# Patterns and Variation in Bike Sharing Usage

Yue Dong
yuedong029@uottawa.ca

Philippe Paradis
philippe.paradis@carleton.ca

April 21, 2015

## Contents

# Acknowledgements

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude is given to our Professor Shirley Mills for the excellent course in data mining, introducing us to R, and giving us an extension to finish the project.

In this project, Yue Dong is mainly responsible for the data visualization, association rule mining, clustering and outlier detections. Philippe Paradis is mainly responsible for the PCA, data imputation, and supervised learning.

**Abstract**

The goal of our project is analyse the Bike Sharing dataset made publicly available on the UCI repository by the Capital Bikeshare company, located in Washington D.C. In particular, a significant portion of the analysis was performed to support the task of predicting the hourly demand for bike rentals, as part of a Kaggle competition using the same dataset. Through our project, we identified several important unsupervised learning techniques that helped us create more predictive features and data instances. We then trained and tuned several different models with our data, such as random forests and various types of neural networks. In the end, we found that neural networks based models perform well on this dataset; in particular, using PCA following by `avNNet` (ensemble of neural networks) performed best, which allowed us the rank 111 (out of 2696 participating teams, over more than 10 months) on the Kaggle leaderboard.

# 1 Introduction

Bicycle sharing systems have seen a surge in popularity throughout the world since the mid-2000s. For example, Bixi was launched in 2008 in Montreal and is now the largest bike sharing system in Canada. It was quickly followed by Capital Bixi in Ottawa/Gatineau in June 2009. There are now more than 700 cities worldwide with bicycle sharing systems.

Part of the reason for the sudden large-scale adoption of those systems is the advance in technology, lowering initial costs and increasing profit margins for companies. Most important of all, advances in information technology significantly improved the viability of this type of business. Modern technology allows companies to automate bike rentals, drop-offs, reservations and fleet tracking. It also makes it more convenient to the user and deters theft by requiring large credit card deposits.

Since bike sharing systems are all automated, tons of very detailed data has been collected about the usage of those systems. In particular, many organisations operating bike sharing systems are public organisations, such as cities, or private-public partnerships. As a consequence, those organisations tend to frequently make part of their data available to the public.

In this project, we used such a detailed dataset of the city bikeshare system in Washington D.C. The full dataset is available at the UCI repository [1] while part of the dataset is also available through the Kaggle competition. In our project, we focused on predicting the number of bike rentals for the bikesharing system in Washington D.C. while using the dataset from the Kaggle competition.

Although the bike sharing dataset is relatively new (2013), there are several papers published on this dataset. Some papers use this dataset for the events detection, such as Gama's paper [1] proposed to use a combination of detectors with background knowledge to improve the detection. But the majority of the papers use this dataset for regression to predict the bike rental demand: Jimmy's paper proposed to use ensemble of Conditional Inference Trees (CTree); Datta's master's thesis [5] proposed to use ensemble of decision trees and random forests; Giot's paper [4] proposed to

augment the dataset by 24 columns in order to predict the bike rental demand 24 hours in advance. Based on these papers, we decided to try ensemble of neural networks and compare the performance with tree-based algorithms.

The rest of the paper are organized as follows. In section 2, we discuss the dataset and two approaches we used to explore the data: visualization and association rule mining. In section 3, We talk about the unsupervised learning used in this project. In section 4, we discuss the supervised learning methods we used for the regression.

# 2 Dataset & Data exploration

## 2.1 Dataset

The dataset we used is the Bike Sharing Dataset from the UCI repository. According to the dataset description, "this dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information." The summary of this dataset is as Table 1 shows. More information of the dataset can be found at http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset.

| | |
|---|---|
| Number of instances: | 17389 (hourly), 731 (daily) |
| Number of Attributes: | 16 |
| Attribute Characteristics: | Categorical, Integer, Date/time, Decimal |
| **Attributes Information:** | |
| **Attribute name** | **Description** |
| instant | record index |
| dteday | date |
| season | season (1:springer, 2:summer, 3:fall, 4:winter) |
| yr | year (0: 2011, 1:2012) |
| mnth | month (1 to 12) |
| hr | hour (0 to 23) |
| holiday | whether day is holiday or not |
| weekday | day of the week |
| workingday | 1: if day is neither weekend nor holiday; 0: otherwise. |
| weathersit | 1: Clear; 2: Mist; 3: Light Snow, Light Rain; 4: Heavy Rain, Ice Pallets, Snow + Fog; |
| temp | Normalized temperature in Celsius |
| atemp | Normalized feeling temperature in Celsius |
| windspeed | Normalized wind speed |
| casual | count of casual users |
| registered | count of registered users |
| cnt | count of total rental bikes including both casual and registered |

Table 1: Summary of the bike sharing dataset

## 2.2 Data Visualization

Data visualization is a way to gain some insight about the dataset we are investigating. We first plotted an scatterplot matrix on bike sharing daily dataset to explore the correlations between the attributes.

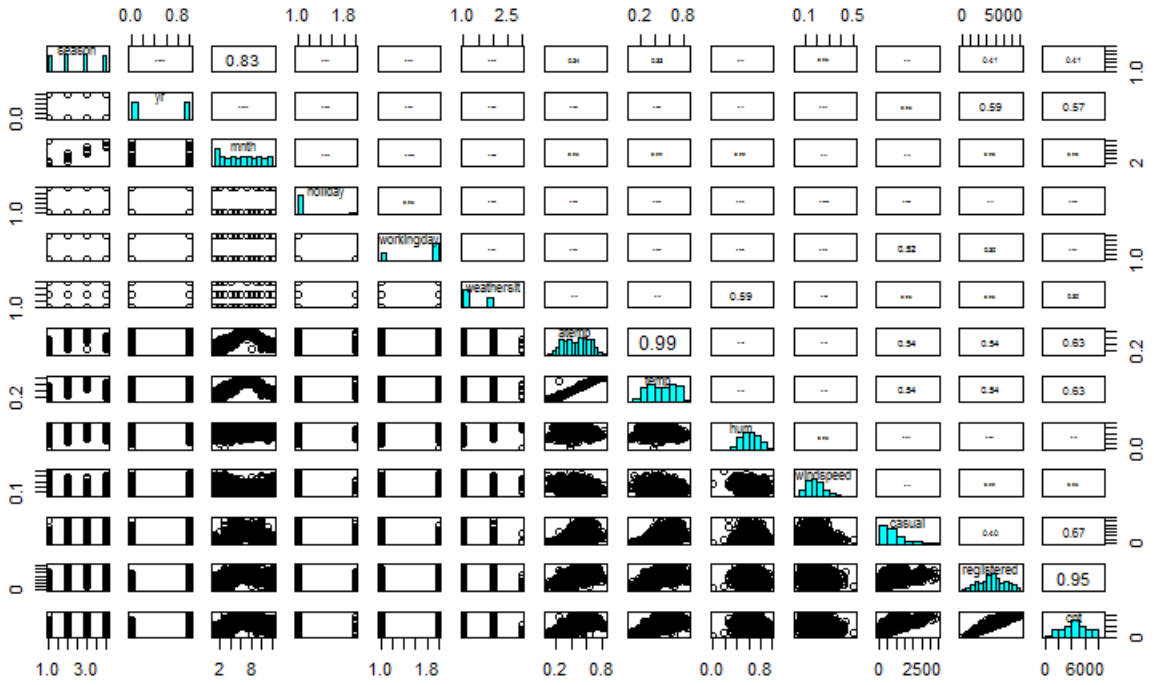



Figure 1: Scatterplot matrix of the bike sharing (daily) dataset

As we can see from Figure 1, there are 0.99 correlation between `temp` and `atemp`, 0.95 correlation between `registered` and `cnt`, and 0.83 correlation between `season` and `mnth`.

High correlation between two variables means that as one variable rises or falls, the other variable rises or falls as well. Since we don't want attributes with high correlation in our dataset, we can pick one attribute from each pair with high correlation.

On the other side, attributes with high correlations to `cnt` can be good predictors for the bike rental counts. From the last column of Figure 1, we obtained a list of attributes with high correlations to `cnt` in decreasing order:

| | registered | casual | temp/atemp | yr | season |
|---|---|---|---|---|---|
| correlation to cnt | 0.95 | 0.67 | 0.63 | 0.57 | 0.41 |

Table 2: Correlations to `cnt` in decreasing order

To further explore the correlations between variables, we plotted a heat map with hierarchical clustering.

