adquirir los datos sobre las que se construiría el modelo. Esto requirió descargar datos de la web de Capital Bikeshare, del gobierno local de Washington DC y diseñar una herramienta de web scrapping para adquirir datos de meteorología de una página web de servicios meteorológicos. Estos datos se agregaron en una sola tabla más manejable.

Una vez reunidos los datos, se utilizó el lenguaje de programación R para visualizar y explorar los datos. Se construyó un modelo inicial de random forest y fue testeado con los datos mediante validación cruzada. Empleando técnicas de visualización, se descubrió un grave problema; el modelo no conseguía capturar una tendencia temporal creciente en el alquiler de bicicletas. Como esto se debía a una de las limitaciones del random forest, el modelo fue mejorado combinando técnicas para remover la tendencia con el algoritmo del random forest, reduciendo el error de manera significativa. Por último, las medidas de importancia implementadas en el random forest fueron utilizadas para extraer conclusiones sobre las variables conductoras de la demanda.

En este proyecto se desarrolló un modelo de predicción de la demanda que puede ser fácilmente entrenado e implementado. Puede ser utilizado por los servicios de bicicletas públicas para intentar solucionar una de las mayores causas de pérdida de clientes que sufren (la falta de bicicletas o de lugares para aparcar en una estación) mediante el reposicionamiento dinámico o el diseño de un sistema basado en incentivos.

Index

1- Introduction

A bicycle sharing system is a service which allows multiple users to share the use of bicycles distributed in kiosks along a city. Users can borrow a bike at a station and return it in a different station.

Even though the first bike sharing programs appeared on the 1960's and 70's in Europe, it wasn't until the mid-2000's with the introduction of Information Technology to monitor usage that bicycle sharing systems became popular worldwide, mostly implemented by government agencies, sometimes in a public-private partnership. As of June 2014, public bike sharing systems were available on five continents, including 712 cities, operating approximately 806,200 bicycles at 37,500 stations (Shaheen, 2015).

The data generated by these systems makes them attractive for researchers of many fields because the duration of travel, departure location, arrival location, and time elapsed are explicitly recorded. Bike sharing systems can be considered as a sensor network, making it a powerful tool for studying mobility in a city.

Figure 1. Capital Bikeshare bicycles (source: www.wikipedia.com)

Bike-sharing has some advantages over owning a bicycle as users don't have to worry about theft or vandalism, parking or storage, and maintenance requirements. However, there are also downsides, since the number of places where bicycles can be rented or returned are limited. A full station will not allow users to return the bicycle, forcing them to look for another station that has some available slots. On the other hand, an empty station will not have any bicycles to rent and users will have to get to another station or simply use another mean of transportation. These inconveniences can be very frustrating for customers and are sometimes the reason some of them switch to other transportation methods.

Simultaneously meeting the demand for bicycles and empty bike slots is a challenging problem because of the imbalances between the return and rent rates at the stations. While the flow of commuters is more or less balanced over the course of a day, the flow of bicycles may behave differently. Not all users of a bicycle sharing system use it on all their trips, for instance a user may choose to ride a bike in a direction and use another mean of transportation on the way back for a variety of reasons (weather, topography of the route, availability and frequency of bus/train service, etc.). In some cases this imbalance is persistent (for example a station located

at the top of a hill) while in others it may be temporary (for example a bike share station near a suburban train station may have high return rates during the morning when commuters get into the city and a high rental rates in the afternoon when they return home).

Even though some bicycle balance incentive systems have been developed (Singla, Santoni, Meenen, & Krause, 2015), some of the imbalance will still persist. In order to satisfy the user demand subject to these imbalances a fleet of light trucks transferring bicycles among stations is usually used on bicycle sharing systems. Repositioning bikes during the night when the system is almost idle is called static repositioning, meanwhile doing so during the day to cope with looming shortages is called dynamic repositioning (Raviv & Kolka, 2013).

In order to plan a repositioning system, a demand forecast model can be used to predict which stations will be full and which ones will run out of bicycles so that the system can relocate them as needed.

As a first approach to solving this problem, a data based forecast method is going to be developed to predict the total demand of bicycles for a bicycle sharing system for each hour, based on meteorological data. The model could be later scaled to fit evert single station.

For this project Washington D.C.'s Capital Bikeshare Systems data is going to be used. A machine learning method has been chosen to solve the problem, namely the random forest algorithm which is one of the most successful techniques on the field.

2- Goals

The aim of this project is to develop a demand forecast model for the Capital Bikeshare system in Washington D.C. This is a model that for every day and hour, given some meteorological variables and whether the day is a holiday or not, attempts to predict the total number of bicycles that will be rented that hour.

In order to build the model, these steps have to be followed:

- Research methodologies that have been used to analyze problems of a similar nature.
- Acquire the required data to build the model.
- Analyze the data and find out which variables drive the demand.
- Design a predictive model to forecast the demand.
- Implement and tune the model to better fit the data.
- Draw conclusions from the previous work.

3- Methodology

3.1- Overview

After having done some research on the problem and the methodologies that have been used to try to solve it before, in order to achieve the proposed goal, the first step was to aquire the data on which the model would be built. This required downloading data from the Capital Bikeshare website, Washington D.C.'s local government website and using a program to scrap weather data from an internet weather service website.

The acquired data had diverse formats and was divided into multiple files. Data belonging to each of the three components above had to be put together on a single table, and then those tables were joined into a table containing all the relevant information. This process was done through the data management language SQL.

Once the data was gathered, the R programming language was used to visualize and explore the data. After that, an initial random forest model was built and tested on the data making use of the previous insights.

Using that model as a starting point, the model was modified and iterated on through the scientific method: hypothesizing and testing if said hypothesis made the model better at predicting the bike rentals.

A final model was developed and used to draw some conclusions on the problem.

3.2- Web scrapping

Web scraping is a computer technique of extracting information from websites. In order to gather the weather information, a web scraping algorithm was coded using Python and the urllib.request Python module.

Since the data that had to be scrapped was stored in a multitude of different URLs each of them containing the weather observations for a given day, the first step in this process was to figure out a way to access each of them. Fortunately, the URLs were built in a way that the date was included in them.

A list of the dates for all days in the 2011-2014 period was built using SQL and stored in a CSV (comma separated value) file. This list was opened using Python and for every line in the list, the URL that contained the weather data for that day was built using string manipulation commands.

Once the URL had been produced, the urllib.request Python module was used to download the content of that URL. The downloaded content was read line by line, each line containing hourly weather data for the given day, and copied on a file.

Once the process had gone through for every date, the file containing all the downloaded information was saved as a CSV, to be later treated using PostgreSQL.

3.3- Brief introduction to machine learning

Machine learning is a subfield of computer science, which evolved from pattern recognition and artificial intelligence. The basic idea behind machine learning is that computers can learn from data through algorithms and once trained they can make data-driven predictions.

Supervised learning is the branch of machine learning that focuses on learning from labeled data, the examples from which the algorithm learns also contain the desired output. Other branches of machine learning are unsupervised learning (where the goal is to find underlying patterns in the data) and reinforcement learning (where the computer tries different options and is given a notion of how well it performed).

Each observation used on supervised learning is usually divided into features or inputs ($X = (x_1, x_2, \dots, x_n)$), the variables that are used to make the prediction, and responses or outputs (y), the values that want to be predicted ($y \sim f(X)$).

To assess the performance of algorithms, observations are usually divided into a training set and a test set. The training set is used together with the machine learning algorithm to teach the computer a model, the test set is used to evaluate how good that model can generalize by comparing predictions of the trained algorithm on the test set with the known results through a loss function. To reduce the variance, this can be done multiple times using different partitions and averaging their results to obtain the validation result; this process is called cross validation and can be used to compare different models.

On supervised learning two different types of problems exist:

- Classification: The algorithm assigns inputs to a class from a set of different classes.
- Regression: The algorithm outputs a continuous value.

The bicycle sharing system demand forecast problem requires of a regression algorithm, since the final value that is being forecasted is the hourly count of rented bicycles.

There are multiple different machine learning regression algorithms that come from different approaches such as decision tree learning, neural networks, support vector machines, etc. However they have in common that no distribution on the data is assumed and this makes them very flexible.

Even though machine learning is a relatively young field, successful cases of applying its techniques to civil engineering problems can be found on the literature. Machine learning methods have shown better accuracy than preexisting methods on sediment transport (Bhattacharya, Price, & Solomatine, 2007), boosted decision trees have been found to have high prediction rates on predicting the outcome of construction litigation , support vector machines appear to be a practical tool for the determination of Over Consolidation Rate …

For this project, the random forest algorithm was chosen due to its robustness and its ability to generalize well (Caruana & Niculescu-mizil, 2006). Some examples of random forests usage to tackle engineering problems are modeling the spatial patterns of fire events (Oliveira, Oehler, San-Miguel-Ayanz, Camia, & Pereira, 2012), hourly urban water demand forecast (Herrera, Torgo, Izquierdo, & Pérez-García, 2010) and exploration of the relationship between soil properties and the deterioration of metallic pipes (Liu, Sadiq, Rajani, & Najjaran, 2010).

4- Literature

Bikesharing systems demand has been modeled in the past as part of systems that optimize fleet repositioning and also as a way to study the way bikesharing systems' users behave.

One of the first attempts to model a bikesharing system's demand was made by Borgnat, Abry, & Rouquier (2009) as a part of their study of Lyon's Vélo'V. Their forecast model combined statistical models to predict de cyclical fluctuations with linear regression.

Barcelona's Bicing has been studied at least twice even though it only offers real time data and its data platform has to be continually scrapped to obtain a certain volume of data. It was first studied by Froehlich, Neumann, & Oliver (2009) who used Bicing data to study the city's mobility. They used predicting methods of increasing complexity: last value, historic mean, historic trend and Bayesian networks. Kaltenbrunner, Meza, Grivolla, Codina, & Banchs (2010) also explored Bicing's patterns, to predict the number of bicycles on a station they used some simple models as well as some time series analysis tools like the Auto-Regressive Moving Average (ARMA).

Another case of a city being studied through its bikeshare systems is Vogel, Greiser, & Mattfeld's (2011) study of Vienna's Citybike Wien's patterns through cluster analysis. K-means clustering, expectation maximization and sequential Information-Bottleneck were the algorithms of choice.

Finally, Caggiani & Ottomanelli (2013) designed a model for fleet repositioning based on simulations. To forecast the demand they used Neural Networks.

5- About Capital Bikeshare

Capital Bikeshare is a bicycle sharing system that serves Washington D.C, neighboring counties Arlington and Montgomery and the city of Alexandria. The service offers more than 2.500 bicycles and over 300 stations. It was the first public-private partnership bike sharing system in the US, and also the largest until May 2013 when New York's Citi Bike launched.

The Capital Bikeshare system is owned by the participating jurisdictions and is operated by Motivate, a Brooklyn, NY-based company that operates several other bikesharing systems including Citibike in New York City, Hubway in Boston and Divvy Bikes in Chicago.

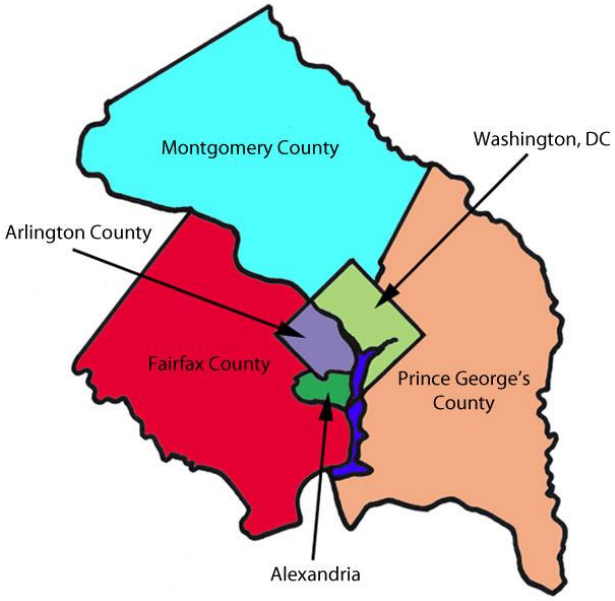

Figure 2 Washington, DC area map (source: http://www.mosquitocontroldc.com/mosquito-control-maryland/)

Trip history data is available for download at the Capital Bikeshare website (http://www.capitalbikeshare.com), which made this research possible.

6- Data Acquisition

Capital Bikeshare data was downloaded from the website (http://www.capitalbikeshare.com) and combined with weather and holiday data to obtain the dataset with which the model was trained. Time, holiday and weather variables were used as predictor variables, although other variables which were not available (like number of registered users) could have been useful.

6.1- Trip history data:

Whenever a rental occurs in the Capital Bikeshare system, their software records basic data about the trip in order to monitor the system. The data is stored, and can be downloaded from the Capital Bikeshare official website. Respecting data privacy policies, private data including member names is not included in the files. The data available when the research was conducted belonged to the 2010Q4-2014Q4 period.

The data came in a .csv format and separated in files, each containing data for one quarter of a year. Those files consisted of a list of trips featuring the following variables:

- Duration: Duration of trip
- Start date: Start date and time
- End date: End date and time
- Start station: Starting station name and number
- End station: Ending station name and number
- Bike #: ID number of bike used for the trip
- Member Type: Whether the user was a Registered or Casual member.

The data was manually downloaded from the website and manipulated using the PostgreSQL database management system to summarize the individual trip data into hourly counts, using the start date and hour as reference. The formats of the different files differed a bit in some cases and were treated separately. At the end all data was aggregated into one table containing data from years 2011-2014. Data from 2010Q4 was not used because it was suspected of containing irregularities, since it was the first period for which data was made available.

6.2- Holiday data:

The website of Washington D.C.'s local government hosts the holiday schedule for the city (http://dchr.dc.gov/page/holiday-schedules). Schedules for years 2011-2014 were manually copied and pasted on a text file, which was later processed using PostgreSQL. The data contained the date and the reason for the holiday, which was disregarded.

This data translated into a binary variable, being 1 in case the day was a holiday and 0 otherwise.

6.3- Weather data:

Weather Underground (http://www.wunderground.com) is an internet weather service that offers historical data as well as weather forecasts for locations all around the world. Hourly weather data for individual days can be easily consulted on the website, however, data for 1461 days was needed so manual download was not an option.

The data was downloaded using the web scrapping technique described in the methodologies chapter. Historical hourly weather data from every day in years 2011-2014 was downloaded and saved into a text file. The data was imported to PostgreSQL and properly structured.

From the downloaded data, the following variables were kept:

- Hour: Time at which the data was recorded.
- Temperature: Temperature measured in Celsius degrees.
- Dew Point: temperature at which the water vapor in a sample of air at constant pressure condenses into liquid water at the same rate at which it evaporates, measured in Celsius degrees.
- Humidity: amount of water vapor in the air relative to the maximum for that temperature, expressed as a percent.
- Mean sea level pressure: Atmospheric pressure at sea level. Measured in hPa.
- Visibility: Distance at which an object or light can be clearly discerned, measured in Km.
- Wind Direction: Direction from which the wind originates. This information is given as a class, featuring 16 different directions (N, NNE, NE, ENE, E, ESE …), plus two other categories (calm and variable).
- Wind speed: Flow velocity of the wind, measured in Km/h.
- Precipitation: Amount of water that rained. Measured in mm.
- Conditions: Description of the weather state (i.e. foggy, overcast, etc.), divided into 25 different classes.

6.4- Resulting dataset:

The above data was joined by date and hour, and a table was created with the following columns:

- Year
- Month
- Day
- Hour
- Temperature
- Dew point
- Humidity
- Pressure
- Visibility
- Wind direction
- Wind speed
- Precipitation
- Conditions
- Holiday
- Casual
- Registered
- Total Count

Out of a total of 35,064 observations, weather data was missing for 16 of them. Since it represented such a low percentage of the dataset, those hours were excluded from the study.

SQL and Python codes used are included in the appendix section.

7- Original Variables

The aim of this section is to gain insights and give a deeper understanding of the variables and the dependence of the casual and registered user bicycle rentals on them.

To accomplish this purpose, a highly visual approach was taken, trying to show relevant interaction through plots. One of the tools used is the boxplot: a way of graphically describing data through their quartiles; the top bound of the box represents the 1st quartile while the bottom one the 3rd quartile, the horizontal line inside the box is the median (2nd quartile) and the whiskers extend to a length of 1.5 times the distance between the 1st and 3rd quartiles (interquartile range), leaving the points outside of the whisker range as outliers represented by dots.

Another technique that has been used in conjunction with the boxplot is bucketing variables, this is transforming a continuous distribution of values into a discrete one by grouping the observations into ranges of values.

Only the most insightful plots have been included in this section. Plots for the rest of variables can be found on appendix C.

- ## Registered

Number of bicycle rentals by registered (monthly or annual pass) users per hour.

Registered users are most likely commuters who use the bicycle as mean of transportation on a regular basis. The fact that there are lots of hours with low rentals is due to night hours having low amounts of rentals.

- ## Casual

Number of bicycle rentals by casual (1-5 day pass) users per hour.

Casual users are a more varied group, probably a mix of tourists, locals using the bikes for leisure purposes, maybe some commuters in need of a temporary alternative, etc.

As can be seen from the plots, registered users account for a significantly higher amount of rentals than casual users do.

- ## Year

As it could expected, both the number of casual and registered user rentals increased as years passed since the service started operating.

- ## Month

Both registered and casual users show similar monthly trends, reducing bike rentals on colder months (December, January, February, March), however, casual users' rentals decrease is more pronounced on those months.

- ## Ndays

Number of days since January 1st 2011, the first day whose data was used for this project.

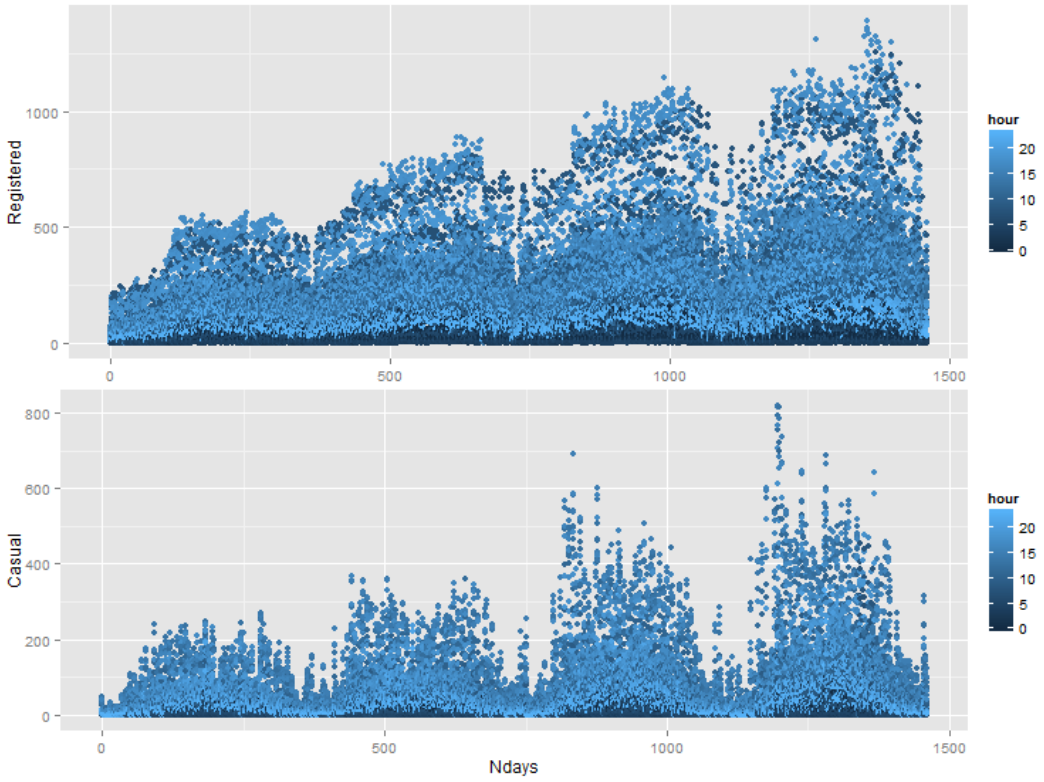

Figure 3. Scatterplot of casual and registered counts depending on the Ndays and hour variables

Previously seen annual and monthly trends are also shown in this plot. Moreover, it can be observed that there is more variability to the number of casual bike rentals than to the registered rentals.

- Weekday

Day of the week, 0 being Sunday, 1 Monday, etc.

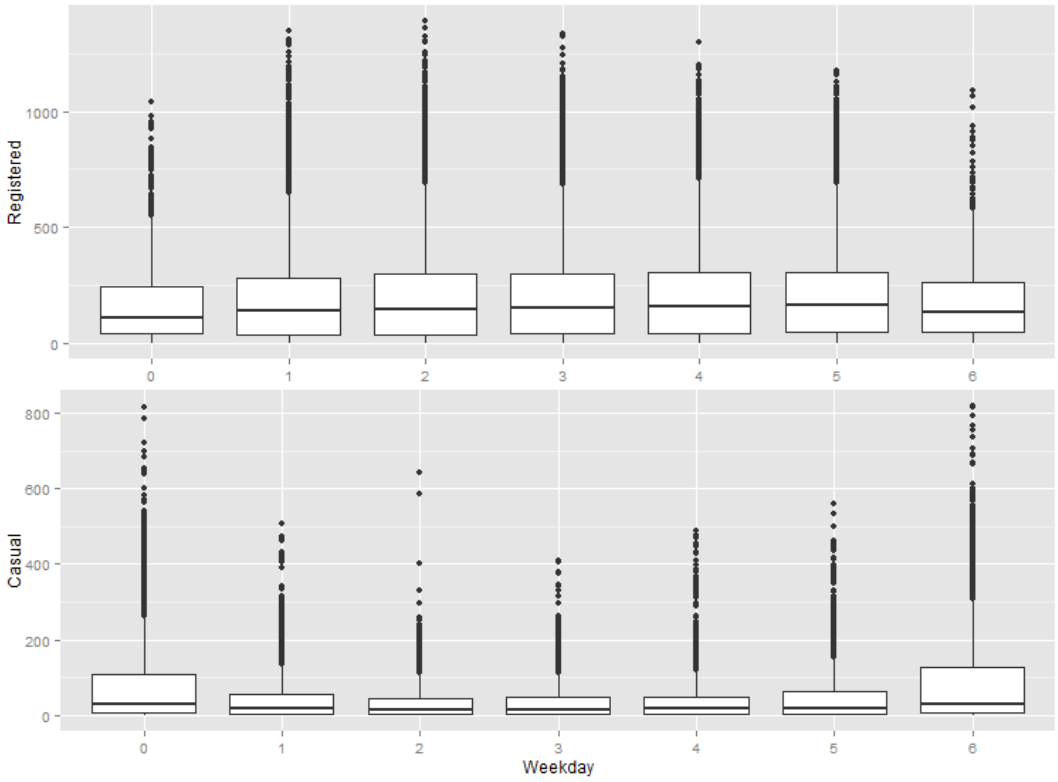

Figure 4. Boxplot of registered and casual counts depending on the weekday

Casual and registered users show different patterns, with casual users renting more bikes during the weekends, while registered users have higher rental rates from Monday to Friday which could be expected from commuters.

- Hour

Hour of the day, from 0 to 24.

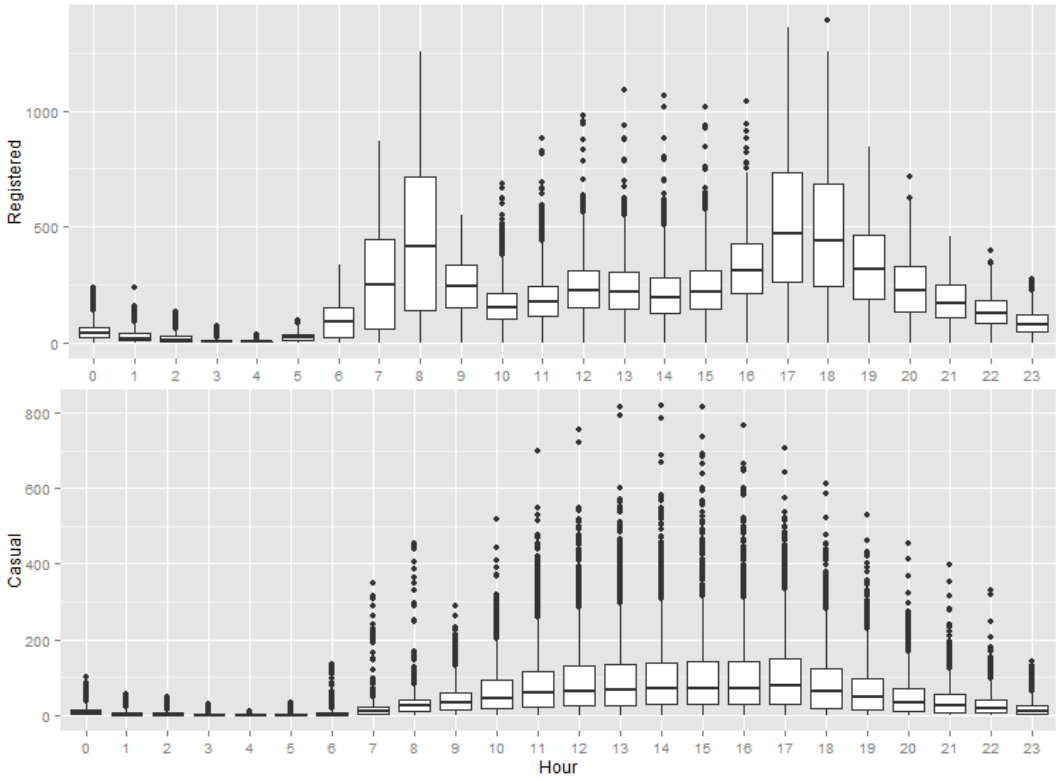

Figure 5. Boxplot of the hourly distribution of registered and casual counts

Typical transportation trends can be observed from the registered user rentals, with two peaks, one on the morning when people go to work and one in the afternoon when workers go back home, and a valley between them. Casual users present a different shape with more or less stable rentals from 10h to 19h.

- Holiday

Whether a day is a holiday (1) or not (0).

Registered users have a lower amount of rentals on holidays than on non-holidays, casual users seem to be barely affected.

- Workingday

Whether a day is a working day (1), a day that is neither a holiday nor part of a weekend, or not (0).

- Temperature

Temperature measured in Celsius degrees.

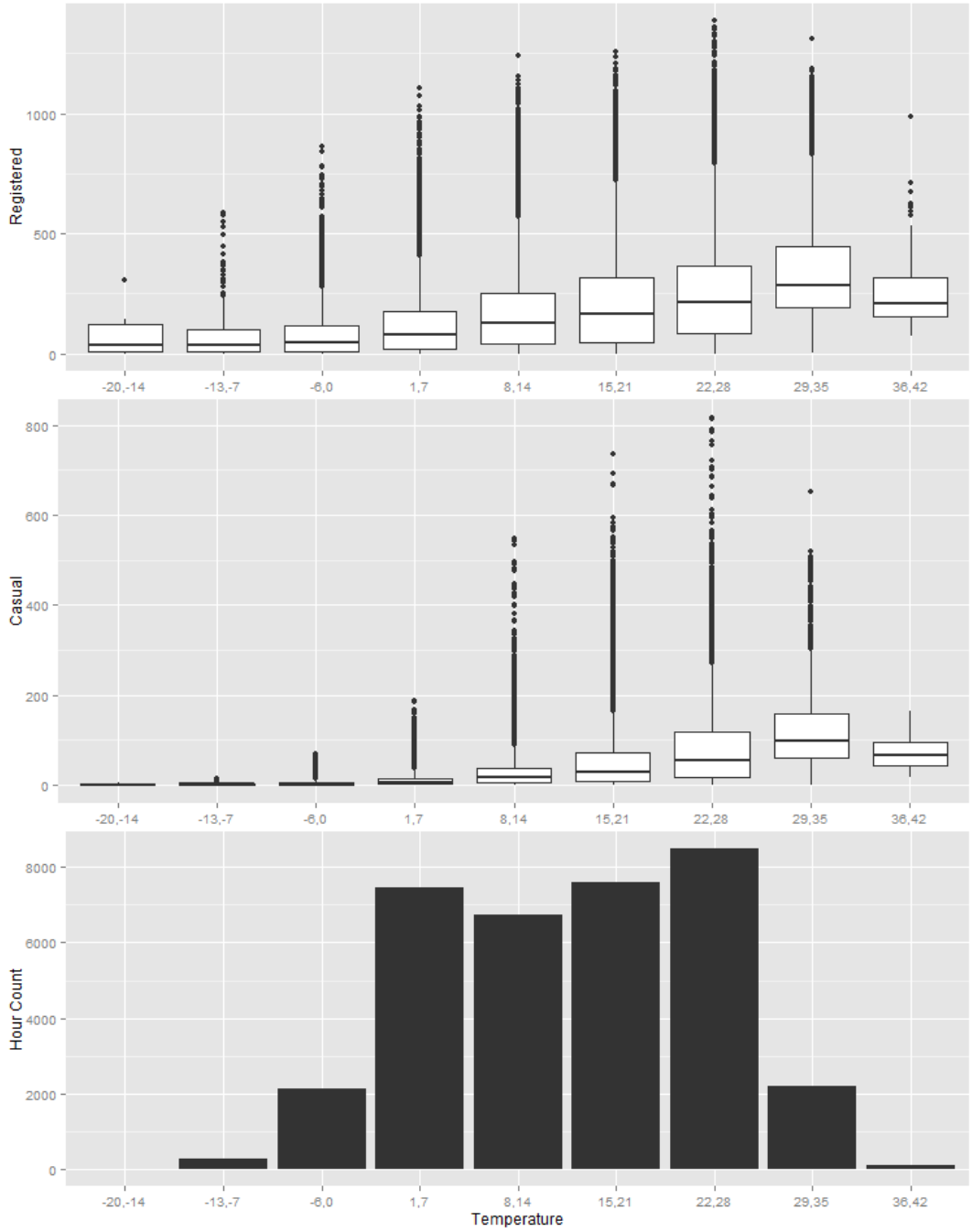

Figure 6. Temperature plots

In both cases the number of rentals increases with temperature up to the 36-42ºC range where it decreases. Optimal temperature range for cycling seems to be 29-35ºC. However, lower temperatures have a more negative impact on casual than on registered users, which could be also observed on the monthly plot.

- Dew point

Temperature at which the water vapor in a sample of air at constant pressure condenses into liquid water at the same rate at which it evaporates, measured in Celsius degrees. Dew point is associated with relative humidity.

The plots look similar to the temperature plots which could indicate high correlation between them.

- Humidity

Amount of water vapor in the air relative to the maximum for that temperature, expressed as a percent. Very low levels of humidity as well as levels close to 100% are uncomfortable for humans.

As with the previous variables, the comfort level of the meteorological variables affects bicycle rentals, but casual rides are more heavily by adverse conditions.

- Pressure

Atmospheric pressure at sea level. Measured in hPa.

- Visibility

Distance at which an object or light can be clearly discerned, measured in Km.

- Wind direction

Direction from which the wind originates. This information is given as a class, featuring 16 different directions (N, NNE, NE, ENE, E, ESE …), plus two other categories (calm and variable).

No insights can be drawn from this plot. Wind distribution is not uniform, but that was to be expected and apparently doesn't have much of an impact on bike rental.

- Wind speed

Flow velocity of the wind, measured in Km/h.

Surprisingly, low wind speeds have a negative impact on bicycle usage which could be caused by some other event correlated with low wind speeds. As expected, high wind speeds also negatively impact bike rentals.

- Precipitation

Amount of water that rained. Measured in mm.

It is clearly visible that an event of rain negatively affects bicycle demands, however data for precipitations over 0.14mm, there is too little data to extract meaningful conclusions.

- Conditions

Description of the weather state (i.e. foggy, overcast, etc.), divided into 25 different categories:

However, 6 of those 25 categories account for 33.689 out of 35.048 observations (96%). These 6 categories are quite similar between them (partly cloudy, scattered clouds, overcast, mostly cloudy). Moreover, some unexpected information is obtained: cloudy days tend to attract more people towards cycling than clear days.

Insights from the data:

Casual and registered users have different bike rental patterns. Registered users behave like commuters: higher usage during weekdays and non-holidays, an hourly trip distribution with two peaks, one in the morning when users go to work and one at the afternoon when they return home. On the other hand casual users register higher usage on holidays and weekends with an hourly distribution without peaks but higher rentals during sunny hours, which leads to thinking they are recreational users.

Moreover, Thomas et al. (2009) found that weather affects leisure usage of bikes more than it affects utilitarian usage, which is the same behavior that can be observed for casual and registered users respectively.

These facts suggested modeling casual and registered user behavior separately and adding the results to get the predicted total count.

8- Algorithm description

8.1- Classification And Regression Trees (CART)

Given a dataset of N observations, each observation $i \in (1, N)$ consisting of an input vector containing the values of p variables $X_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and an output numerical value y_i, tree-based methods partition the input space into rectangles and fit a constant model into each one.

$$f(X) = \sum_{k=1}^{M} c_k \cdot I(X \in R_k) \tag{8.1}$$

Adopting the minimization of the sum of squares $\sum(y_i - f(X_i))^2$ as the decision criterion, it is easy to see that the best c_k is the average of y_i in the rectangle R_k:

$$\hat{c}_k = \text{avg}(y_i | X_i \in R_k) \tag{8.2}$$

Rectangles are made by growing a binary tree: the first node contains all of the observations, which are then split into two new nodes, each of the following nodes is split into two other nodes until each terminal node (called leaf) has a size of n_s observations.

These splits are made by choosing a splitting variable $j \in \{1,2, \dots, p\}$ and a split point s (or two mutually exclusive subsets of classes in the case of categorical variables) and defining the pair of half-planes:

$$R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\} \tag{8.3}$$

At each node the optimal combination of j and s is sought so that solves:

$$\min_{j,s} \left[\min_{c_1} \sum_{x_j \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_j \in R_2(j,s)} (y_i - c_2)^2 \right] \tag{8.4}$$

And as previously stated, for any j and s is solved by:

$$\hat{c}_1 = \text{avg}(y_i | X_i \in R_1(j, s)) \text{ and } \hat{c}_2 = \text{avg}(y_i | X_i \in R_2(j, s)) \tag{8.5}$$

Having found the best split, data is partitioned into two regions and the splitting process is repeated on each of them.

The tree is grown until the minimum node size n_s is reached.

If the tree is used as an individual predictor, pruning is recommended since a full grown tree may overfit the data. Pruning is the process of merging leaves on the same tree branch hence reducing the size of the tree in order to avoid overfitting.

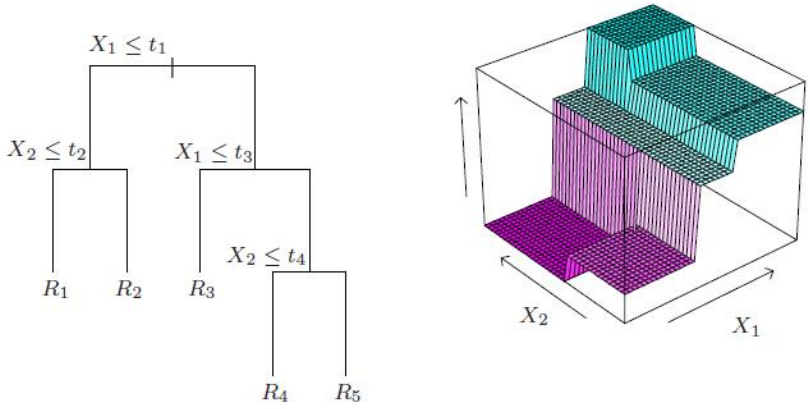

Figure 7. Illustration of the mechanism behind CART (source: Elements of Statistical Learning)

8.2- Bootstrap

Given a training set of N observations, each observation $i \in (1, N)$ consisting of an input vector containing the values of p variables $X_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and an output y_i, bootstrapping is the process of randomly drawing datasets with replacement from the original data, each sample the same size as the original training set (Singh & Xie, 2010).

8.3- Bagging

Bootstrap AGGregation or bagging is done by averaging the prediction (or voting in case of classification) over a collection of bootstrap samples, thereby reducing its variance.

8.4- Random Forest

The random forest algorithm was introduced by Leo Breiman on 2001 in an attempt to design a powerful yet computationally inexpensive predictive tool.

The model consists on drawing B bootstrap samples of the dataset and growing regression (or classification) trees on each of them with the following modifications:

- Trees are grown to full size (until the minimum node size is reached, no pruning).
- At each split, the splitting variable is selected from a sample of m variables chosen at random instead of from all the original variables.

The prediction is made by averaging (or voting) the predictions from all the trees.

Even though the random forest is a non-parametric algorithm it does have three metaparameters:

- Number of trees (B, referred to as ntree in some of the literature and in the randomforest package for R): The accuracy of the ensemble grows as the number of trees increases until it converges and stabilizes.
- Number of variables to choose from at each split (m, referred to as *mtry* in some of the literature and in the randomForest package for R): The correlation between trees

decreases with m (ESL). Breiman suggested to use a value of $\sqrt{p}$ for classification and $p/3$ for regression as default values, however, it has been found that sometimes values closer to p work better. When $m = p$, the random forest algorithm becomes equivalent to unpruned bagging of regression trees.

- Node size (n_s): In his papers Breiman used values of 1 and 5. In general increases on the node size will yield small decreases in both accuracy and computation time.

Apart from being an accurate prediction model, the random forest has some other unique and useful features.

In the original random forest paper Breiman stated that there are two reasons for using bagging: enhancing accuracy (in conjunction with random features) and giving ongoing estimates of the generalization error through out-of-bag estimation. These estimates are explained as follows.