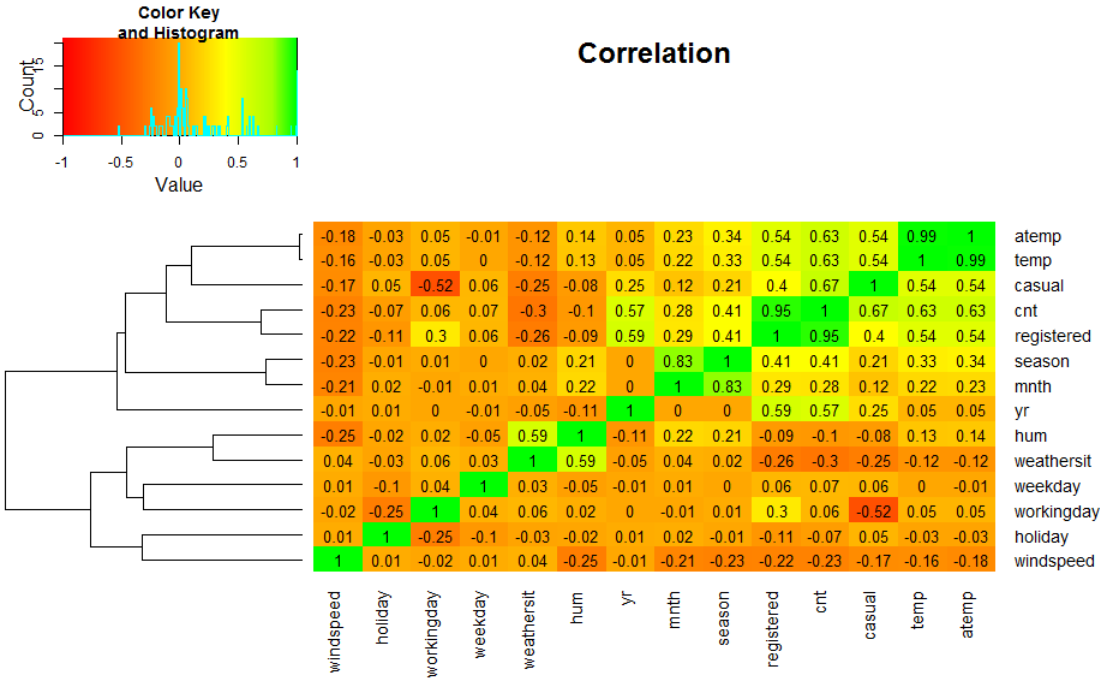


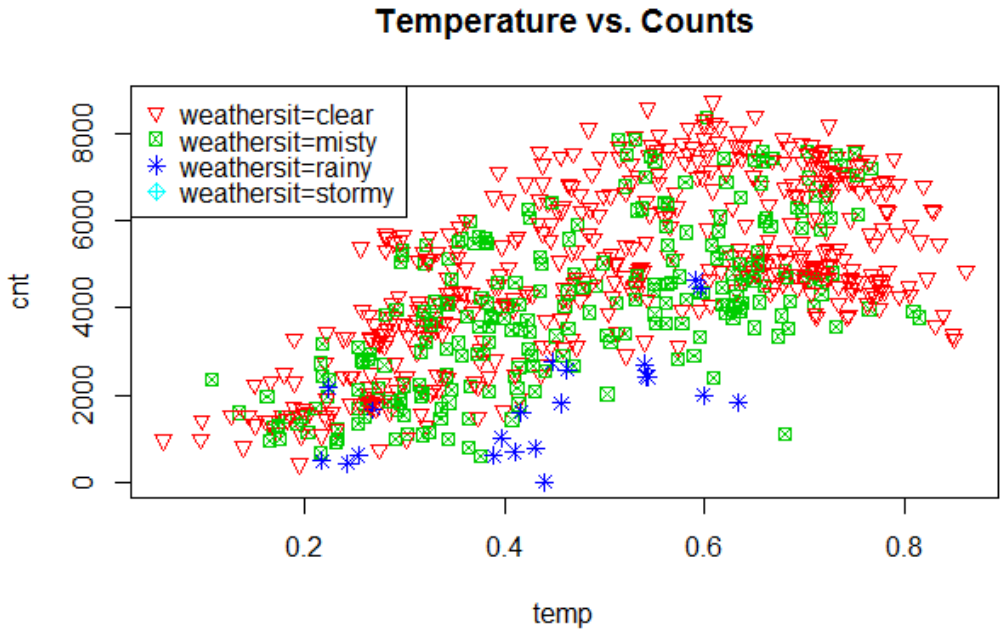

Figure 2: Heat map of the bike sharing (daily) dataset

From Figure 2, we found that `workingday` has a negative correlation with `casual` (-0.52) which corresponds to the intuition that more casual users go out biking when it is not workingday.

Moreover, the hierarchical clustering grouped `atemp` and `temp`, `cnt` and `registered`, `season` and `mnth`, `hum` and `weathersit`, `weekday` and `workingday` together in the lowest level. If we want two clusters from the hierarchical clustering, we will get `atemp, temp, casual, cnt, registered, season, mnth, yr` as one cluster; and `hum, weathersit, weekday, workingday, holiday, windspeed` as the other cluster.

These correlations give us an insight about the dataset. Based on the above observations, we decided to further investigate on the following relationships: 1) `temp/weathersit` to `cnt`, 2) `hr/workingday` (weekday and non-holiday) to `cnt`, 3)`season/mnth` to `cnt`, 4) `registered` to `casual`.

**Explore 1. temp/weathersit to cnt** Most likely, temperature and weather influence the bike demand. We thus plotted `temp` against daily `cnt` with weather information:




The rental bike counts increases when the temperature increases. With the same temperature, the better the weather is, the more bike rental counts are.

To compare the effects of feeling temperature and temperature on bike rental demand, we plotted `atemp` against the average `cnt` and `temp` against the average `cnt`:

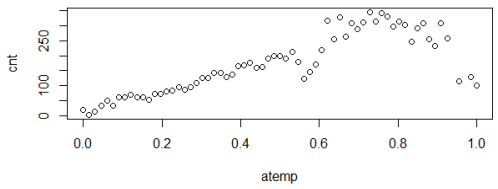



Figure 3: Atemp against average cnt

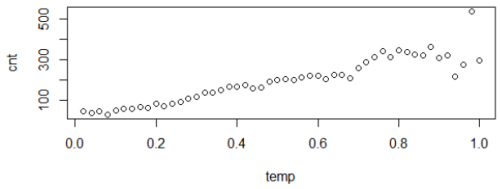



Figure 4: Temp against average cnt

We can confirm the high linear correlation between `temp` and `cnt` from Figure 4 since the plot is most linear. On the other hand, we can see from Figure 3 that the bike rental demand is the highest at `atemp`= 0.7, it start to decrease when `atemp` exceeds 0.7.

**Explore 2. hr/workingday (weekday and non-holiday) to cnt** The hourly bike rental demand probably shows a different pattern on weekdays and weekends. Many people rent bikes to work on weekdays, and other people rent bikes for leisure on weekends. To investigate if this affects the bike rental demand, we first plotted the counts on a typical week:

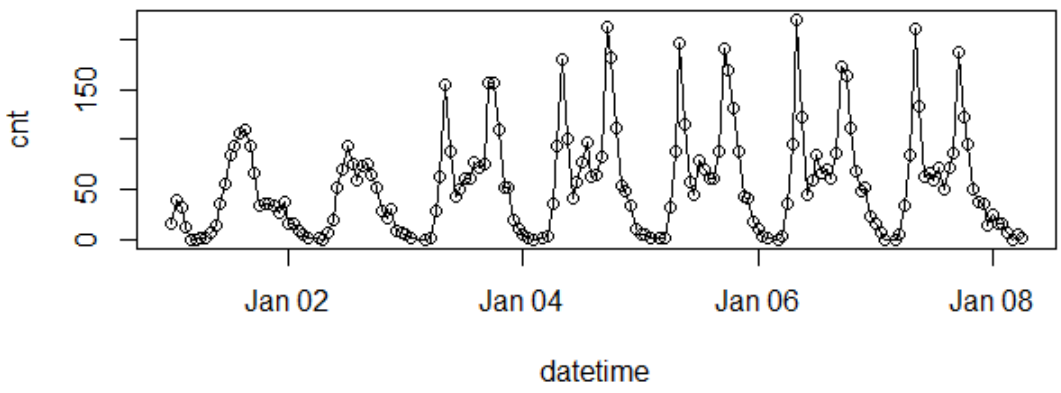
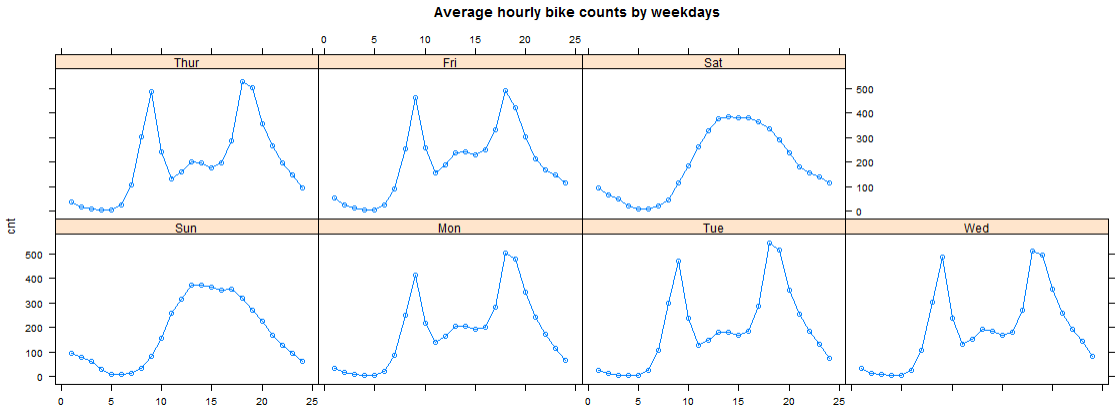



Figure 5: A typical week of bike rental counts (Sat.- Fri.)