For each observation its outcome is predicted by averaging the predictions of the regression trees for which that observation wasn't included on the bootstrap sample. This is called the out-of-bag regressor. The out-of-bag estimate for the generalization error is the error rate of the out-of-bag regressor on the dataset, and can substitute cross validation as a method of assessing the model's performance.

One of the major criticisms machine learning receives is that although its models deliver accurate predictions they don't shed any light into the inner workings of the problem being solved. This is not true for the random forest.

In his first article about random forests, Breiman described on measure of variable importance that can be obtained from the random forest, later on he added three other measures. In this project only the first and second ones will be considered.

The first measure is also makes use of the out-of-bag data. For each tree the prediction error on the out-of-bag portion is recorded (error for classification, MSE for regression), after that the same is done after permuting each variable. The difference between the two errors is averaged over all trees and normalized by the standard deviation of the differences. This gives a measure of the relative importance of every variable. The second measure works in a similar way, but tracking changes on node purity instead of on MSE.

Random forests can also be used for clustering and outlier detection.

9- Model Tuning

As previously stated, the Random Forest algorithm has 3 metaparameters, namely, minimum node size, number of variables per split and number of trees. The purpose of this section is to describe the method of selecting the best possible set of parameters for the bikesharing system demand forecast problem, which could be described as the parameter configuration that better generalizes from the data. A way to quantify this notion is through the cross validation score. Thus the metaparameters that minimize the cross validation score are the solution to this problem.

The first step in that direction is choosing a loss function. Given the nature of the problem, the Root Mean Squared Error (RMSE) metric was chosen.

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(X_i))^2} \tag{9.1}$$

Other considered loss functions were Root Mean Squared Log Error (RMSLE)

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(f(X_i) + 1))^2} \tag{9.2}$$

And Mean Absolute Error (MAE)

$$\frac{1}{N}\sum_{i=1}^{N}|y_i - f(X_i)| \tag{9.3}$$

However RMSE does a better job at punishing high errors on some observations (compared to MAE) and making the error independent of the scale (compared to RMSLE), which in this case were desired qualities.

The loss function is usually evaluated on a test set, multiple times through cross validation or in the case of a random forest, through the out-of-bag data procedure. These procedures tend to sample data via random processes, which in most cases is a good strategy. Nonetheless, data for this problem has a time component which implies its values can be correlated, and sampling through random procedures would result in an overly optimistic estimate of error. The main issue that was to be addressed was avoiding to use data to train the algorithm that belonged further in time than the test data. To account for this, the cross validation process was manually designed to better suit this problem. The five following train-test splits were made and evaluated:

- First 50% of days as train, following 10% as test.
- First 60% of days as train, following 10% as test.
- First 70% of days as train, following 10% as test.
- First 80% of days as train, following 10% as test.
- First 90% of days as train, last 10% as test.

This approach although not effective in the use of data as k-fold cross validation, was required due to de nature of the problem. A perhaps more suitable approach would have been to train

on an interval of days and test on the following day, repeating the process for a significantly big enough portion of the dataset and averaging the results. Nevertheless, since training the algorithm is the most time consuming task, this approach was dismissed.

Finding the optimal metaparameters is not a classical optimization problem that can be solved via gradient descent, since there isn't an analytical way to describe the cross validation score. Grid search has been suggested (Chih-Wei Hsu, Chih-Chung Chang, 2008) to solve a similar problem (choosing the parameters for a support vector machine), which although being less sophisticated and more computationally expensive it is effective in solving the problem.

The method is quite simple, a grid with different values for each of the parameters is created and the algorithm is evaluated through cross validation on each parameter configuration. The best configuration is chosen, taking mainly into account the cross validation score, but using computation time to decide between close scores. If necessary, a second grid can be used in order to achieve better precision on locating the optimal configuration.

10- First model: Random Forest

As it can be deduced from the algorithm instructions, expected computations time grows linearly with the number of trees and with the number of variables per split. Computation time also grows as the minimum node size decreases.

The used grid contained the sets of parameters:

- Number of trees (B): {1,5,10,20,50,100}
- Number of variables to choose from at each split (m): {1,6,11,16} (being 16 the total number of variables)
- Node size (n_s): {1,5,20}

That amounted to a total of 72 different parameter configurations each of which was cross validated 5 times, for a total of 360 forests grown to tune the casual model's parameters and another 360 to tune the registered model's parameters.

Since these computations and further iterations required heavy computing, a Google Cloud server was used to run the computations, featuring the following specifications: Intel Xenon CPU @ 2.50GHz, 3.75 GB RAM, running Windows Server 2008 R2 Datacenter.

10.1- Modeling casual and registered counts separately

10.1.a) Registered model

The three different plots represent different minimum node sizes, colors represent different values for *m*, the x axis represents the number of trees while the y axis is the cross validation error.

The grid search algorithm showed that in this case values of *m* of 6, 11 and 16 had similar perfomances in terms of error, *m* = 1 gives a significantly worse result.

Error stabilizes for a number of trees of about 20, with higher numbers of trees not accomplishing significant performance improvements.

Minimum nodesize seems to have a big impact on models with a low number of trees, however it has a lower impact on forests with more than 10 trees.

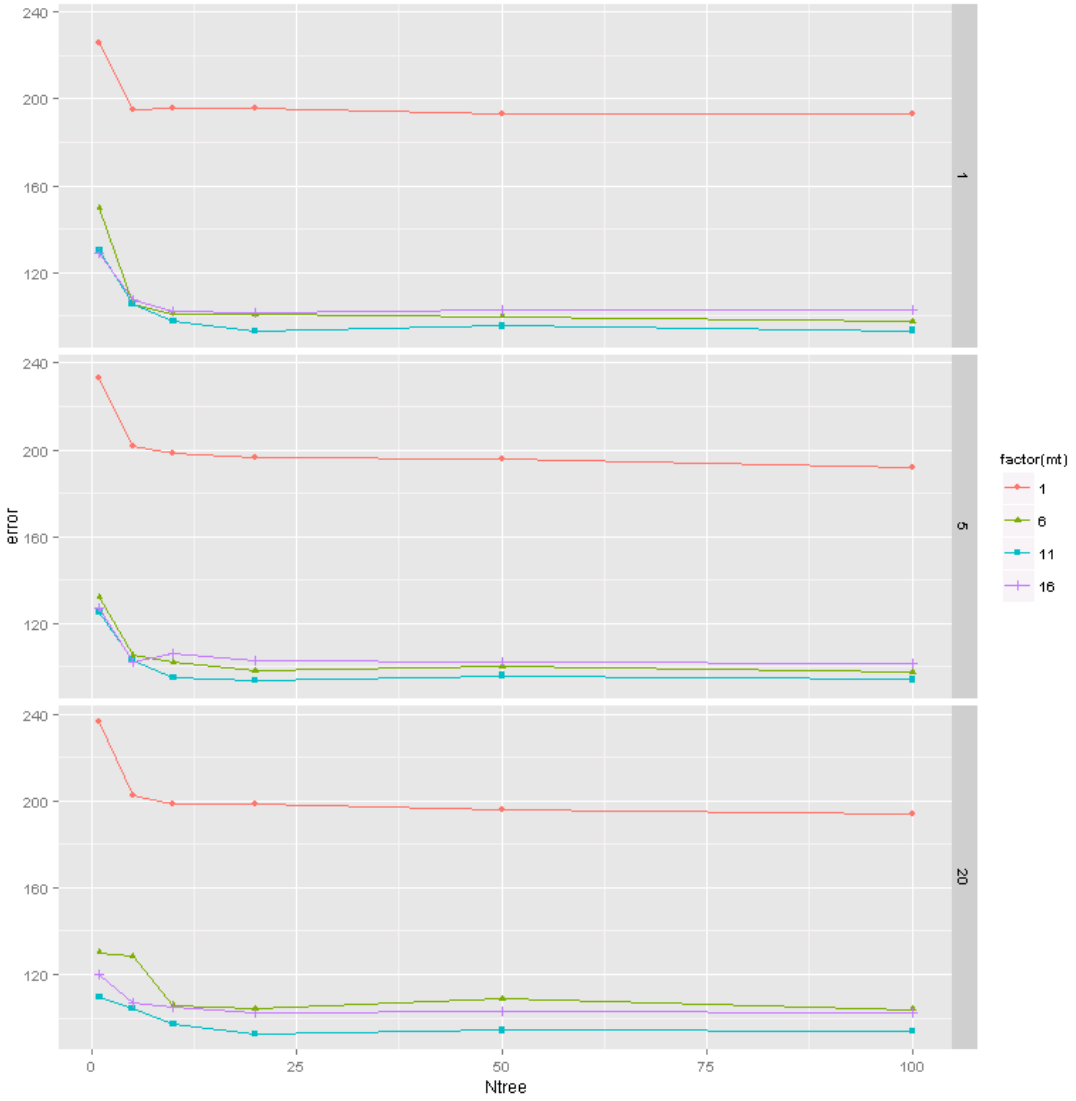

Figure 8. Registered model learning curves

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
20	11	20	114,369	77,283	86,165	80,578	103,393	**92,358**
1	11	20	105,687	79,184	90,876	85,044	104,418	**93,042**
1	11	100	107,539	77,097	94,029	84,572	103,352	**93,318**
5	11	20	108,709	76,811	94,253	85,572	103,880	**93,845**
20	11	100	112,445	76,775	92,526	85,035	102,682	**93,893**
5	11	100	110,246	76,367	94,605	86,330	103,118	**94,133**
20	11	50	114,068	75,898	93,605	84,622	102,599	**94,159**
5	11	10	111,982	79,584	92,976	86,810	103,925	**95,055**
1	11	50	111,792	76,593	101,015	83,021	104,390	**95,362**
5	11	50	111,467	76,717	106,345	82,708	102,546	**95,956**

Table 1. Top10 Metaparameter configurations for the registered model

Having a closer look at the results, *m*=11 is present in all of the top configurations and seems to be better than the alternatives. Other parameters have more even representations.

Since the first option was one of the least computationally expensive, it was selected as the chosen parameter configuration.

10.1.b) Casual model

In this case *m* = 6 clearly performs better that the alternatives with 11 and 16 running up. Like in the registered model *m* = 1 gives a significantly worse result.

Conclusions regarding the number of trees and minumum node size are the same as in the registered model.

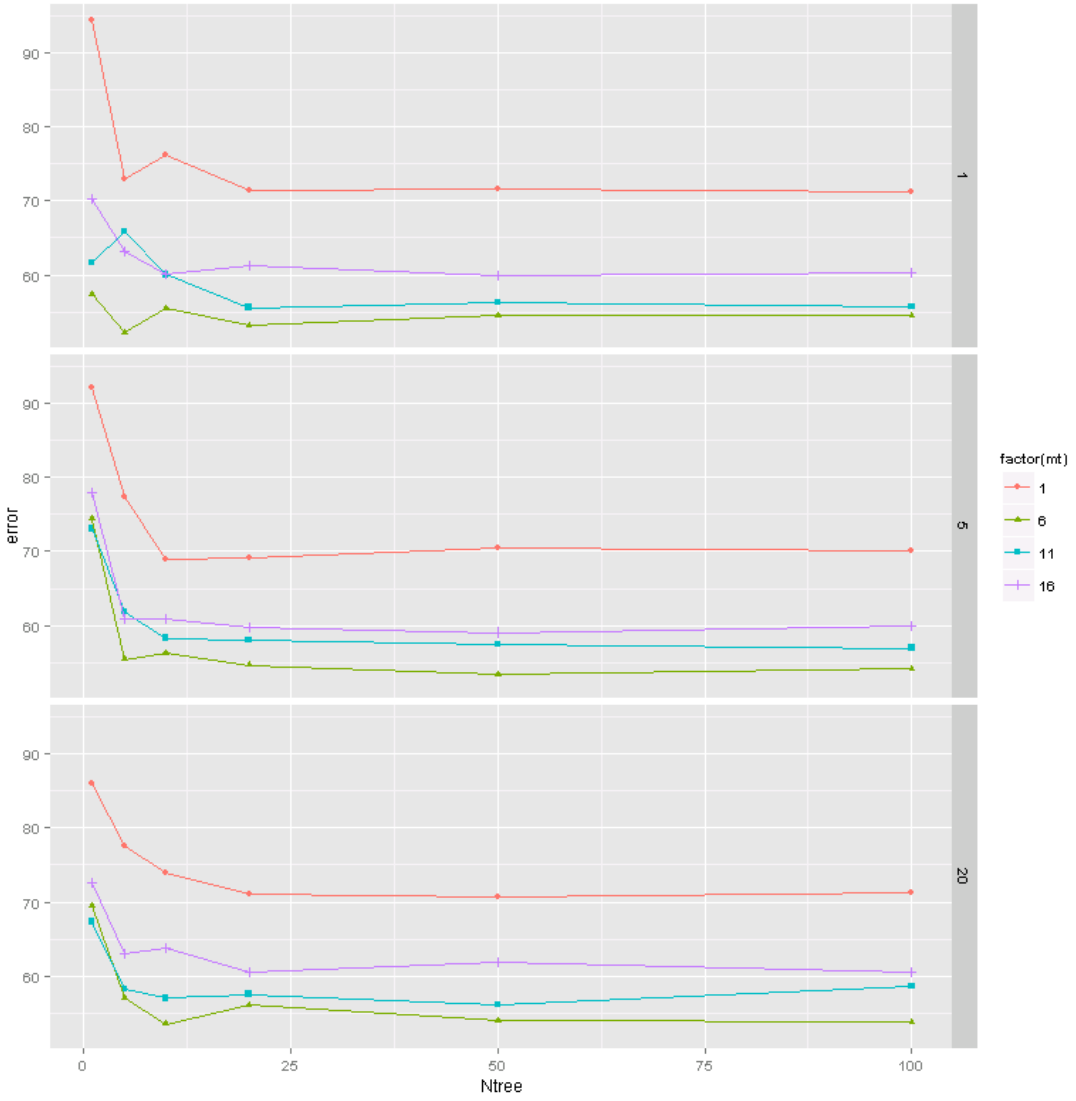

Figure 9. Casual model learning curves

The configuration with the best cross validation result has a very low number of trees, which indicates its good score may have been a matter of chance. Moreover, looking at the RMSE for the different splits it can be seen that its advantage comes from a very good prediction on the third split even though its scores are worse than the second best configurations in all the other splits.

In the light of the above points, the second best configuration was chosen (n_s=1, m=6, B=20).

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
1	6	5	50,202	45,573	29,605	69,115	66,899	**52,279**
1	6	20	49,137	41,591	47,125	62,833	65,623	**53,262**
5	6	50	49,792	41,138	46,391	66,444	63,516	**53,456**
20	6	10	48,847	44,940	40,947	67,500	65,399	**53,527**
20	6	100	49,664	42,301	46,757	64,624	65,450	**53,759**
20	6	50	49,595	41,920	48,347	64,838	65,179	**53,976**
5	6	100	49,418	41,088	49,892	65,794	64,621	**54,163**
1	6	100	48,359	42,420	50,191	66,688	64,860	**54,504**
1	6	50	48,126	40,812	50,603	66,619	66,455	**54,523**
5	6	20	49,252	41,337	52,455	64,262	66,007	**54,663**

Table 2. Top10 Metaparameter configurations for the casual model

10.1.c) Combining models to calculate the total count

Since the total count is the result of the addition of the casual and the registered counts, the predicted total count was calculated as the sum of the predicted registered count and the predicted casual count.

CV1	CV2	CV3	CV4	CV5	CV
136,269	100,607	118,808	125,185	100,077	**116,189**

Table 3. Cross validation errors for the separately predicted model

10.2- Total count model

The interpretation of the results is very similar to that of the registered model.

Looking at the ten best parameter configurations it can be observed that nodesize = 20 and m = 11 yields the best results with a number of trees of 20 or more.

The first parameter configuration was selected.

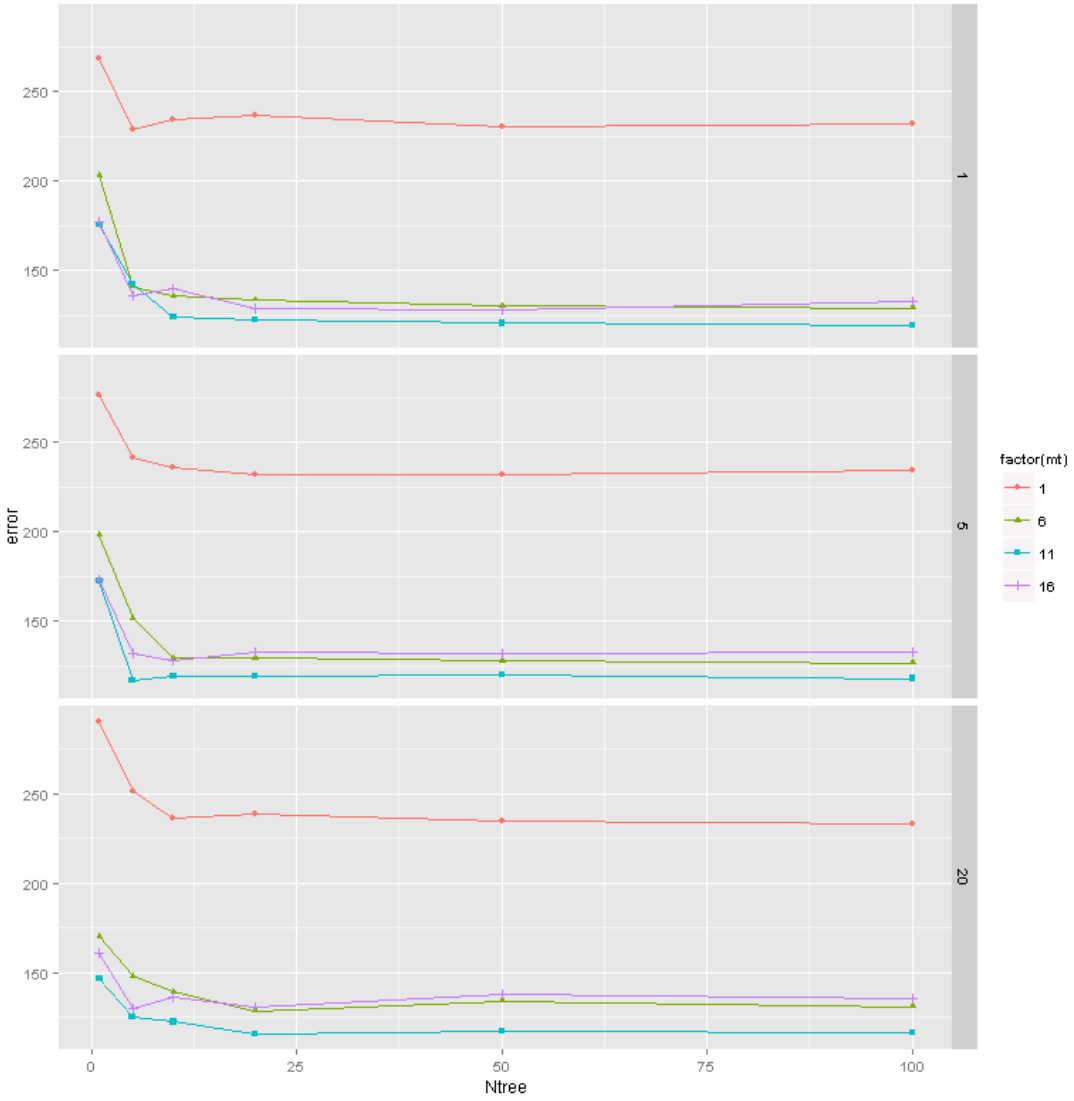

Figure 10. Total count model learning curves

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
20	11	20	135,512	97,823	119,286	121,998	102,349	**115,394**
20	11	100	141,768	97,610	117,964	120,403	103,146	**116,178**
20	11	50	139,094	95,763	126,384	119,897	103,849	**116,997**
5	11	5	131,846	100,58	109,402	131,242	112,131	**117,043**
5	11	100	146,073	95,592	123,778	121,901	103,857	**118,240**
5	11	10	149,725	101,681	104,358	125,308	114,765	**119,167**
1	11	100	146,347	95,525	126,271	124,564	104,336	**119,409**
5	11	20	152,606	97,854	120,216	124,085	103,052	**119,562**
5	11	50	146,394	95,065	131,572	123,416	105,290	**120,348**
1	11	50	149,083	98,894	129,428	122,288	103,368	**120,612**

Table 4. Top10 Metaparameter configurations for the total count model

Both alternatives have similar error rates even though the total count model has a slight edge.