We can see that the bike demand is different on weekdays and weekends in Figure 5. To check if it is true generally, we plotted the average hourly counts conditioned on day of the week from the whole dataset.




In general, there are two peaks of bike rental during weekdays: around 8-9am and around 5-6pm when people go to work and get off work. On the other hand, the peak bike rentals happen from 12pm to 4pm on weekends when people go outdoors for leisure.

A natural question to ask here is that what's the bike rental demand is on holidays, especially holidays on weekdays? Is it going to be similar as the pattern of a weekday or that of weekends? To answer this question, we plotted the average hourly counts conditioned on day of the week AND if it's a holiday.

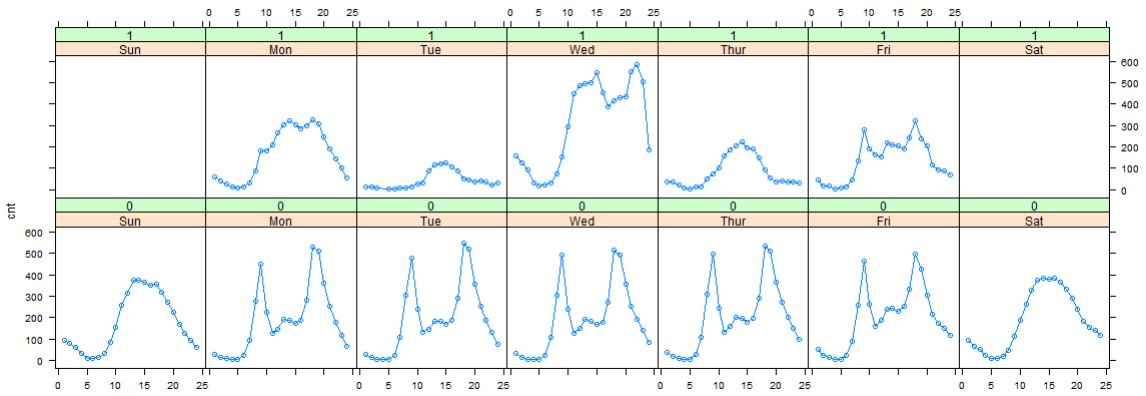


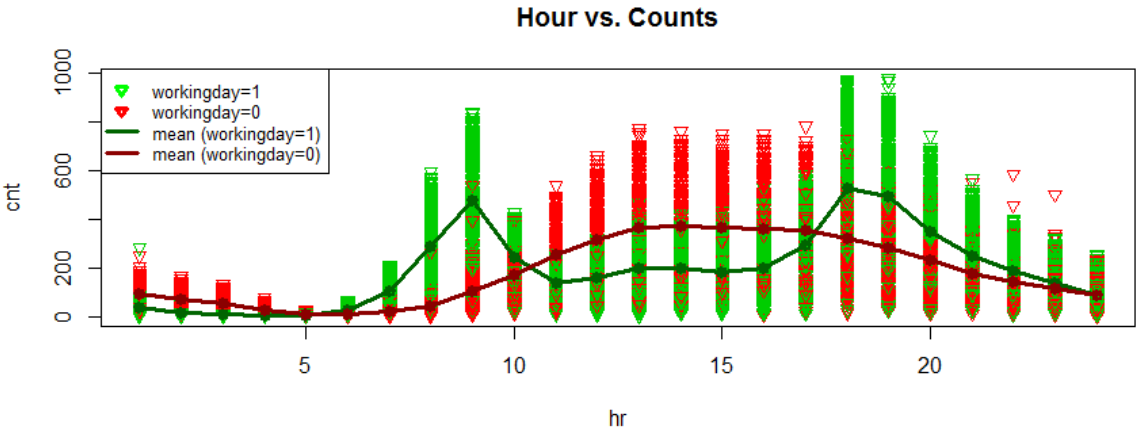

Figure 6: Average hourly bike rental on different weekdays and holidays (1 indicates holiday, 0 indicates non-holiday)

Here are the observations from Figure 6:

1. There is no holiday on weekends from the dataset since the plot of bike rental on Sunday and Saturday with holiday=1 are empty.

2. Whenever there is a holiday on weekdays, the bike rental shows very different pattern than that of usual weekdays. It can be similar to the weekends' bike rental pattern with lower counts (Monday, Tuesday and Thursday in Figure 6) or a combination of weekdays' pattern and weekends' pattern (Monday and Friday in Figure 6).

Therefore, workingday (weekday and non-holiday) is a good predictor for the hourly bike counts as the following graph shows:




The bike rentals are concentrating from 7am to 9pm on all days. The peak rental occurs at 8-9am and 5-6pm on workingdays and occurs from 10am to 4pm on non-workingdays.

**Explore 3. mnth/season to cnt** Let's see if bike rental demand depending on the `mnth`/ `season`. To see the relationship of month and bike rental counts, we plotted a boxplot:

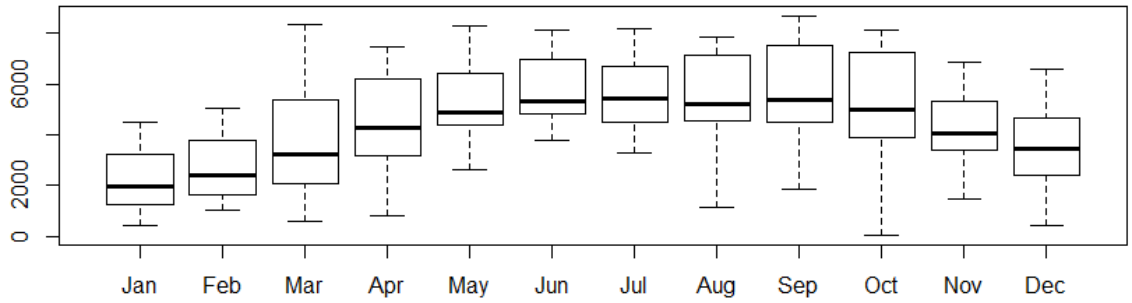



Figure 7: Boxplot of counts on different months

The bike rental demand is the lowest in January and is the highest from May to October. To see if the bike rental patterns are similar among months, we plotted a coplot of hourly bike counts conditioned on month.

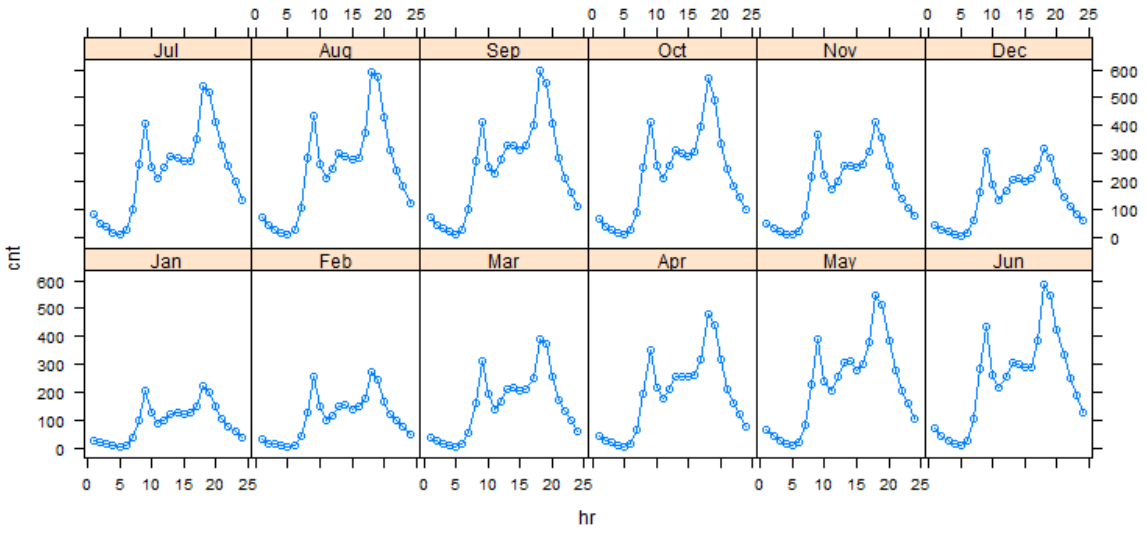



Figure 8: Average hourly bike rental on different month

The bike rentals are the highest in the month from May to October, and the patterns of hourly rental are similar in these months with the afternoon peak higher than the morning peak. The difference between the afternoon peak and the morning peak might due to the casual users going out biking in the afternoon during the summer. From November to February, the bike rentals are low and the morning peak has similar counts as that of the afternoon peak.

Next, we investigated the relationship of `season` with bike rental counts. We first plotted a coplot of hourly counts conditioned on `season`:

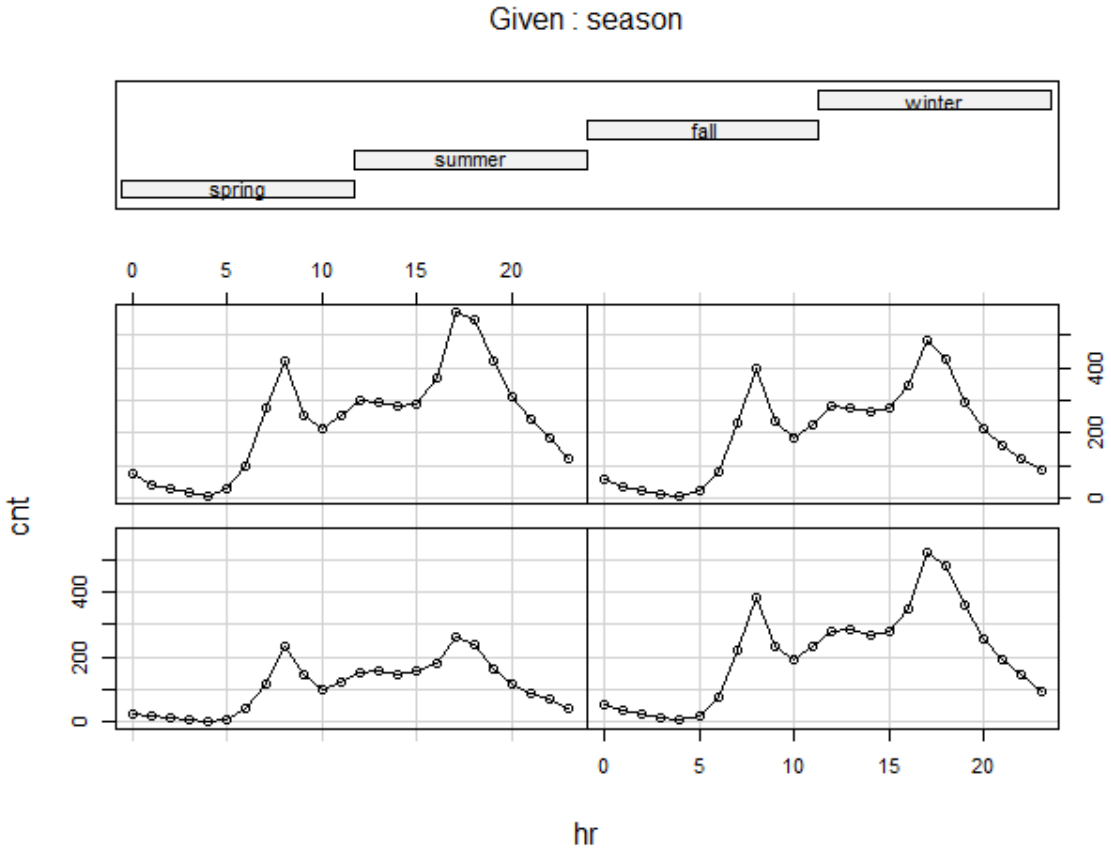



Figure 9: Coplot of hourly counts conditional on season

The bike demand is the highest in the fall (July to September) and is the lowest in the spring(January to March). The bike rental patterns are similar in the summer, the fall, and the winter with afternoon peak higher than morning peak.

We then used parallel coordinate plots in Ggobi to compare all attributes in different seasons:

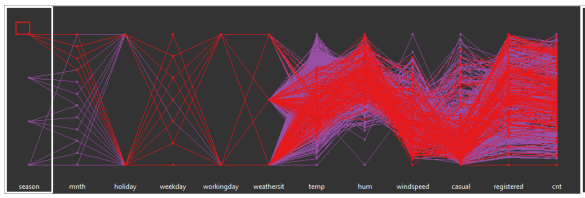



Figure 10: Spring

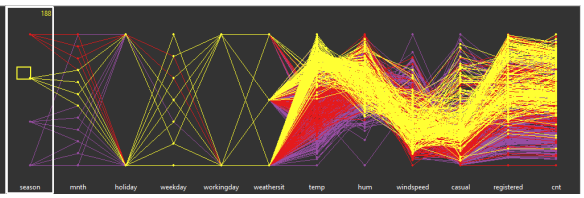



Figure 11: Summer

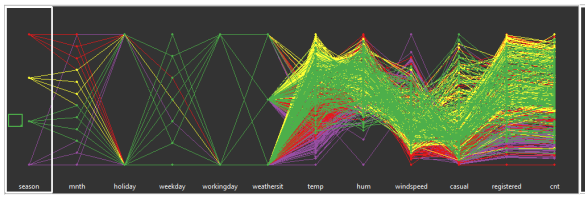



Figure 12: Fall

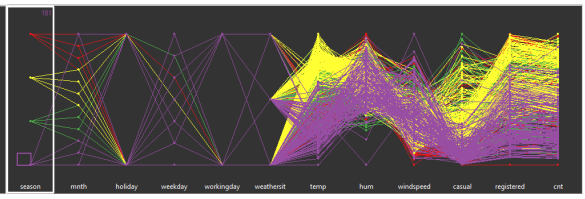



Figure 13: Winter

From the above plots, we noticed that casual users' bike rental demand are different than that of registered users in different seasons. The demand pattern is summarize in the following table:

| Season | Spring | Summer | Fall | Winter |
|---|---|---|---|---|
| Registered | median, high | median, high | median, high | low, median |
| Casual | low | low, median, high | low, median | low |

Table 3: Bike rental demand of casual/registered users in different season

**Explore 4. workingday to registered/casual** We found in Explore 3 that casual users behave differently than registered users in different seasons. To see if workingdays contribute to the difference, we plotted a coplot of hourly counts conditioned on `season` with colored workingdays.

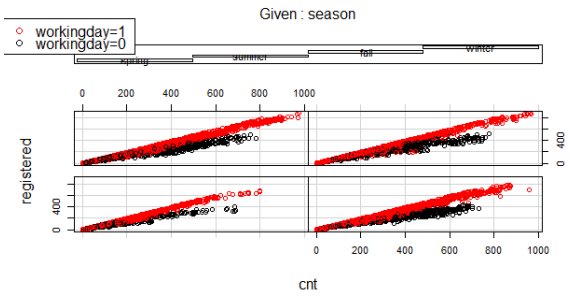



Figure 14: Registered-cnt|season

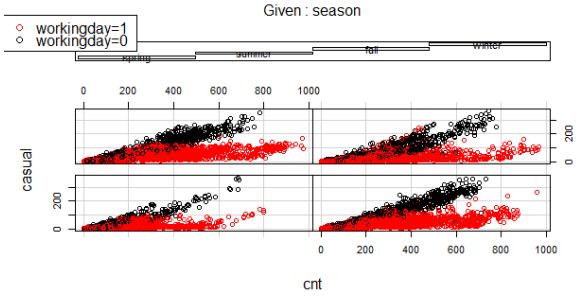



Figure 15: Casual-cnt|season

From Figure 14 and Figure 15, we can see that registered users and casual users behave differently on workingdays and non-workingdays. More registered users rent bike on workingdays, and more casual users rent bike on non-working days. A coplot on `registered` against `casual` conditioned on the month can explain this difference even better:

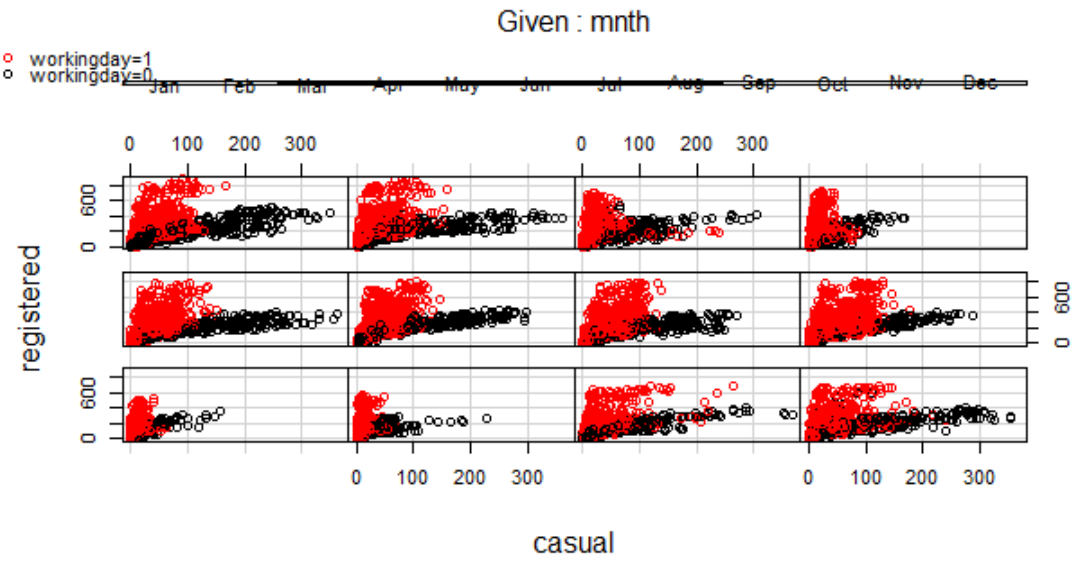



Figure 16: Coplot of registered-casual| mnth colored on workingday

In Figure 16, `registered` and `casual` almost have a linear correlation on the non-workingday (black dots). But there is no strong linear correlation between the two on workingdays(red dots). This suggests us that building separated models for `registered` and `casual` on `workingday` and non-`workingday` can improve the prediction of the total counts .