10.3- Improving the model

At this point, the aim was to find where the model failed and improve its performance.

One of the first things that was discovered is the incapability of random forests to predict results higher (or lower) than those of any of the observations, even if there is an easily detectable upward (or downward) trend, like it is with the relationship count-ndays.

This is because of how the random forest algorithm works, making predictions that resemble prior observations. To better understand the phenomenon the total count model was run using the first 3 years as the train set and the last year as the test set. Plots on figures 11 and 12 greatly help to visualize this phenomenon:

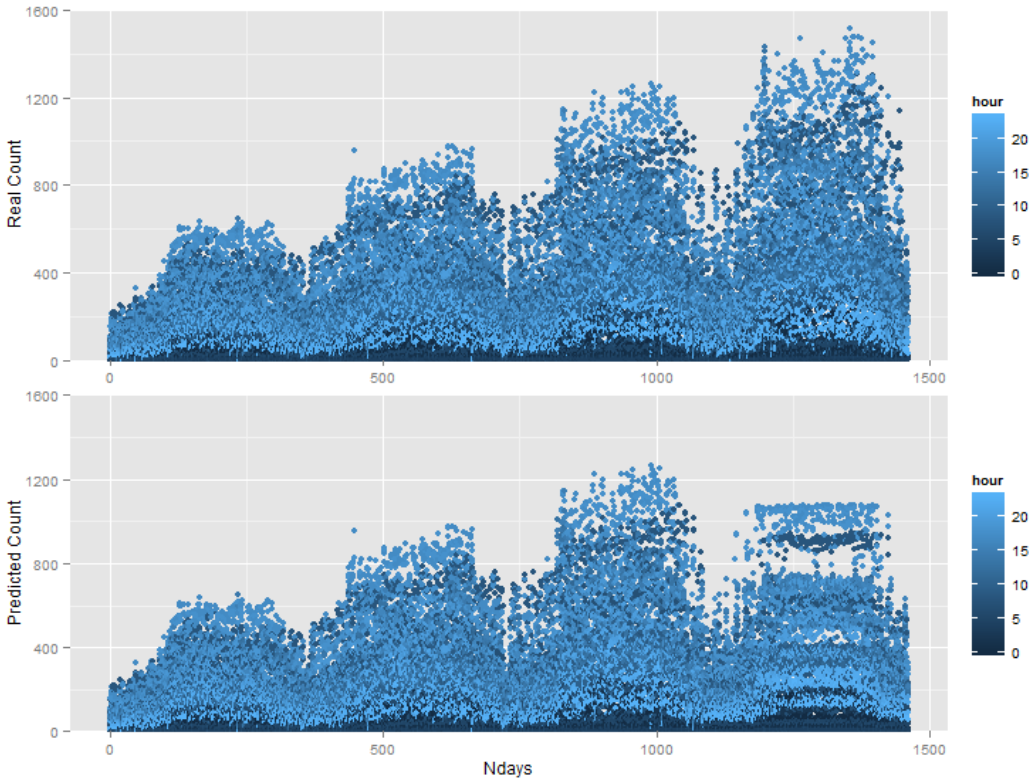

Figure 11. First model, real and predicted count comparison over time

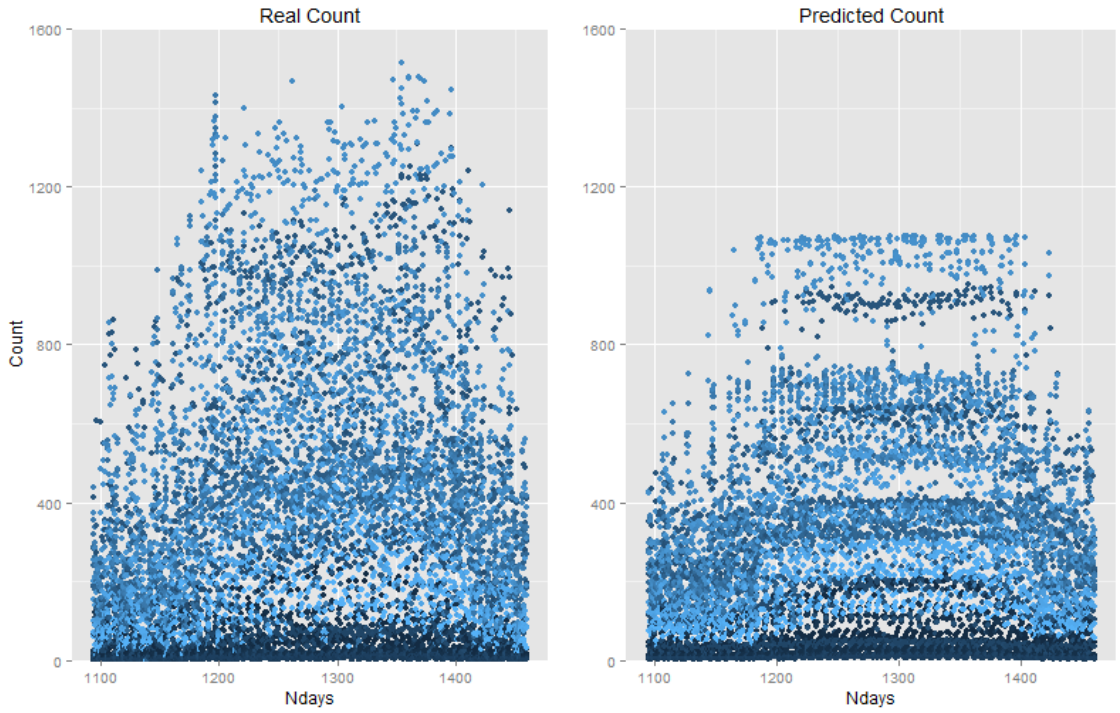

Figure 12. First model, direct comparison between real and predicted counts

From these plots it can be observed that while the model successfully captures the daily and annual cycles it fails to capture the upward trend on time. To solve that problem, a combination of models can be used: one focuses on the trend, while the other takes care of everything else.

11- Linear de-trending and Random Forests

In the light of the conclusions from the last chapter, it was decided to use a linear regression model to de-trend the data:

$$y^* = \frac{y}{a \cdot \text{Ndays} + b} \tag{11.1}$$

Where a and b are the slope and the intercept resulting from fitting a linear model to the data and y^* is the de-trended count. Afterwards, the random forest algorithm was used on the same training data to predict y^* and finally the trend was reintroduced to the data.

$$f(X) = f^*(X) \cdot (a \cdot ndays + b) \tag{11.2}$$

Where $f^*(X)$ is the random forest algorithm applied to the de-trended data and $f(X)$ the predicted count. When running the random forest on the de-trended data, the Ndays variable was substituted by Ydays, the number of days passed since January first of the same year.

This new model seems to work under the assumption of a continuous linear growth on time, which is obviously wrong; at some point the demand growth will slow down and the profile will flatten.

The actual thought process behind the model is that the demand grows as a function of the user base of the system, which is also growing. In this project both of these growths are unknown, but since their combination up until now seems to yield a linear growth of the demand over time, this has been used to de trend the data. However, given access to the user base data, the de-trending process would have been done modeling the demand as a function of user base related variables.

The same grid search process previously described was used to find optimal values for the metaparameters, however, since the plots show very similar shapes to those previously shown, they have been omitted and only the tables with the top results are included. The final count was validated since it is the actual value trying to be predicted, not the de-trended count.

11.1- Modeling casual and registered counts separately

11.1.a) Registered model

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
5	11	100	49,204	50,435	72,206	64,862	87,261	**64,794**
1	11	100	49,487	50,903	72,780	64,964	87,081	**65,043**
5	11	50	49,655	51,091	72,813	65,129	87,333	**65,204**
1	11	50	50,159	51,351	73,109	64,554	87,153	**65,265**
20	11	100	49,678	51,438	74,081	64,989	86,979	**65,433**
20	11	50	49,725	51,503	73,744	65,982	86,925	**65,576**
20	11	20	50,294	52,137	75,273	65,623	87,423	**66,150**
5	11	20	50,827	52,617	73,793	66,274	88,665	**66,435**
1	11	20	51,389	52,967	73,851	66,693	88,544	**66,689**
1	16	100	51,083	52,780	74,158	66,288	89,785	**66,819**

Table 5. Top10 Metaparameter configurations for the de-trended registered model

The de-trending process reduces the average error by about 30%. However, even though it reduces the error on every cross-validation split, it doesn't reduce it in the same rate on each of

them. While the error in the first split is halved, it is only reduced by about 20% on splits 3, 4 and 5.

11.1.b) Casual

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
1	16	50	34,365	34,680	26,419	40,893	62,433	**39,758**
5	11	100	34,190	34,520	25,948	41,966	62,179	**39,761**
5	16	100	34,525	33,985	26,811	41,303	62,351	**39,795**
1	11	100	34,174	34,269	25,742	43,154	61,759	**39,820**
1	6	100	34,977	34,770	24,894	43,258	61,829	**39,946**
1	16	100	34,773	34,874	26,500	41,505	62,109	**39,952**
1	11	50	34,266	35,021	26,269	42,005	62,274	**39,967**
20	11	100	34,593	34,194	26,779	41,987	62,655	**40,042**
5	16	50	34,920	34,314	26,697	41,558	62,824	**40,063**
20	16	100	35,157	33,721	26,788	41,875	62,843	**40,077**

Table 6. Top10 Metaparameter configurations for the de-trended casual model

In this case, the error is reduced by 25%. As in the previous case, the RMSE is reduced in all splits although at different rates. Errors on splits 1 and 4 are reduced by 35-40% while on splits 3 and 5 the error falls by less than 15%.

11.1.c) Combining models to calculate the total count

Since the total count is the result of the addition of the casual and the registered counts, the predicted total count was calculated as the sum of the predicted registered count and the predicted casual count.

CV1	CV2	CV3	CV4	CV5	CV
62,685	63,599	84,715	80,715	87,256	**75,689**

Table 7. Cross validation errors for the de-trended separately predicted model

11.2- Total count model

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
5	11	100	62,900	64,834	86,677	79,669	87,297	**76,275**
1	11	100	62,952	65,533	86,990	79,757	86,469	**76,340**
5	11	50	63,492	65,285	87,390	79,833	87,743	**76,749**
1	11	50	63,551	65,600	86,717	81,047	87,750	**76,933**
20	11	100	63,684	65,207	89,041	80,815	89,118	**77,573**
1	11	20	64,659	65,662	87,651	81,782	89,510	**77,853**
20	11	50	64,558	65,051	88,944	81,108	89,921	**77,916**
1	16	100	65,066	67,957	88,548	81,605	89,619	**78,559**
5	16	100	64,770	67,547	89,154	81,195	90,203	**78,574**
5	11	20	64,304	69,108	88,892	80,550	90,723	**78,715**

Table 8. Top10 Metaparameter configurations for the de-trended total count model

In this case the average error is reduced by 36% and shows a combination of the error reduction patterns shown above.

11.1- Conclusions

The de-trending greatly improved the results on both the combined and the total count model. To try to better understand how the de-trending works, and look for insight on how to improve the model, some interesting relationships were plotted on figures 13 and 14.

On figure 13, it can be observed that when linearly de-trending, counts are all rescaled to the same scale. This wouldn't happen if the trend wasn't linear (another de-trending model would've been required) and also it wouldn't make sense to run the random forest on the de-trended data if it wasn't on a similar scale.

As stated at the beginning of this section, the rescaling of the data would be even better if the model had access to user base data.

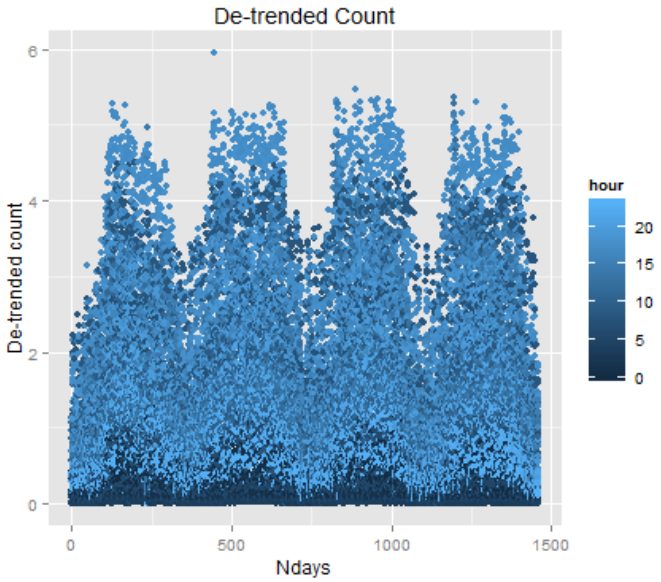

Figure 13. De-trended count over time

Figure 14 compares the real and predicted counts using the linear de-trending model, similarly to what figure 12 does for the initial model. It can be clearly observed that the issues present in the initial model regarding predicted counts reaching a hard cap because of how the random forest algorithm works is no longer present. De-trending and re-trending the data (prior and post using the random forest algorithm respectively) greatly helps to overcome those issues.

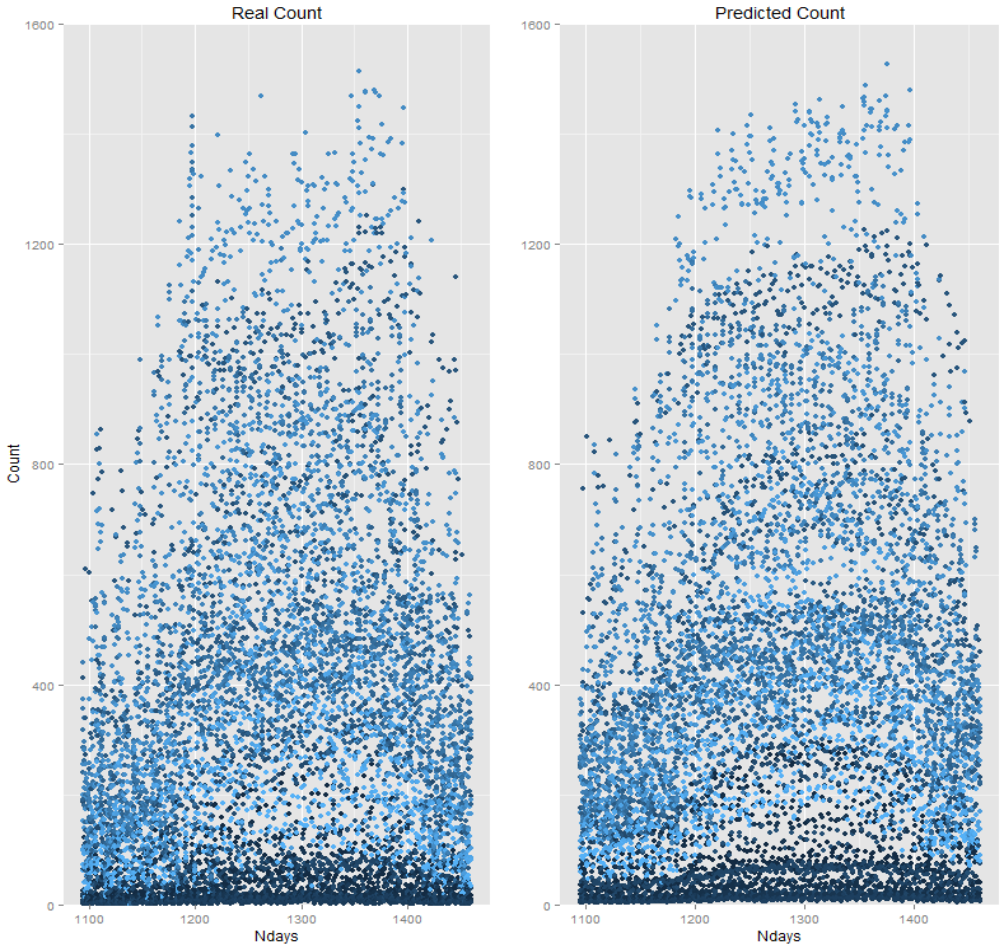

Figure 14. Comparison between real and predicted counts using the de-trend model

Even though the de-trending methodology could be improved, no major flaws on it can be discovered from these plots.

12- Variable importance

Making use of the variable importance tools the random forest algorithm has to offer, the predictive power of each variable was explored, in order to know which the most important variables were. The increase in Mean Squared Error and increase in node purity variable importance measures were used. Figure 15 shows the results of calculating variable importance on the linear regression de-trending plus random forest model, applied to directly predicting the total counts.

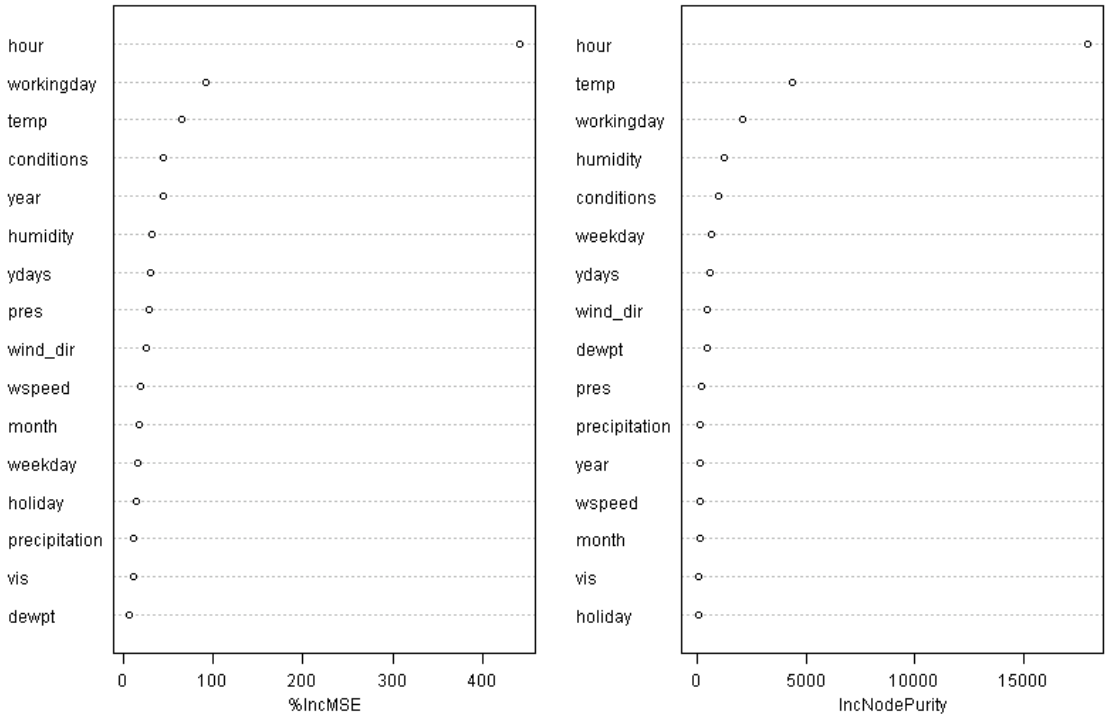

Figure 15. Variable importance plot

It can be easily observed that both measures are very similar but not exactly the same in this case. Hour is by far the most important variable, followed by workingday and temperature. Other variables have been shown to be far less important.

One of the reasons for the low predictive power of some variables may be that their predictive power is split between some variables (weather variables may all contain similar information) and thus each of those variables are chosen less frequently on the random forest splits even if as a whole they are chosen very often. Having 9 weather variables, they are probably related between them and thus their predictive power is split among them, making each of them rank relatively low on variable importance.

13- Feature engineering

Under the hypothesis that not only instant weather variables affect the bikeshare users behavior, but their behavior is also affected by weather fluctuations during the day; some new weather variable were created.

For each numeric weather variable (temperature, dew point, humidity, precipitation, wind speed, visibility) new variables were created to capture daily fluctuations: daily maximums, daily minimums and daily averages.

The random forest algorithm was applied to the linearly de-trended data to directly predict the total counts. Metaparmeter optimization was run using $B = 50$ and $n_s = 5$, varying only the values for m. The results are included in the following table:

n_s	m	B	CV1	CV2	CV3	CV4	CV5	CV
5	30	50	64,658	63,867	88,090	79,611	89,532	**77,151**
5	22	50	65,117	63,520	89,125	79,194	90,140	**77,419**
5	14	50	75,972	74,699	95,259	89,921	97,641	**86,698**
5	4	50	138,010	154,027	139,961	179,948	178,086	**158,006**

Table 9. Metaparameter configurations for the de-trended total count model using the new daily variables

No forecast accuracy was gained from those extra variables. As shown on the previous section, predictive power was already split between weather variables so it is logical in a way that the addition of more weather variables causes no increase on prediction power.

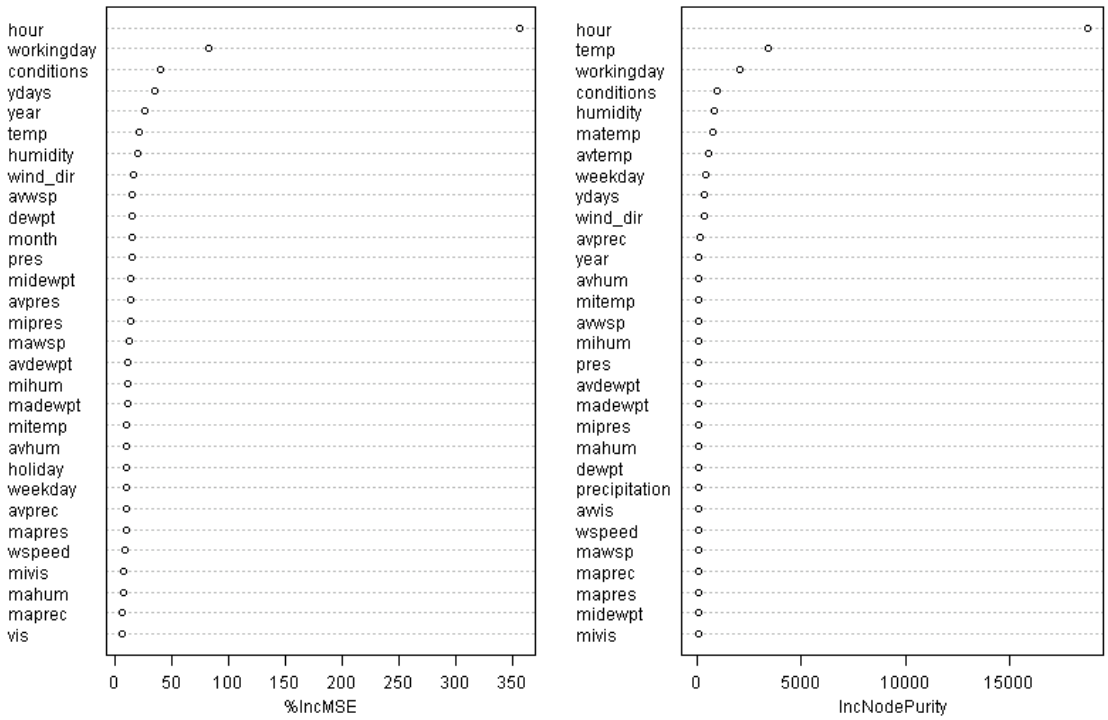

Figure 16. Variable importance plot

Figure 16 shows that the addition of more weather variable caused the predictive power of weather variables to further split among them, making them individually rank lower.

Since the addition of those extra variables doesn't increase predictive accuracy while it does increase computation time, it is fair to state that the model works better without those extra variables.

14- Conclusions and further steps

14.1- Conclusions regarding the regression model

With reference to the regression model itself and the obtained results, one could state the following conclusions:

1. Machine learning techniques and specifically the random forest have a great potential in the field of Civil Engineering and transportation. If there is a wealth of data available, machine learning algorithms can be used to make very accurate forecasts.
2. Even if it feels a bit counterintuitive, for the initial model (only random forest) and the de-trended random forest; predicting the total count using two separate models for casual and registered users and adding them up gives a very similar accuracy to predicting it using a single model. However, these two approaches aren't exactly the same. The model that separately forecasts registered and casual users can be used to better understand each type of user's behavior. On the other hand, the single model approach is half as computationally expensive to train and optimize. Choice between these two options should be made depending on the objective of the model.
3. A major flaw in the random forest algorithm is its inability to capture trends shown in the training data if the test set has higher or lower values for the variable driving the trend that those on the train set. This can be fixed by de-trending if the trend can be easily observed or its cause is previously known as shown in this project. This applies to monotonic increasing or decreasing trends in general, not only temporal trends.
4. Additional weather variables built using the existing ones to account for weather fluctuations during the day doesn't help improve the model's accuracy.

14.2- Conclusions regarding the project as a whole

After assessing the accuracy of the model through the regression model conclusions, the potential usefulness of those results can be addressed. Both sets of conclusions have been separated because these conclusions are in a way consequence of the former.

Forecast methods and particularly those fueled by data and engineered using machine learning techniques have shown to be a very useful tool on the operation of a bicycle sharing system.

The models described in this project take a bit of time to tune to find optimal parameters but once those parameters are optimized, they can be quickly trained and deployed. The linear regression de-trending in combination with the random forest model could be used to predict demands for bikes as well as demand for parking spots on each station (using a different model for each of those demands). That would allow the bikeshare system operator to anticipate those demands using a dynamic repositioning system, easing one of the major causes of customer loss for bikesharing systems.

Moreover, this models could also be applied to bicycle balance incentive systems in order to design the incentive model, also combining parking slots and bike demands to predict which stations will be in need of more bicycles and which won't. Also, machine learning techniques could be developed to measure the incentive system's effectiveness.

This project is a prime example of the application of modern analytical methodologies to civil engineering and transportation problems in order to generate value by improving the experience of transport system's users.

16- Bibliography

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection, *4*, 40–79. http://doi.org/10.1214/09-SS054

Bergmeir, C. (2015). A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction, (April).

Bhattacharya, B., Price, R. K., & Solomatine, D. P. (2007). Machine Learning Approach to Modeling Sediment Transport. *Journal of Hydraulic Engineering*, *133*(4), 440–450. http://doi.org/10.1061/(ASCE)0733-9429(2007)133:4(440)

Borgnat, P., Abry, P., & Rouquier, P. F. J. (2009). Studying Lyon ' s V ´ elo ' V : A Statistical Cyclic Model.

Breiman, L. (2001). Random Forrest. *Machine Learning*, 1–33. http://doi.org/10.1023/A:1010933404324

Breiman, L. (2002a). Looking Inside the Black Box, 1–35. http://doi.org/10.1080/10508610802471096

Breiman, L. (2002b). Machine Learning, 1–39.

Breiman, L. (2002c). Manual on setting up, using, and understanding random forests v3. 1. *Technical Report, Http://oz.berkeley.edu/users/breiman, Statistics Department University of California Berkeley, …*, 29. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Manual+On+Setting+Up,+Using,+And+Understanding+Random+Forests+V3.1#0

Breiman, L. (2002d). Software for the Masses (and an Example), 1–29.

Caggiani, L., & Ottomanelli, M. (2013). A Dynamic Simulation based Model for Optimal Fleet Repositioning in Bike-sharing Systems. *Procedia - Social and Behavioral Sciences*, *87*, 203–210. http://doi.org/10.1016/j.sbspro.2013.10.604

Caruana, R., & Niculescu-mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. *In Proc. 23 Rd Intl. Conf. Machine Learning (ICML `06)*, 161–168. http://doi.org/10.1145/1143844.1143865

Chih-Wei Hsu, Chih-Chung Chang, and C.-J. L. (2008). A Practical Guide to Support Vector Classification. *BJU International*, *101*(1), 1396–400. http://doi.org/10.1177/02632760022050997

Daddio, D. (2012). Maximizing Bicycle Sharing: an empirical analysis of capital bikeshare usage. *Master's Thesis at UNCCH*. Retrieved from http://rethinkcollegepark.net/blog/wp-content/uploads/2006/07/DaddioMP_Final-Draft.pdf

Froehlich, J., Neumann, J., & Oliver, N. (2009). Sensing and predicting the pulse of the city through shared bicycling. *IJCAI International Joint Conference on Artificial Intelligence*, (2), 1420–1426. http://doi.org/10.1.1.150.4370

Genuer, R., Poggi, J.-M., & Tuleau, C. (2008). Random Forests: some methodological insights. Retrieved from http://arxiv.org/abs/0811.3619

Herrera, M., Torgo, L., Izquierdo, J., & Pérez-García, R. (2010). Predictive models for forecasting hourly urban water demand. *Journal of Hydrology*, *387*(1-2), 141–150. http://doi.org/10.1016/j.jhydrol.2010.04.005

Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J., & Banchs, R. (2010). Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, *6*(4), 455–466. http://doi.org/10.1016/j.pmcj.2010.07.002

Krstajic, D., Buturovic, L. J., Leahy, D. E., & Thomas, S. (2014). Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of Cheminformatics*, *6*(1), 1–15. http://doi.org/10.1186/1758-2946-6-10

Liu, Z., Sadiq, R., Rajani, B., & Najjaran, H. (2010). Exploring the Relationship between Soil Properties and Deterioration of Metallic Pipes Using Predictive Data Mining Methods. *Journal of Computing in Civil Engineering*, *24*(3), 289–301. http://doi.org/10.1061/(ASCE)CP.1943-5487.0000032

Oliveira, S., Oehler, F., San-Miguel-Ayanz, J., Camia, A., & Pereira, J. M. C. (2012). Modeling spatial patterns of fire occurrence in Mediterranean Europe using Multiple Regression and Random Forest. *Forest Ecology and Management*, *275*, 117–129. http://doi.org/10.1016/j.foreco.2012.03.003

Raviv, T., & Kolka, O. (2013). Optimal inventory management of a bike-sharing station. *IIE Transactions*, *45*(10), 1077–1093. http://doi.org/10.1080/0740817X.2013.770186

Shaheen, S. A. (2015). *Public Bikesharing in North America: Understanding Impacts, Business Models, and Equity Effects of Bikesharing Systems During Rapid Industry Expansion | MTI Research in Progress*. Retrieved from http://transweb.sjsu.edu/MTIportal/research/projects/rpd/rpd1131.html

Singh, K., & Xie, M. (2010). Bootstrap : A Statistical Method. *International Encyclopedia of Education*, 46–51. http://doi.org/www.stat.rutgers.edu/~mxie/rcpapers/bootstrap.pdf

Singla, A., Santoni, M., Meenen, M., & Krause, A. (2015). Incentivizing Users for Balancing Bike Sharing Systems.

Thomas, T., Jaarsma, R., & Tutert, B. (2009). Temporal variations of bicycle demand in the Netherlands : The influence of weather on cycling.

Vogel, P., Greiser, T., & Mattfeld, D. C. (2011). Understanding bike-sharing systems using Data Mining: Exploring activity patterns. *Procedia - Social and Behavioral Sciences*, *20*, 514–523. http://doi.org/10.1016/j.sbspro.2011.08.058

16.1- Software used

R Development Core Team (2008). R: A language and environment for statistical computing. Available at http://www.R-project.org.

PostgreSQL Global Development Group. PostgreSQL Reference version 9.4. Available at http://www.postgresql.org

Python Software Foundation. Python Language Reference, version 3.4. Available at http://www.python.org