To further explore the difference between casual and registered users, we plotted hourly bike rental counts for casual users and for registered users in a daily basis for the whole dataset. Here are some interesting findings:

1. In winter, less casual users go out biking on weekdays. But the amount of casual users renting bikes on weekends doesn't reduce much from the summer to the winter.

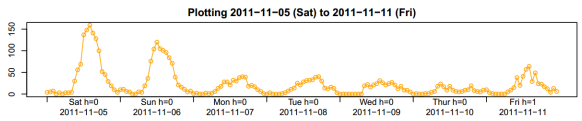



Figure 17: Winter - casual users

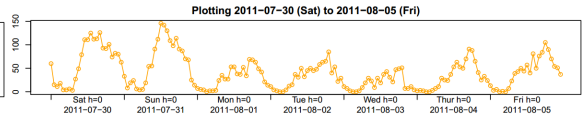



Figure 18: Summer - casual users

2. Compare to the usual summer weeks, casual users' bike rentals are very low in the week when school starts. This indicates that a large percentage of casual users are students.

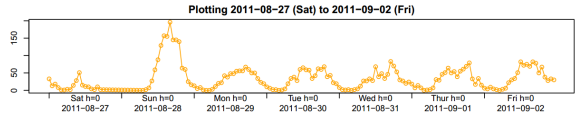



Figure 19: The week before school starts - casual users

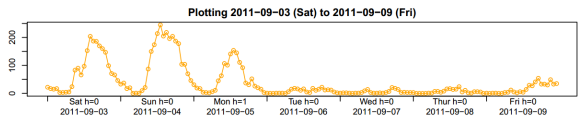



Figure 20: The week when school starts - casual users

3. Christmas is an atypical week for casual users' bike rentals.

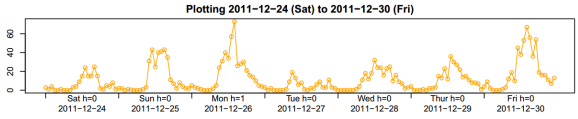



Figure 21: Christmas 2011- casual users

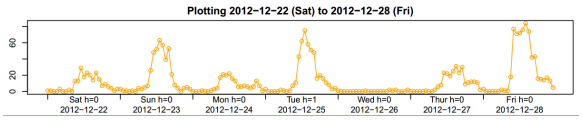



Figure 22: Christmas 2012- casual users

4. The bike rentals for registered users don't change that much from the summer to the winter on weekdays. However, more registered users go bike on weekends during the summer.

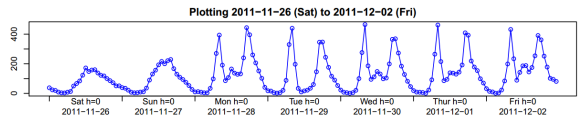



Figure 23: registered typical winter

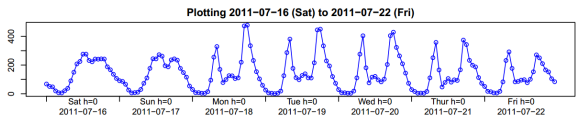



Figure 24: registered typical summer

5. Christmas is an atypical week for registered users' bike rentals.

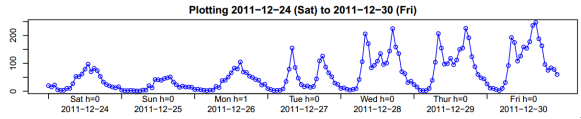



Figure 25: registered Chritmas 2011

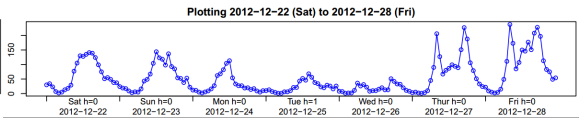



Figure 26: registered Christmas 2012

**Preliminary Conclusion.** In short we have identified strong correlation between counts and these predictors: 1. Temperature 2. Hour of the Day 3. Working Day 4. Month of the Year.

## 2.3 Association Rule Mining

To investigate more about the dataset, we did an association rule mining. The bike sharing dataset contains a mixture of categorical and numeric attributes and therefore need some preparation before using apriori algorithm on it. Here are the steps for our experiment in association rule mining:

1. We fist removed the unrelated features: `instant`, `dteday`.
2. We then mapped the seven remaining continuous attributes `(temp, atemp, hum, windspeed, registered, casual, and cnt)` to ordinal attributes by building suitable categories.
3. Next, we coerced the data set to transactions (a binary incidence matrix).
4. Finally, we ran Apriori algorithm with support = 0.01, confidence = 0.3.

After the rule mining, we sorted the rules by lift with "`cnt`=high" in the right hand side. Here are the top 5 rules:

```
lhs                    rhs           support confidence     lift
1  {weekday=Sat,
casual=high}      => {cnt=high} 0.03283174          1 3.768041
2  {casual=high,
registered=high}  => {cnt=high} 0.02872777          1 3.768041
3  {hum=medium,
casual=high}      => {cnt=high} 0.03283174          1 3.768041
4  {mnth=Sep,
registered=high}  => {cnt=high} 0.03419973          1 3.768041
5  {mnth=Jun,
registered=high}  => {cnt=high} 0.03009576          1 3.768041
```

The first rule tells us that when it is Saturday and bike rentals from casual users are high, there will be a large chance that total bike rentals are high.

We also sorted the rules by lift with "`registered`=high" on the right hand side. The following are the top 5 rules:

```
lhs                rhs                  support confidence     lift
1  {weekday=Fri,
cnt=high}      => {registered=high} 0.03967168          1 3.673367
2  {weekday=Wed,
cnt=high}      => {registered=high} 0.03693570          1 3.673367
3  {weekday=Thur,
casual=medium} => {registered=high} 0.01231190          1 3.673367
4  {weekday=Thur,
cnt=high}      => {registered=high} 0.04377565          1 3.673367
5  {casual=low,
cnt=high}      => {registered=high} 0.12859097          1 3.673367
```

in general, these five rules tell us that when it is weekdays, `registered` is high.

Finally, we obtained the following top 5 rules by sorting with lift and with "`casual`=high" on the right hand side:

```
lhs                    rhs              support confidence     lift
1  {season=summer,
weekday=Sat,
cnt=high}         => {casual=high} 0.01504788          1 16.61364
2  {season=summer,
weekday=Sat,
workingday=0,
cnt=high}         => {casual=high} 0.01504788          1 16.61364
3  {season=summer,
weekday=Sat,
atemp=medium,
```

```
cnt=high}          => {casual=high} 0.01094391          1 16.61364
4  {season=summer,
yr=2012,
weekday=Sat,
cnt=high}          => {casual=high} 0.01504788          1 16.61364
5  {season=summer,
weekday=Sat,
weathersit=clear,
cnt=high}          => {casual=high} 0.01231190          1 16.61364
```

Here, we can see that when it is summer weekends and `cnt` is high, there will be a very large chance that casual users' bike rentals are high.

Rule 4 attracted our attention since it indicates that "`yr`=2012" can help to predict high bike rental demand from casual users. We therefore plotted hourly `casual`, `registered`, and `cnt` conditioned on `yr` to compare the difference.