17- Appendix A: SQL Code

```
-- load bicycle data

CREATE TABLE tmp1 (
    Duration            varchar(80),
    Start_date          varchar(80),
    End_date            varchar(80),
    Start_station       varchar(80),
    End_station         varchar(80),
    Bike               varchar(80),
    Member_Type         varchar(80)
);
CREATE TABLE tmp2 (
    Duration            varchar(80),
    Start_date          varchar(80),
    Start_station       varchar(80),
    End_date            varchar(80),
    End_station         varchar(80),
    Bike        varchar(80),
    Member_Type     varchar(80)
);
CREATE TABLE tmp3 (
    Duration            varchar(80),
    Start_date          varchar(80),
    Start_station       varchar(80),
    End_date            varchar(80),
    End_station         varchar(80),
    Bike               varchar(80),
    Member_Type         varchar(80)
);

COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2010-Q4-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2011-Q1-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2011-Q2-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2011-Q3-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2011-Q4-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2012-Q1-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2012-Q2-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2012-Q3-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2012-Q4-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2013-Q1-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2013-Q2-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp1 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2013-Q3-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;
COPY tmp2 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2013-Q4-Trips-
History-Data2.csv' DELIMITER ',' CSV HEADER;
COPY tmp2 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2014-Q1-Trips-
History-Data2.csv' DELIMITER ',' CSV HEADER;
```

```
COPY tmp2 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2014-Q2-Trips-
History-Data2.csv' DELIMITER ',' CSV HEADER;
COPY tmp2 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2014-Q3-Trips-
History-Data3.csv' DELIMITER ',' CSV HEADER;
COPY tmp3 FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/2014-Q4-Trips-
History-Data.csv' DELIMITER ',' CSV HEADER;

drop table bike_rides1;
create table bike_rides1 as
    select   cast( coalesce(substring(Start_date from '#"%#"/%/%' for
'#'), substring(Start_date from '#"%#"-%-%' for '#'), '-1' ) as integer)
as month,
        cast( coalesce(substring(Start_date from '%/#"%#"/%' for '#'),
substring(Start_date from '%-#"%#"-%' for '#'), '-1' ) as integer) as
day,
        cast( coalesce(substring(Start_date from '%/#"%#" %'  for '#'),
substring(Start_date from '%-#"%#" %'  for '#'), '-1' ) as integer) as
year,
        cast( coalesce(substring(Start_date from '% #"%#":%' for '#'),
substring(Start_date from '% #"%#"-%' for '#'), '-1' ) as integer) as
hour,
        Member_Type,
        Start_date
    from tmp1;

drop table bike_rides2;
create table bike_rides2 as
    select   cast( coalesce(substring(Start_date from '#"%#"/%/%' for
'#'), substring(Start_date from '#"%#"-%-%' for '#'), '-1' ) as integer)
as month,
        cast( coalesce(substring(Start_date from '%/#"%#"/%' for '#'),
substring(Start_date from '%-#"%#"-%' for '#'), '-1' ) as integer) as
day,
        cast( coalesce(substring(Start_date from '%/#"%#" %'  for '#'),
substring(Start_date from '%-#"%#" %'  for '#'), '-1' ) as integer) as
year,
        cast( coalesce(substring(Start_date from '% #"%#":%' for '#'),
substring(Start_date from '% #"%#"-%' for '#'), '-1' ) as integer) as
hour,
        Member_Type,
        Start_date
    from tmp2;

drop table bike_rides3;
create table bike_rides3 as
    select   cast( coalesce(substring(Start_date from '%/#"%#"/%' for
'#'), substring(Start_date from '%-#"%#"-%' for '#'), '-1' ) as integer)
as month,
        cast( coalesce(substring(Start_date from '%/#"%#" %'  for '#'),
substring(Start_date from '%-#"%#" %'  for '#'), '-1' ) as integer) as
day,
        cast( coalesce(substring(Start_date from '#"%#"/%/%' for '#'),
substring(Start_date from '#"%#"-%-%' for '#'), '-1' ) as integer) as
year,
        cast( coalesce(substring(Start_date from '% #"%#":%' for '#'),
substring(Start_date from '% #"%#"-%' for '#'), '-1' ) as integer) as
hour,
        Member_Type,
        Start_date
    from tmp3;
```

```
-- join the 3 tables;

drop table bike_rides;
CREATE TABLE bike_rides (
    month           integer,
    day             integer,
    year            integer,
    hour        integer,
    Member_Type     varchar(80),
    start_date      varchar(80)
);

insert into bike_rides table bike_rides1;
insert into bike_rides table bike_rides2;
insert into bike_rides table bike_rides3;

create table ride_counts as
    select day, month, year, hour,
        sum( case when Member_Type = 'Casual' then 1 else 0 end) as
Casual,
        sum( case when Member_Type in ('Registered', 'Subscriber') then
1 else 0 end) as Registered
    from bike_rides
    where day != 0 and year > 2010
    group by day, month, year, hour

create table days as
    select distinct day, month, year
    from ride_counts
    where year > 2010
    order by year, month, day;

copy (select * from days) to 'C:/Users/Oriol/Documents/Tesina/RAW
data/days.csv' With CSV;


-- weather data from wunderground

CREATE TABLE weather_1 (
    year        integer,
    month       integer,
    day         integer,
    hour            varchar(80),
    temp            decimal,
    DewPt           decimal,
    humidity            varchar(80),
    pres            decimal,
    vis         decimal,
    wind_dir        varchar(80),
    wind_sp     varchar(80),
    wind_raf        varchar(80),
    precipitation   varchar(80),
    events      varchar(80),
    conditions      varchar(80),
    wdirdeg     varchar(80),
    a           varchar(80)
);

COPY      weather_1      FROM      'C:/Users/Oriol/Documents/Tesina/RAW
data/weather.txt' DELIMITER ',';
```

```
create table weather_2 as
    select  year,
        month,
        day,
        cast( coalesce(substring(hour from '#"%#":%' for '#'),  '-1' )
as integer)
            + case when substring(hour from '% #"%#"M' for '#') = 'P'
and not cast( coalesce(substring(hour from '#"%#":%' for '#'),  '-1' )
as integer) = 12 then 12
                when ( substring(hour from '% #"%#"M' for '#') = 'A' and
cast( coalesce(substring(hour from '#"%#":%' for '#'),  '-1' ) as
integer) = 12) then - 12
                else 0 end  as hour,
        cast( coalesce(substring(hour from '%:#"% #"%' for '#'),  '-1'
) as numeric) as minutes,
        temp,
        DewPt,
        humidity,
        pres,
        vis,
        wind_dir,
        wind_sp,
        wind_raf,
        precipitation,
        events,
        conditions,
        wdirdeg
    from weather_1;

create table weather_hp1 as
    select   case when hour < 23 then year else date_part('year',
(cast(year||'-'||month||'-'||day as date)+1)) end as year,
        case when hour < 23 then month else date_part('month',
(cast(year||'-'||month||'-'||day as date)+1)) end as month,
        case when hour < 23 then day else date_part('day', (cast(year||'-
'||month||'-'||day as date)+1)) end as day,
        case when hour < 23 then hour + 1 else 0 end as hour,
        60 - minutes + 0.5 as minutes,
        temp,
        DewPt,
        humidity,
        pres,
        vis,
        wind_dir,
        wind_sp,
        wind_raf,
        precipitation,
        events,
        conditions,
        wdirdeg
    from weather_2

insert into weather_2 table weather_hp1;

create table min as
    select min(minutes) as minutes, hour, day, month, year
    from weather_2
    group by hour, day, month, year;

create table weather_3 as
    select a.*
```

```
    from weather_2 as a
    inner join min as b
        on a.minutes = b.minutes and a.hour = b.hour and a.day=b.day and
a.month=b.month and a.year=b.year;

create table weather_4 as
    select year, month, day, hour, temp, dewpt, humidity, pres, vis,
wind_dir,
        case when wind_sp = 'Calm' then 0 else cast(wind_sp as numeric)
end as wspeed,
        case when precipitation = 'N/A' then 0 else cast(precipitation
as numeric) end as precipitation,
        conditions
    from weather_3;

create table weather as
    select year, month, day, hour,
        max(temp) as temp,
        max(DewPt) as DewPt,
        max(humidity) as humidity,
        max(pres) as pres,
        max(vis) as vis,
        max(wind_dir) as wind_dir,
        max(wspeed) as wspeed,
        max(precipitation) as precipitation,
        max(conditions) as conditions
    from weather_5
    group by year, month, day, hour;

-- get the holidays

CREATE TABLE holidays_1 (
    weekday     varchar(80),
    d_m         varchar(80),
    y_o         varchar(80)
);

COPY      holidays_1      FROM      'C:/Users/Oriol/Documents/Tesina/RAW
data/holidays.txt' DELIMITER ',';

create table holidays_2 as
    select substring(d_m from ' #"%#" %' for '#') as month,
        cast( substring(d_m from ' % #"%#"' for '#') as integer) as day,
        cast( substring(y_o from '%#"20__#"%' for '#') as integer) as
year,
        1 as holiday
    from holidays_1;

create table holidays as
    select case when month = 'January' then 1 when month = 'February'
then 2 when month = 'March' then 3
            when month = 'April' then 4 when month = 'May' then 5 when
month = 'June' then 6
            when month = 'July' then 7 when month = 'August' then 8 when
month = 'September' then 9
            when month = 'October' then 10 when month = 'November' then
11 else 12 end as m_num,
        month,
        day,
        year,
        holiday
```

```
    from holidays_2

-- Joining the data

CREATE TABLE hours (
    h       integer
);

COPY hours FROM 'C:/Users/Oriol/Documents/Tesina/RAW data/hours.csv'
DELIMITER ',';

create table days_hours as
    select h as hour, day, month, year
    from days as a
    full outer join hours as b
        on TRUE;

select distinct year from days_hours;

create table ride_counts_2 as
    select a.hour, a.day, a.month, a.year, coalesce(casual, 0) as
casual, coalesce(registered, 0) as registered
    from days_hours as a
    left join ride_counts as b
        on a.year = b.year and a.month = b.month and a.day = b.day and
a.hour = b.hour;

create table w_h as
    select a.*, coalesce(b.holiday, 0) as holiday
    from weather as a
    left join holidays as b
        on a.month = b.m_num and a.day = b.day and a.year = b.year;

create table w_h_b as
    select coalesce(a.year, b.year) as year,
        coalesce(a.month, b.month) as month,
        coalesce(a.day, b.day) as day,
        coalesce(a.hour, b.hour) as hour,
        coalesce(temp, -9999) as temp,
        DewPt,
        humidity,
        pres,
        vis,
        wind_dir,
        wspeed,
        precipitation,
        conditions,
        holiday,
        coalesce(a.casual, 0) as casual,
        coalesce(a.registered, 0) as registered
    from ride_counts_2 as a
    left join w_h as b
        on a.year = b.year and a.month = b.month and a.day = b.day and
a.hour = b.hour;

create table all_data as
    select *
    from w_h_b
    where not ((temp = -9999) or (dewpt = -9999) or (humidity = 'N/A')
or (pres = -9999) or (vis = -9999.0) or (wspeed = -9999.0));
```

```
copy (select * from all_data) to 'C:/Users/Oriol/Documents/Tesina/RAW
data/all_data.csv' With CSV header;
```

18- Appendix B: Python Code

```
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        return False

import numpy
days = numpy.zeros((1461, 3))

f = open("days.csv","r")
x =f.read()
k = 0

with open("days.csv","r") as f:
    for line in f:
        text = str(line)
        m = len(text)
        i = 0
        j = 0
        while i < m :
            if is_number(text[i]) :
                y = int(text[i])
                i += 1
                while is_number(text[i]) :
                    y = 10*y + int(text[i])
                    i += 1
                days[k][j] = y
                j += 1
            i += 1
        k += 1

import urllib.request
file = open("weather.txt", "w")

for i in range(1461) :
    response =
urllib.request.urlopen('http://www.wunderground.com/history/airport/KDCA/'+str(int(days[i][2]))+'/'+str(int(days[i][1]))+'/'+str(int(days[i][0]))+'/DailyHistory.html?req_city=Washington%20D.C.&req_statename=District%20of%20Columbia&format=1')

    x = str(response.read())
    m = len(x)
    j = 0
    while (not is_number(x[j])) :
         j += 1
    while j < m-10 :
        st = str(int(days[i][2])) + ', ' + str(int(days[i][1])) + ', '
+ str(int(days[i][0])) + ', '
        while (x[j] != "<") :
            st = st + x[j]
            j += 1
        file.write(st + "\n")
        j += 8

file.close()
```