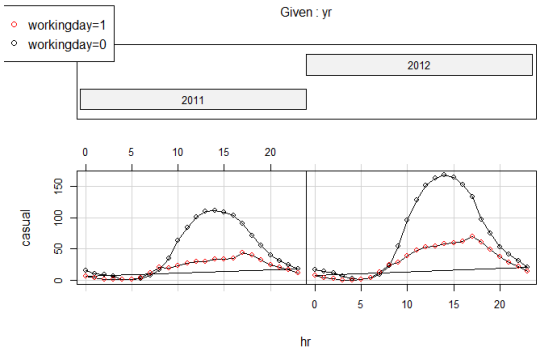



Figure 27: Coplot of casual-hr| yr

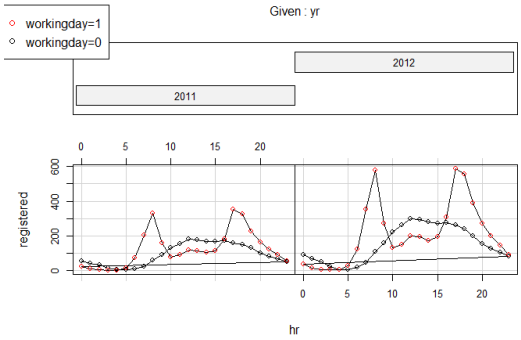



Figure 28: Coplot of registered-hr| yr

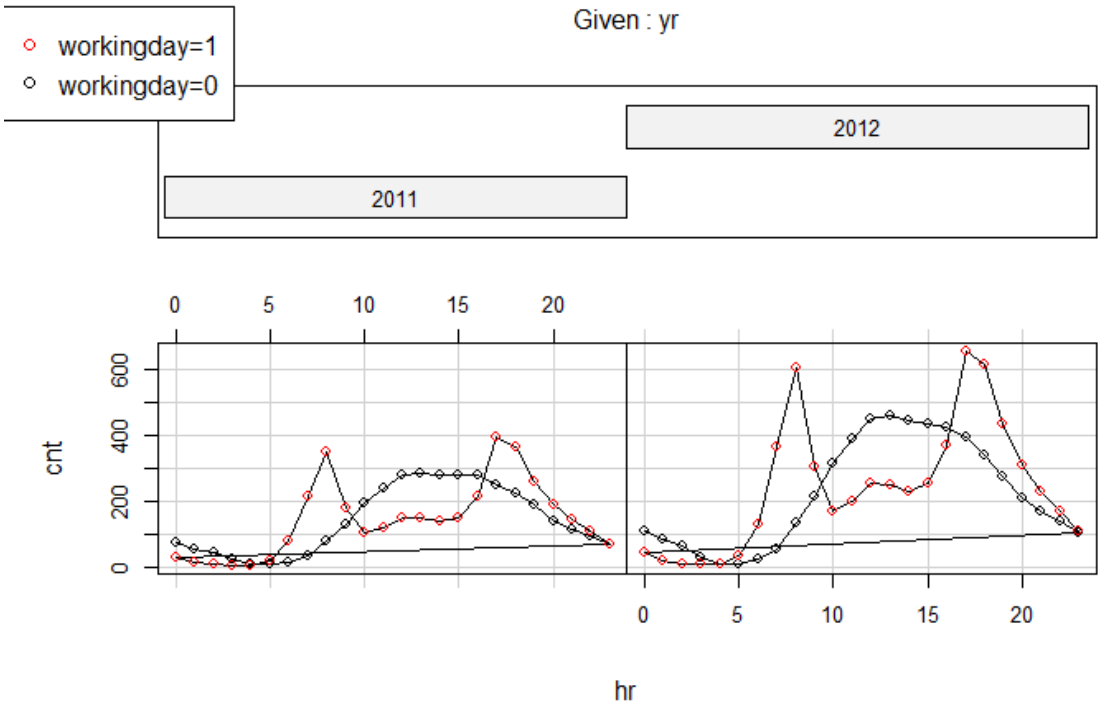



Figure 29: Coplot of cnt-hr| yr

From the above figures, the bike rental demands from the registered users and the casual users both increase from 2011 to 2012 without changing on the underlying pattern of the rental. Therefore, `yr` is also a strong predictor for the total counts.

In summary, the association rule mining didn't give us too much more information about the dataset. But it confirmed that casual and registered users behave differently. In addition, it helped us to realize year is also a strong predictor for the bike rental demand.

# 3 Unsupervised Learning

## 3.1 Dimensionality Reduction: Features Selection

The first step in performing dimensionality reduction is simply to apply common sense, i.e. features selection. Are all the features we have really necessary to begin with?

We can drop the `instant` column, which is merely numbering the data. This is redundant because R already numbers the column for us.

## 3.2 Dimensionality Reduction: PCA

The bike sharing dataset has a fairly high dimension due to the large number of categorical variables. For example, the categorical variable `season` has 4 possible

| Total variance explained | Number of top principal components required |
|---|---|
| 50% | 19 |
| 60% | 24 |
| 70% | 29 |
| 80% | 33 |
| 90% | 38 |
| 95% | 41 |

values, `mnth` has 12, `hr` has 24, `weekday` has 7 and `weathersit` has 4.

Before calling `prcomp`, it is necessary to convert our dataset to numeric data only. This involves getting rid of all categorical variables (i.e. with class of type `factor`). In order to do so, we introduce *dummy variables* for each levels of each categorical variable. For example, the variable `season` will be replaced by four variables `season.winter`, `season.spring`, `season.summer` and `season.fall`. Each dummy variable takes the value 0 or 1, depending on the value of the original categorical variable.

Note that we can always discard one of the dummy variables. Indeed, in general if there are $n$ possible options (and assuming that one and only one option must always be chosen), then the correct option can always be deduced from the knowledge of the $n-1$ options.

So, we used the R function `model.matrix` to do all this for us (including only keeping $n-1$ dummy variables for a categorical variable with $n$ levels). Note that we did not keep all features of the dataset — we only kept the features that would be relevant for doing regression later on, since this is the main reason we are interested in PCA. We kept the following features:

```
m.predictors <- data.frame(model.matrix(
~ season + yr + mnth + hr + weekday + workingday +
weathersit + atempdiff + humdiff + windspeed,
data = bike.hfx
))
```

This produced a dataset with 52 features. At this point, we performed PCA with `prcomp` and we computed how many principal components would be necessary to keep various percentages of the total variance in our dataset (see Table 3.2).

One of the benefits of using PCA would be the reduced computation time for further analysis, assuming that we would drop a certain amount of the principal components.

However, we were not sure if reducing the number of principal components would actually help produce better results in supervised learning experiments. To our surprise, it did — see Section 4.3.4 for details.

## 3.3 Data Reduction: Clustering Kmeans + Davies-Bouldi Index

The bike sharing dataset has 17365 data instances for the hourly bike sharing dataset. Instead of dimension reduction to reduce the number of attributes, we can use data reduction to reduce the number of cases. In this section, we consider using clustering methods to group similar cases together. By doing so, we can have a reduced representation for the dataset which produces the same or similar analytical results.

### 3.3.1 Data transformation

The $k$-means clustering uses Euclidean distance to calculate the similarity between instances, therefore the attributes of the input should be continuous numerical values. The bike sharing dataset has a mixture of categorical and numerical values, we thus need some data transformation before running the k-means algorithm.

Instead of using all information to clustering the data, we considered only cluster the daily bike rental patterns. In other words, we fed the hourly counts to the k-means algorithm to group the days with similar counts pattern. To achieve this, we transformed the `cnt` in hourly dataset into a $713 \times 24$ matrix as follows:

```
head(bike.24hourscnt)
    hr0 hr1 hr2 hr3 hr4 hr5 hr6 hr7 hr8 hr9 hr10 hr11 hr12 hr13 hr14 hr15
1  16  40  32  13   1   1   2   3   8  14   36   56   84   94  106  110
2  17  17   9   6   3  NA   2   1   8  20   53   70   93   75   59   74
3   5   2  NA  NA   1   3  30  64 154  88   44   51   61   61   77   72
4   5   2   1  NA   2   4  36  94 179 100   42   57   78   97   63   65
5   6   6   2  NA   2   3  33  88 195 115   57   46   79   71   62   62
6  11   4   2  NA   1   4  36  95 219 122   45   59   84   67   70   62
  hr16 hr17 hr18 hr19 hr20 hr21 hr22 hr23
1   93   67   35   37   36   34   28   39
2   76   65   53   30   22   31    9    8
3   76  157  157  110   52   52   20   12
4   83  212  182  112   54   48   35   11
5   89  190  169  132   89   43   42   19
6   86  172  163  112   69   48   52   23
```

Our goal is using k-means and DBI to find a proper way to cluster the dataset. For example, we want to find some clusters as the following graph shows:

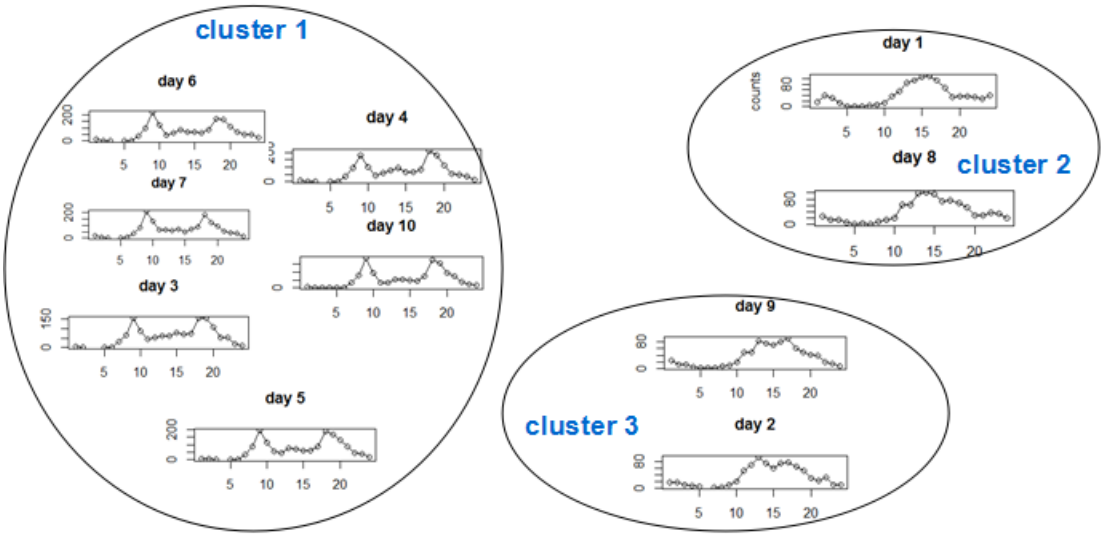



Figure 30: One example of our goal for k-means

### 3.3.2 Fill in missing values

As we can see from the last section, the bike sharing dataset has lots of missing values. For example, the bike counts on the second day at hour 5 is missing. A check shows that there are 76 days with missing values, and 14 days with more than 1 missing values.