19- Apendix C: Variable analysis plots

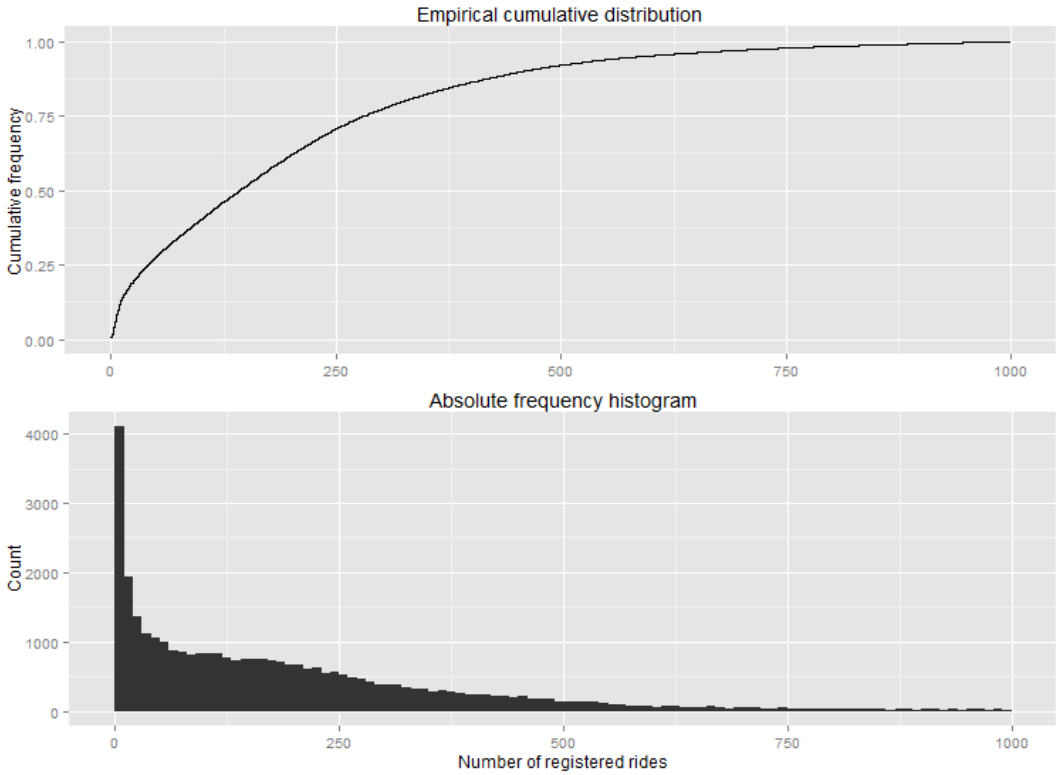

Figure 17. Registered users plots

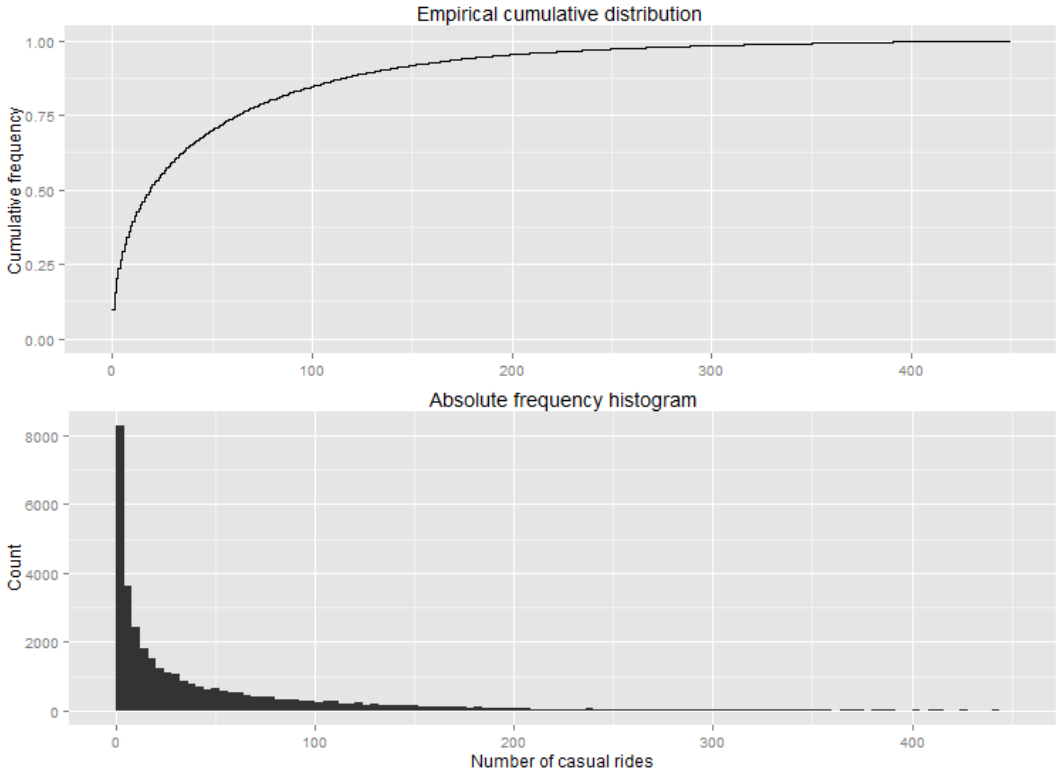

Figure 18. Casual users plots

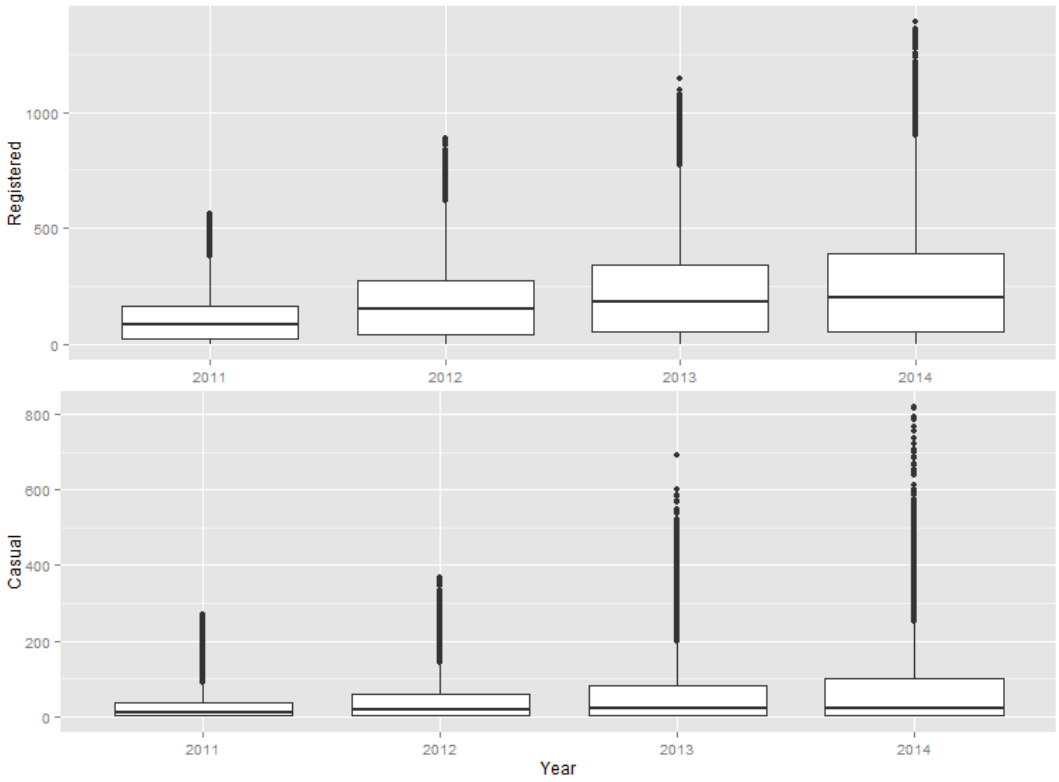

Figure 19. Year plots

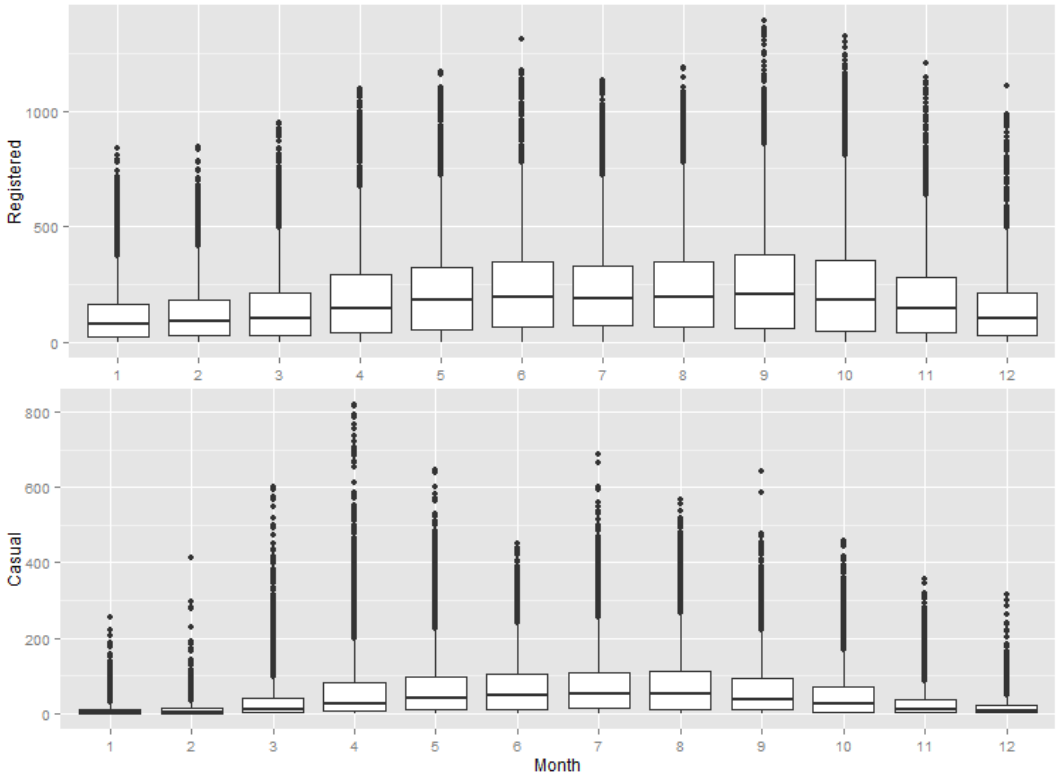

Figure 20. Month plots

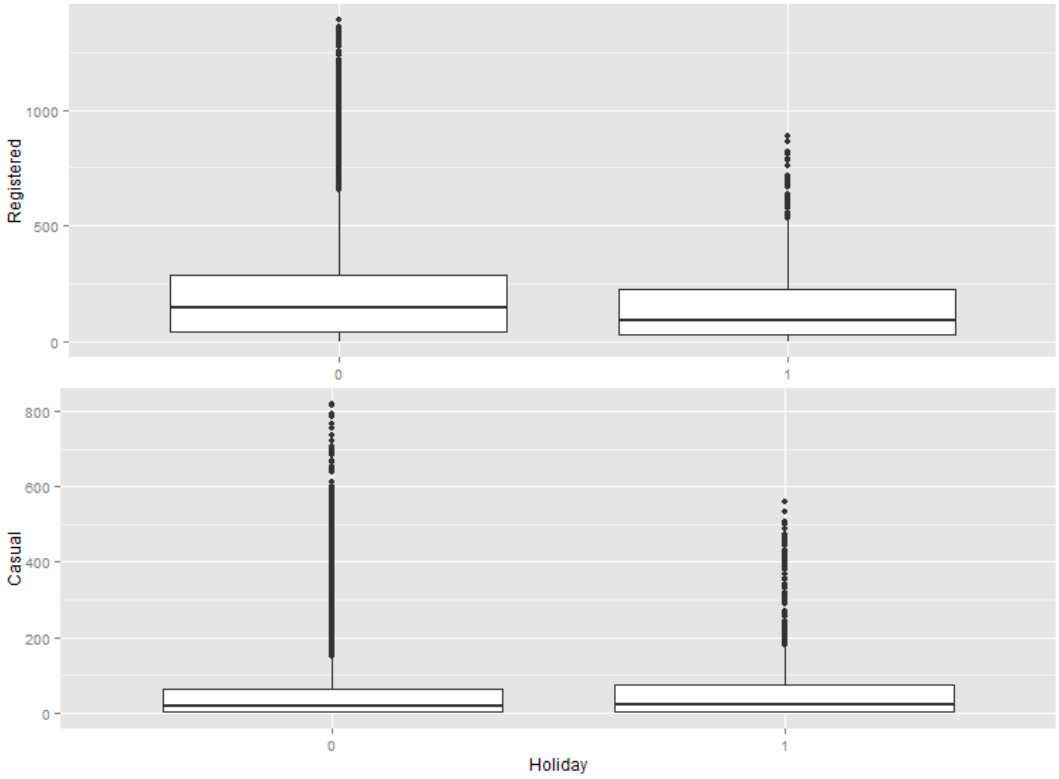

Figure 21. Holiday plots

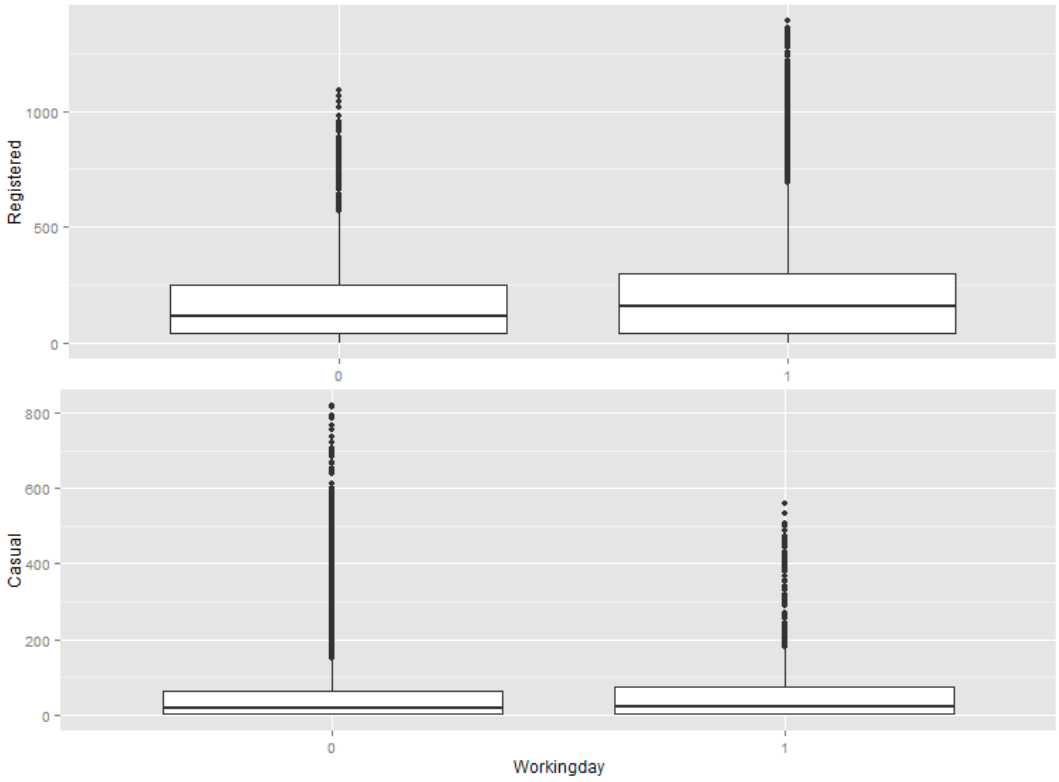

Figure 22Workingdayplots

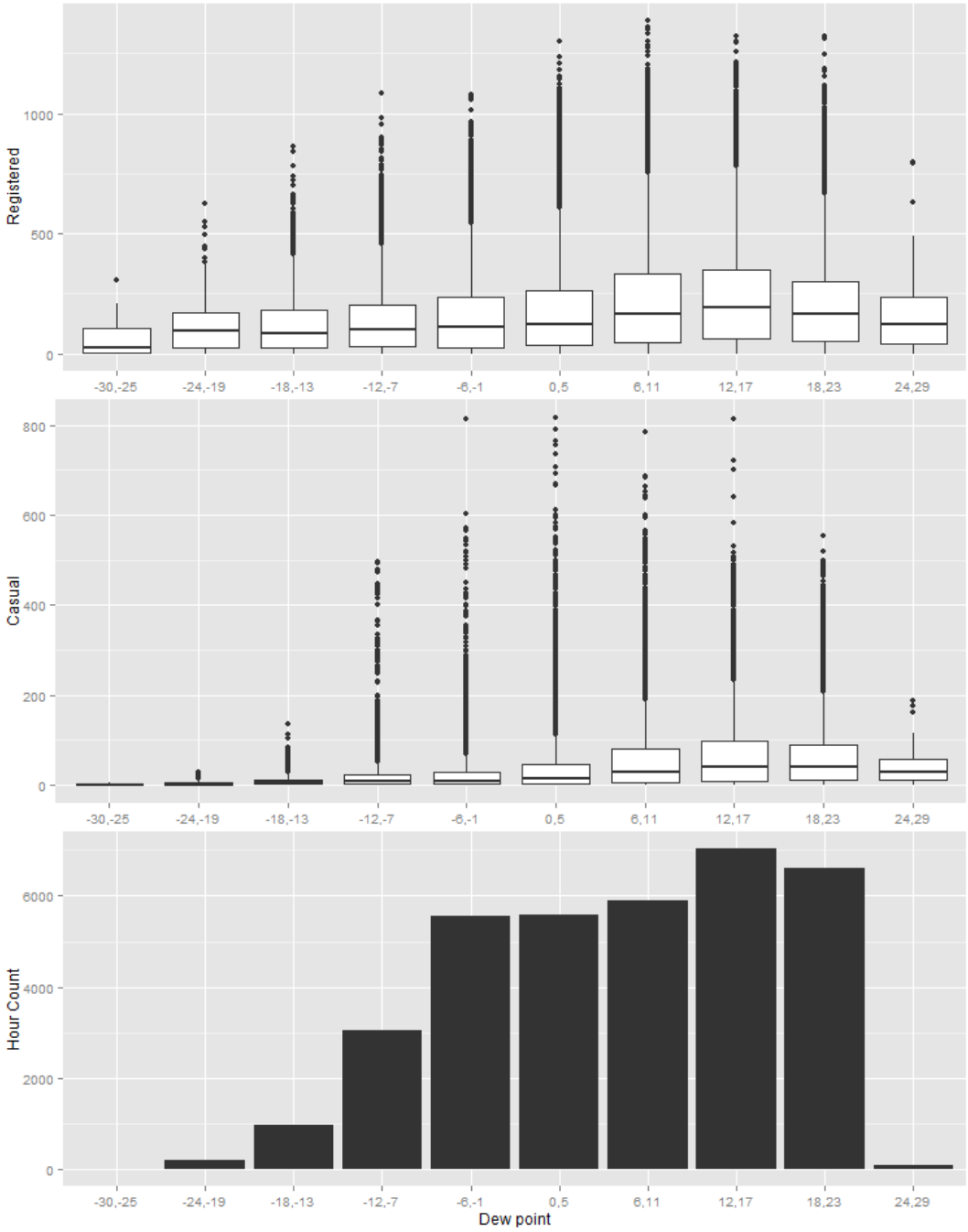

Figure 23. Dew point plots

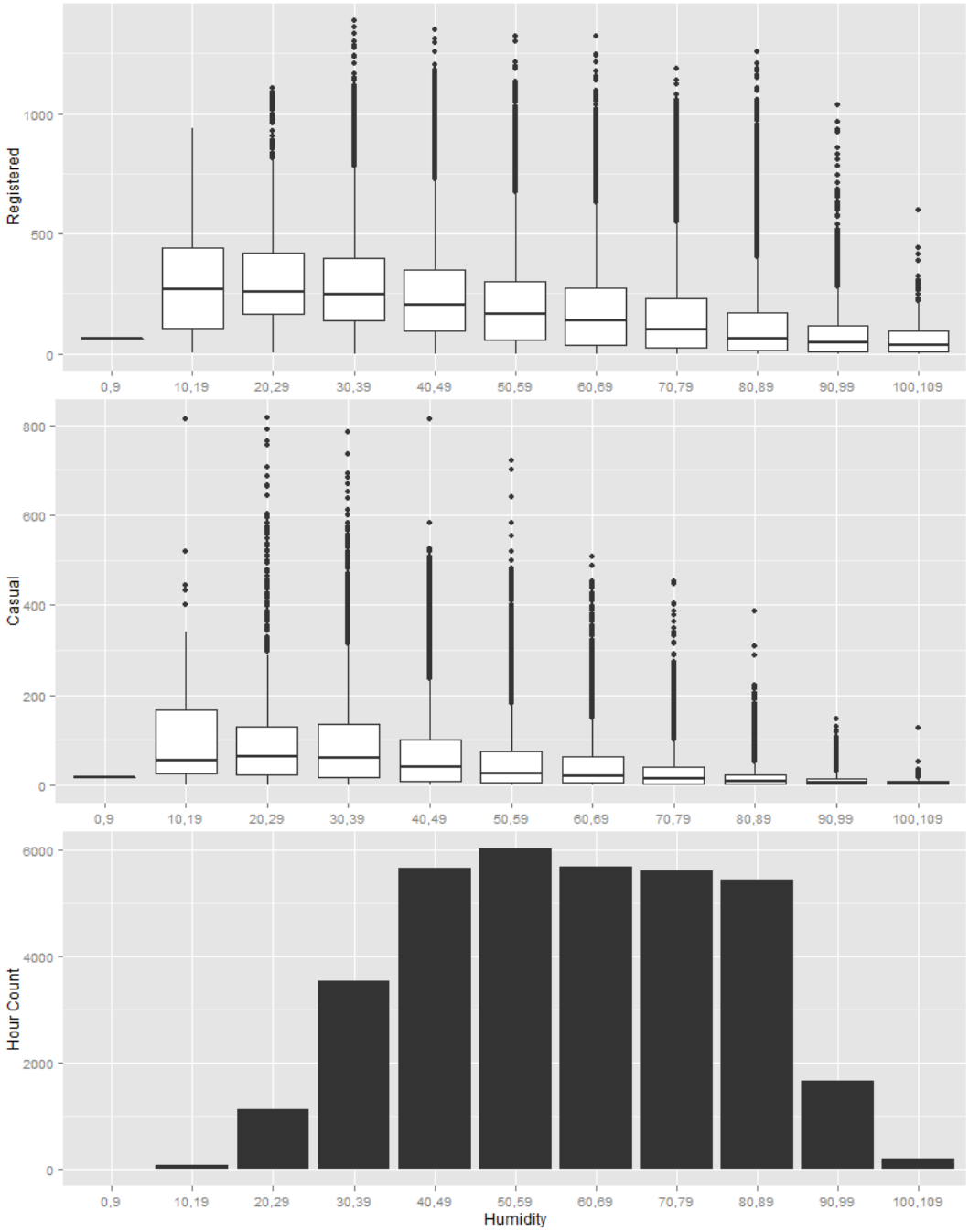