We can fill up the missing values by three ways with the "zoo" package:

1. fill the missing value by the last observation carried forward (locf)
2. fill the missing value by linear interpolation (linear)
3. fill the missing value by cubic spline (spline)

To compare the performances of these three methods, we removed the data of hour (3,5,6,10,11,15,16) on day 1 and day 10. Then we filled the missing values with the three filling methods. Finally, we computed the Root Mean Square Error for the comparison.

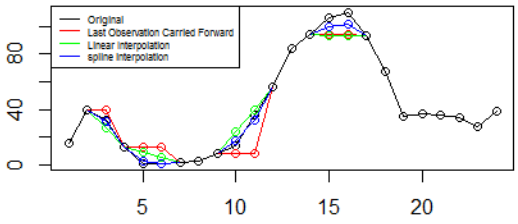



Figure 31: 3 filling methods on day 1, rmse(locf)=8.093207, rmse(linear)=5.273207, rmse(spline)=2.306192

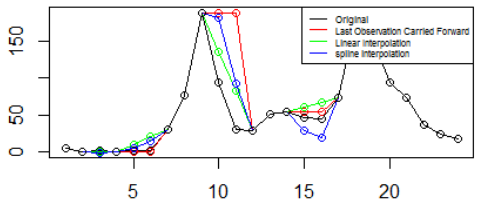



Figure 32: 3 filling methods on day 10, rmse(locf)=37.4316, rmse(linear)=15.07681, rmse(spline)=22.98093

From the root mean square errors of the three methods on workingday (day 10) and non-workingday (day 1), we found linear or spline interpolation fit our dataset better. We chose a combination of linear and spline interpolation to fill the whole dataset: we first used cubic spline to fill the dataset, then the rest miss values are filled by linear interpolation.

### 3.3.3 K-means

After filling up the missing values, we performed k-means from 2 to 9 clusters. The following is the plot of these clusters with the first two principle components as the axis.

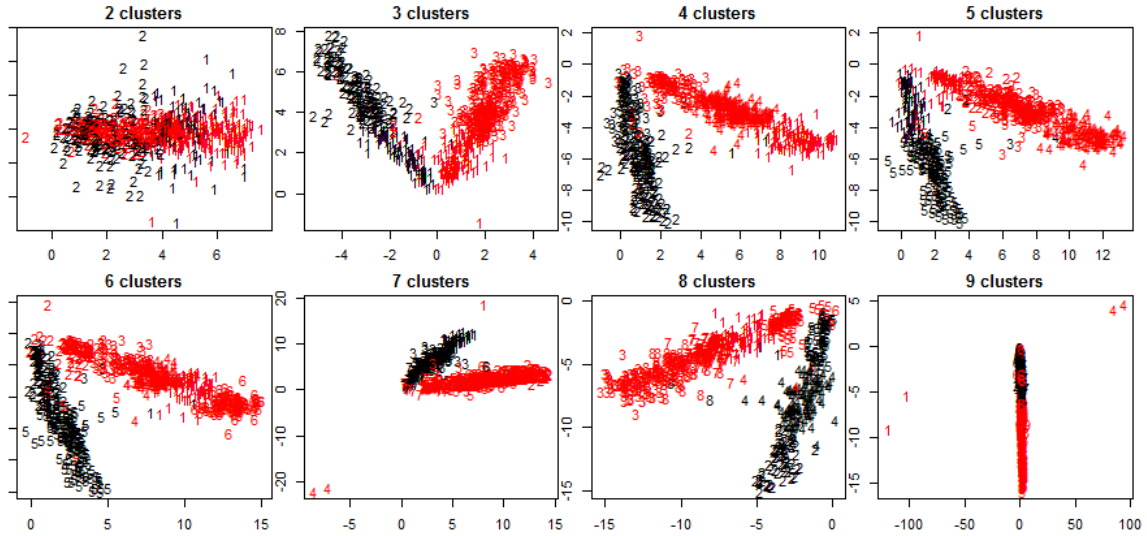



Figure 33: Clusters with workingday (red) and non-workingday (black)

We can see that workingday give a good separation of clusters when there are 3 to 8 clusters. To investigate what are other attributes used to separate the data in k-means, we printed out the average temperature on each cluster with the workingday. We use k=4 as an example:

```
Cluster #1
Working days: 246 (100%)
Total days: 246
Cluster #2
Working days: 107 (53.8%)
Total days: 199
Cluster #3
Working days: 4 (2.84%)
Total days: 141
Cluster #4
Working days: 143 (98.6%)
Total days: 145
```

```
Cluster #1
Avg temperature: 0.4974
Total days: 246
Cluster #2
Avg temperature: 0.3197
Total days: 199
Cluster #3
Avg temperature: 0.5463
Total days: 141
Cluster #4
Avg temperature: 0.5776
Total days: 145
```

As we can see, cluster 1 and 4 are mainly composed of workingdays: 100% in cluster 1, 98.6% in cluster 4; cluster 3 are mainly composed of non-workingdays (97.26%). Cluster 2 is a mixture of workingdays and non-workingdays, but the average temperature in cluster 2 is obviously lower than that in other clusters. Therefore, workingday and `temp` are good attributes to separate the clusters.

A drawing of average hourly counts on each cluster confirmed our findings:

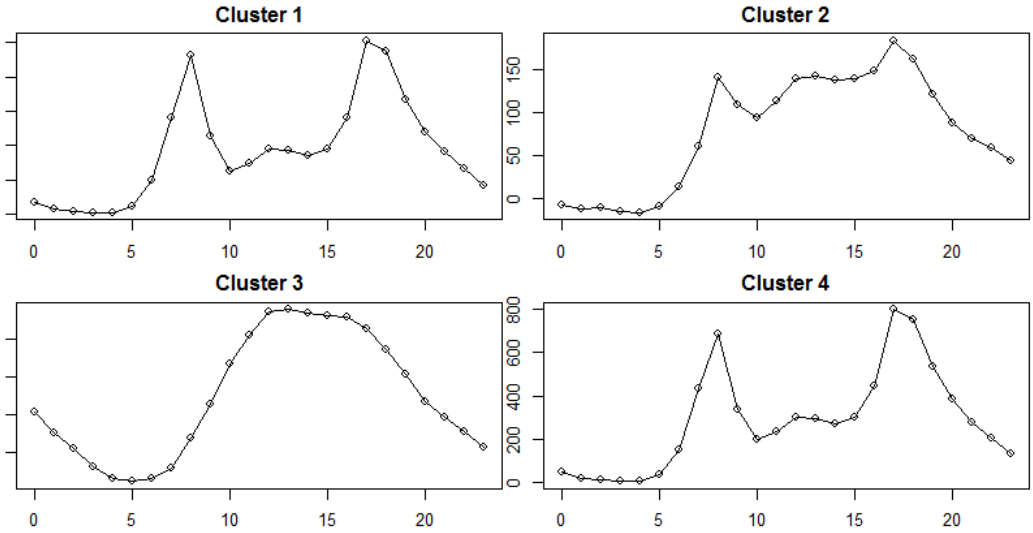



Figure 34: Average counts plot on each cluster

The following are our observations:

1. Cluster 1 and cluster 4 are having the pattern of workingdays. However, the counts in cluster 1 is lower which indicates cluster 1 is representative for days with the lower temperature.

2. Counts in cluster 2 look like a mixture of working and non-working days with low counts. It is more likely in the winter.

3. Cluster 3 are high counts with the pattern of non-workingday.

### 3.3.4 DBI

To determine how many clusters we should choose for the k-means, we used Davies-Bouldin index method. After 8 runs, we found the following graph is representative with the average minimum DBI=8.