Figure 24. Humidity plots

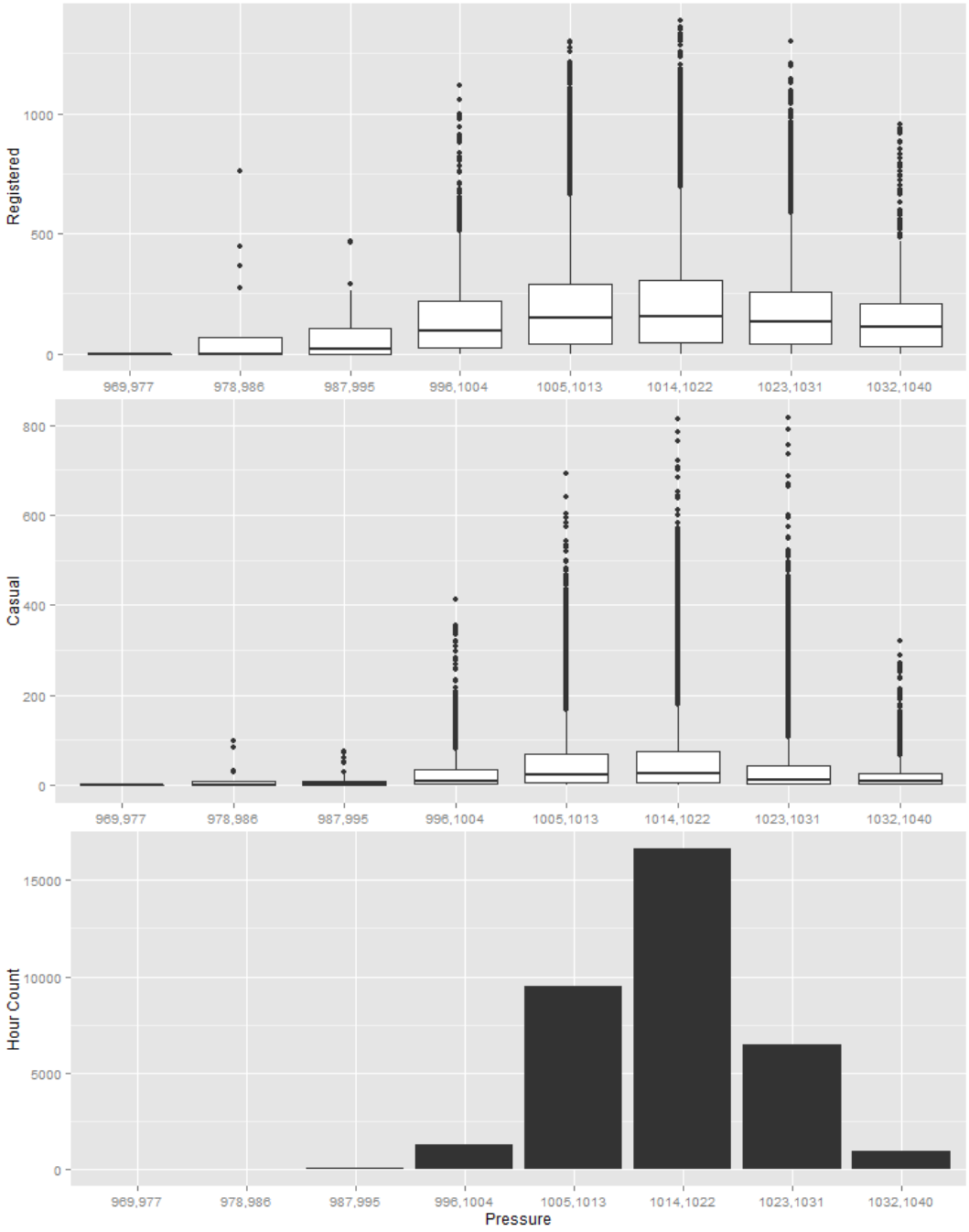

Figure 25. Pressure plots

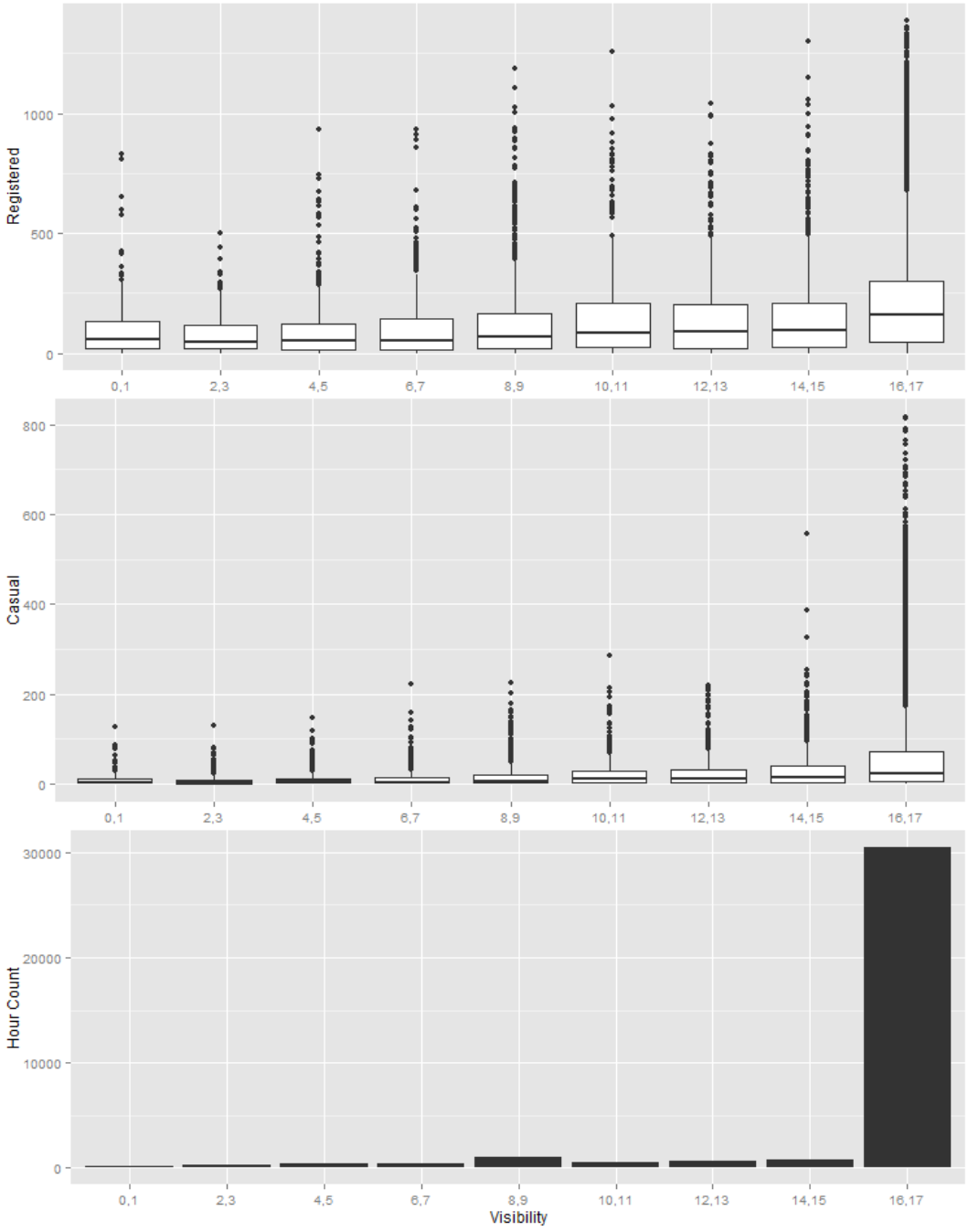

Figure 26. Visibility plots

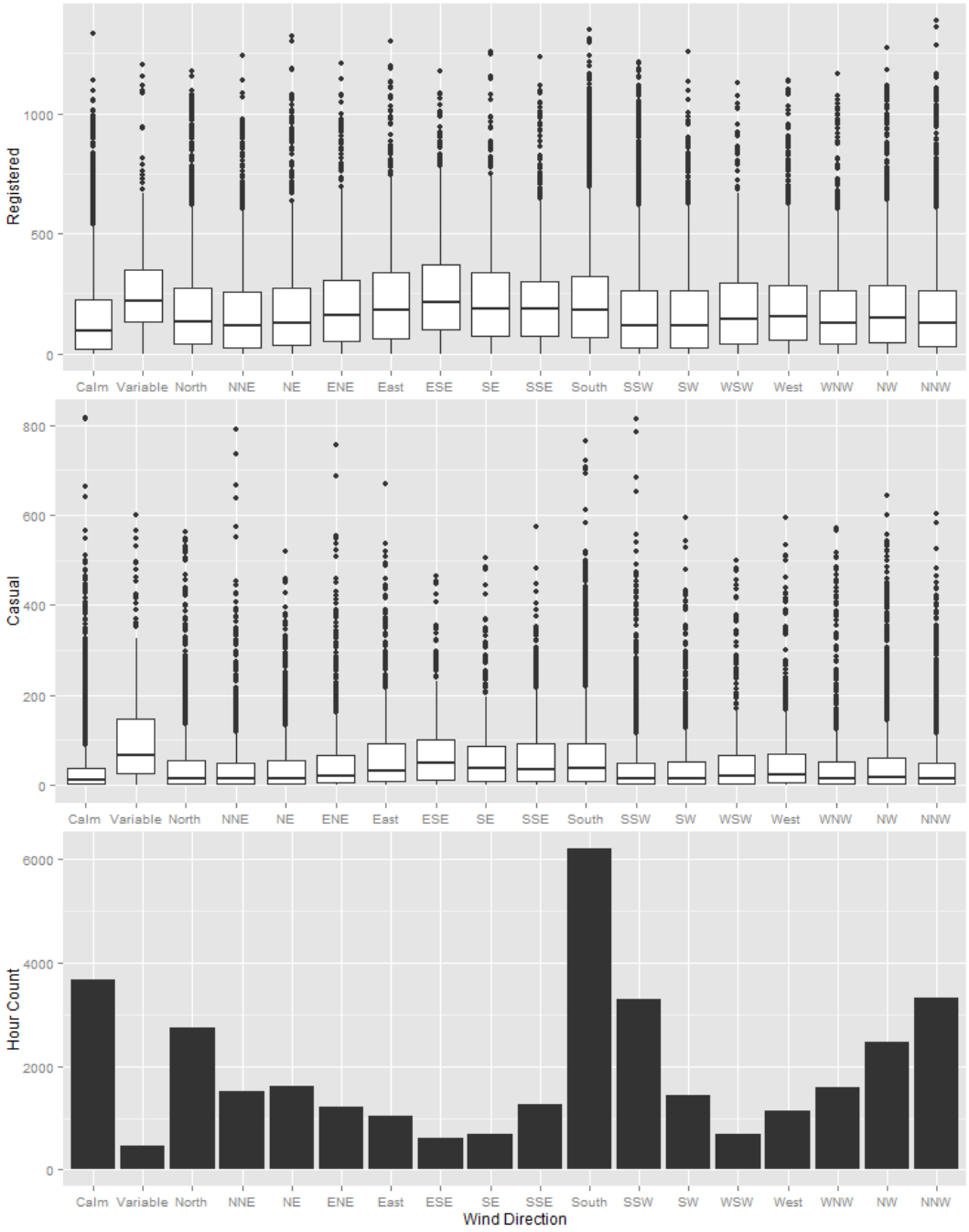

Figure 27. Wind direction plots

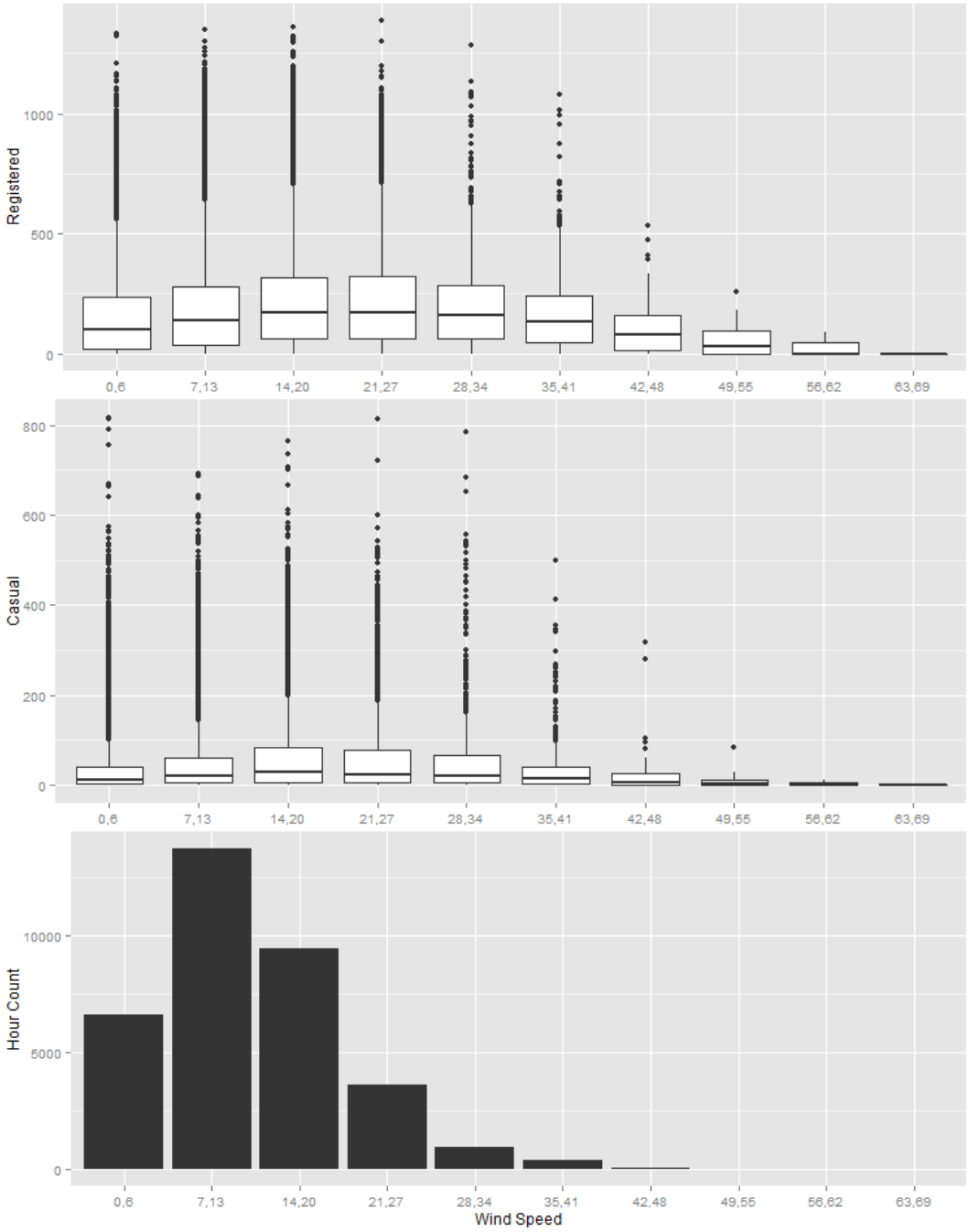

Figure 28. Wind speed plots

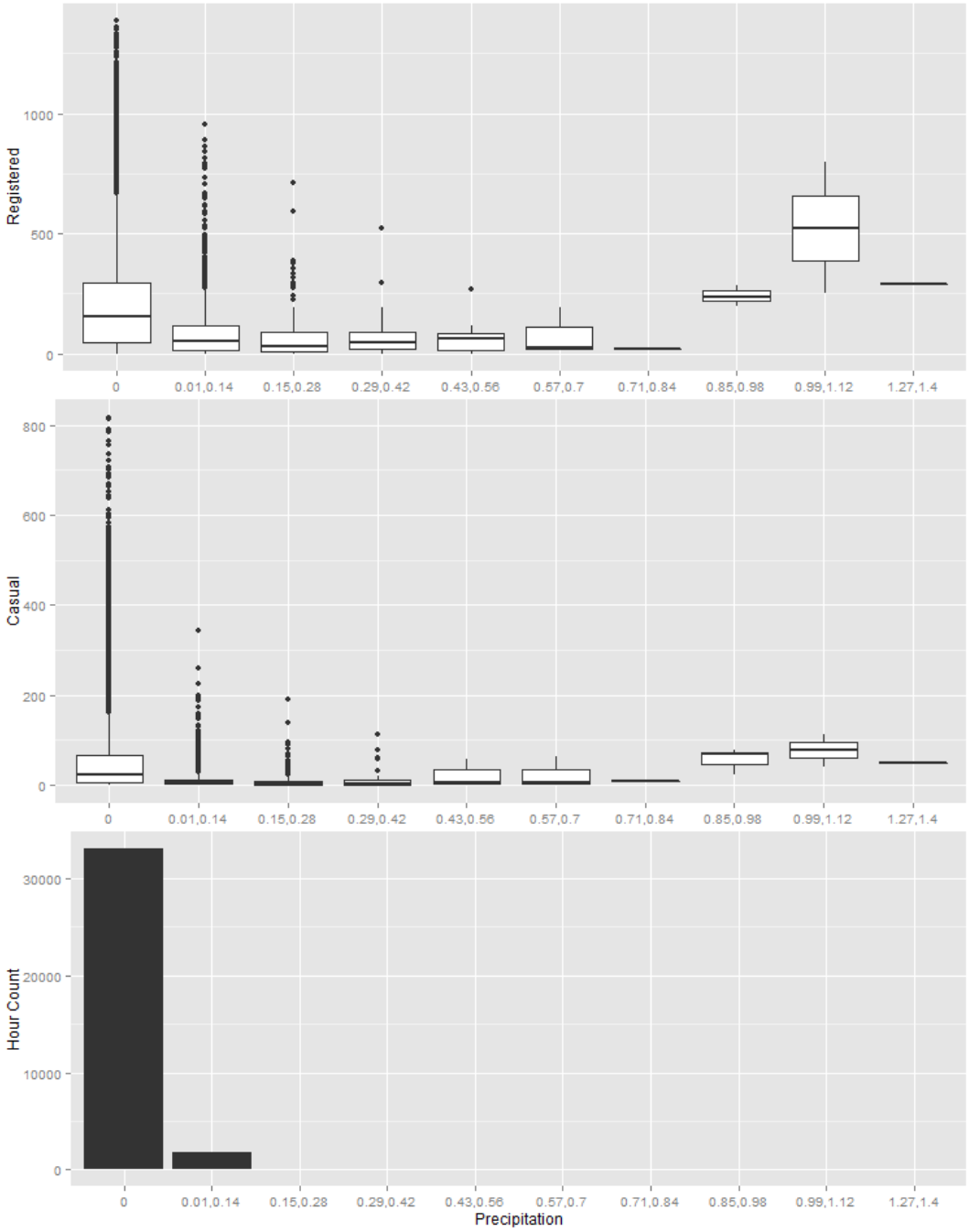

Figure 29. Precipitation plots

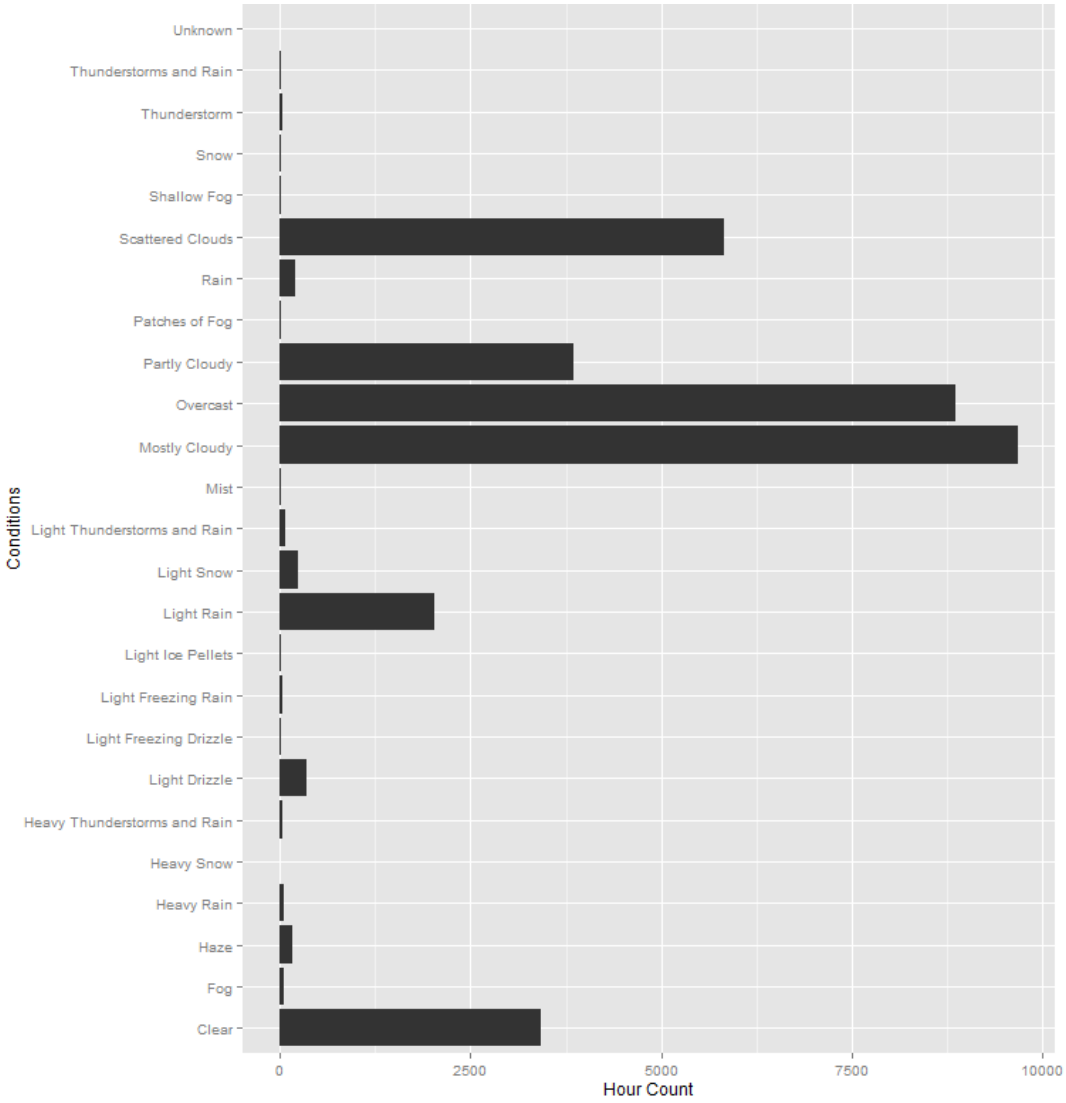

Figure 30. Condition plots 1

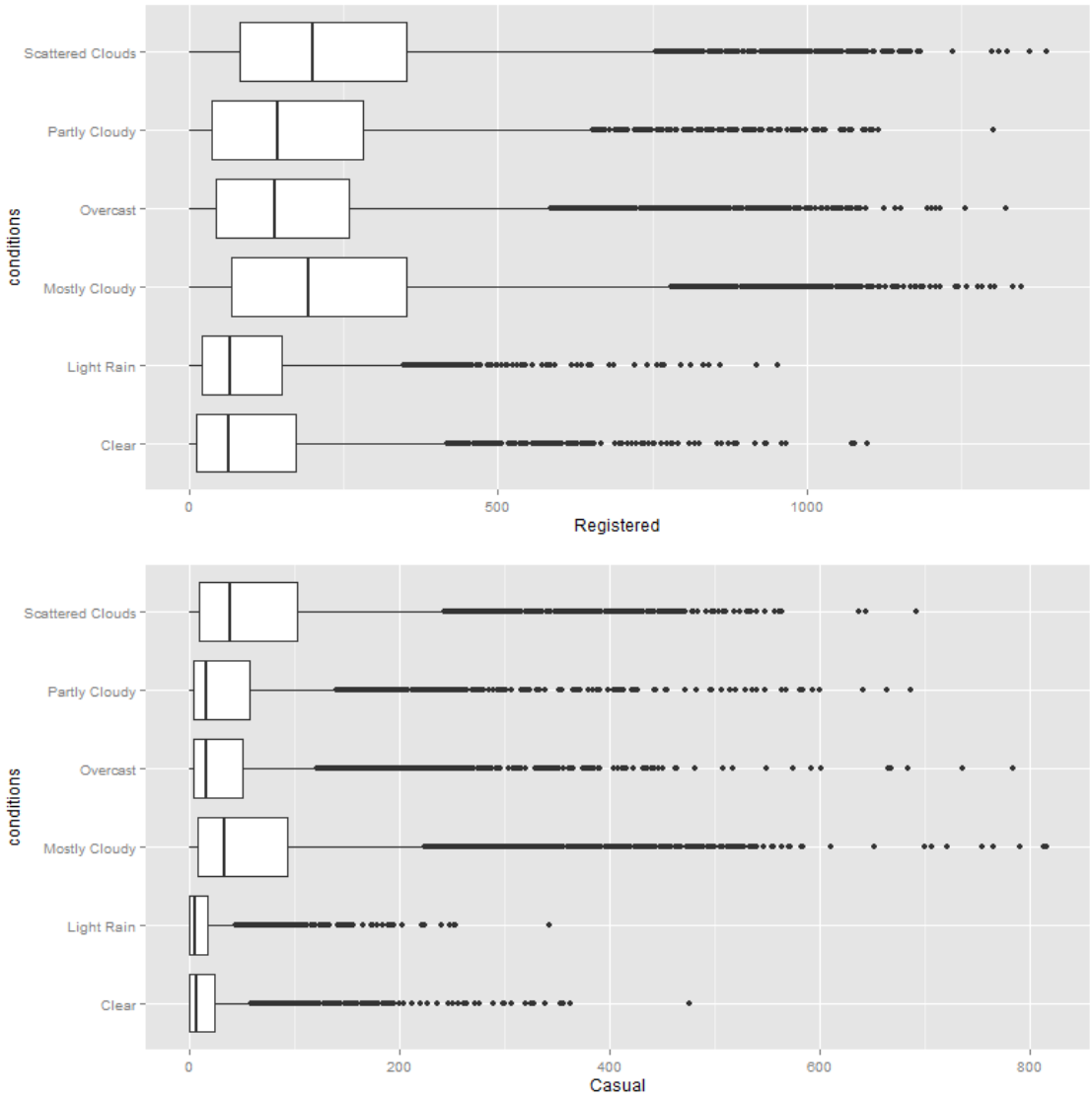

Figure 31. Condition plots 2