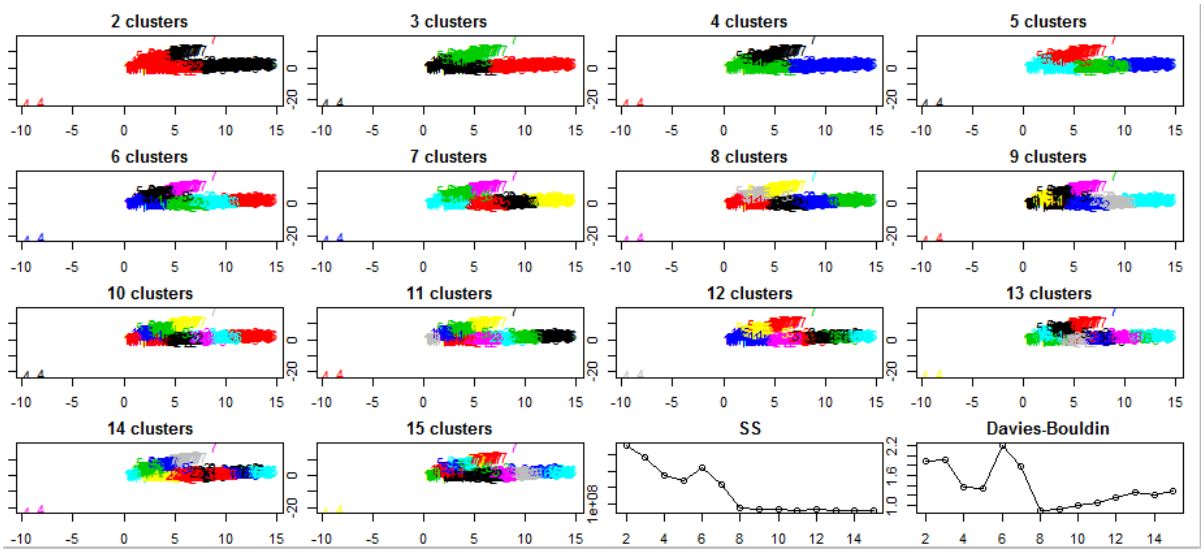



Figure 35: 2 to 15 clusters with DBI

## 3.4 Data Reduction: Outlier Detection

Another way to reduce the data is to remove extreme values. By doing so, we can find a better representation of the data. In this section, we consider to use Interquartile Range ( "A filter for detecting outliers and extreme values based on interquartile ranges.") in WEKA and lofactor in R to detect outliers.

We first tried Interquartile Range filter in WEKA. This "smart" algorithm gave us 21 outliers which are all holidays:

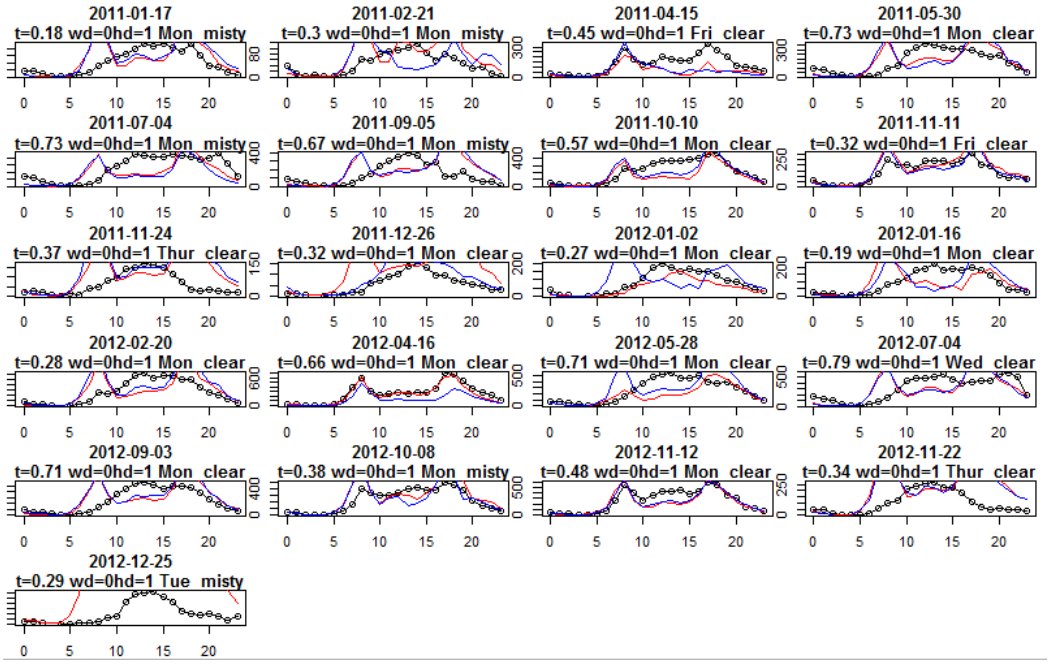



Figure 36: Outliers detected by WEKA (black line is the count on this day, red is 7 days ago, black is 7 days after)

We can see that the black lines are very different than red lines (7 days ago) and blue lines (7 days after) in all 21 days. This is because all of these 21 holidays are happening on weekdays. When the holiday happens on weekdays, the bike rental pattern is more similar to that of non-workingdays.

We decided to remove the holiday information in the dataset and ran Interquartile Range filter again. Then we obtained the following days as outliers:

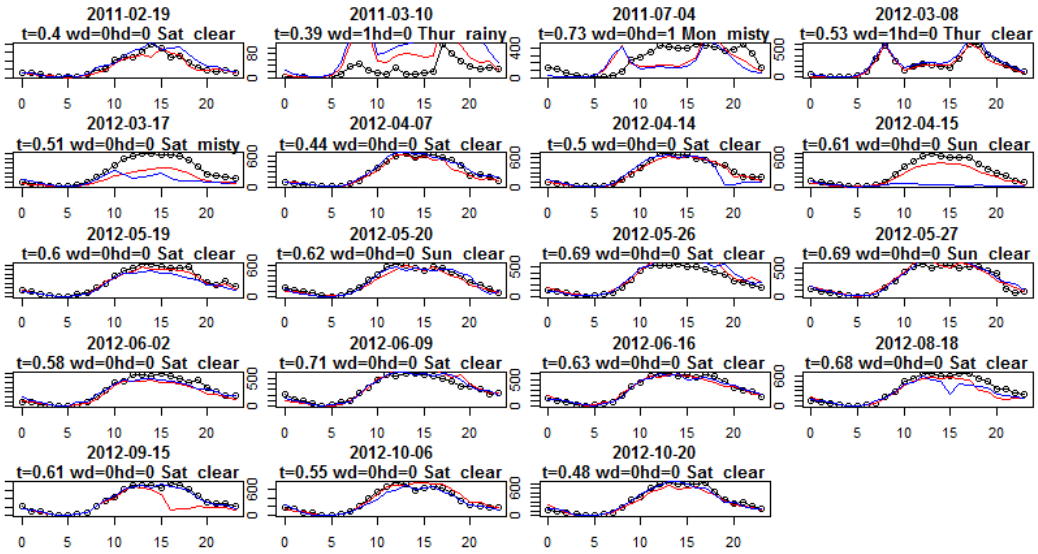
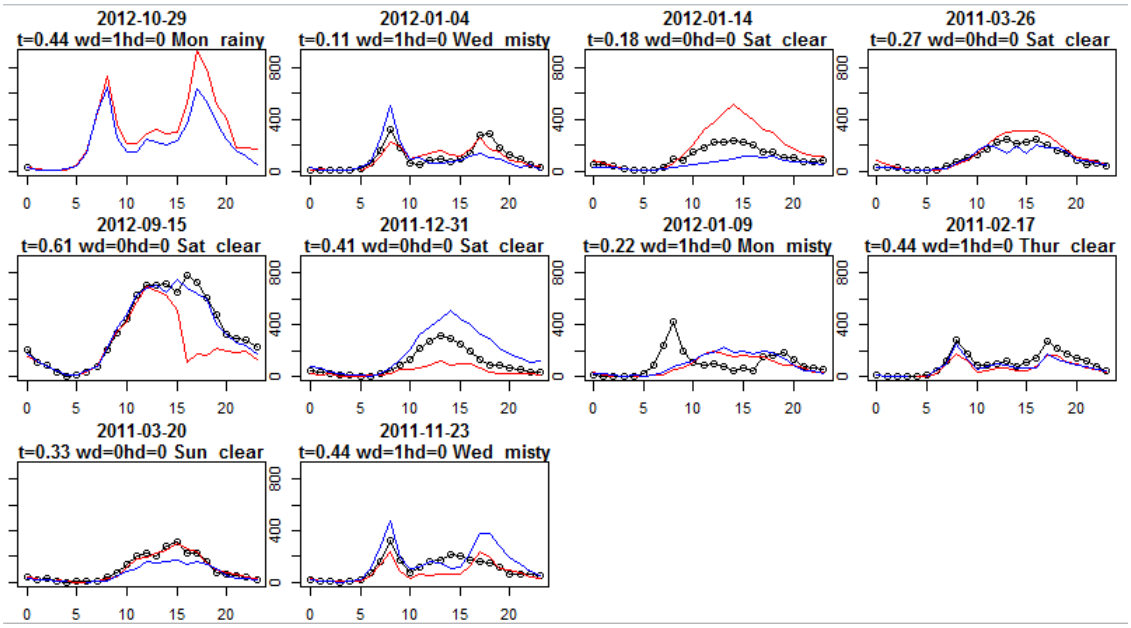



Figure 37: Outliers detected by WEKA without holiday information

It seems that only on 2011-03-10 and 2011-07-04, the black lines are different than red/blue lines from Figure 37.

We also tried the lofactor outlier detector in R:




We noticed that on 2012-10-29, there is a big storm based on Gama's paper about event detection [1], the detector in R caught this. We then used the events detected in Gama's paper as a baseline comparison for our outlier detectors.

**Table 4** Detected Events after verification phase by background knowledge

| Date | Event | Impact |
|---|---|---|
| 29-10-2012 | **Sandy** | 5 |
| 30-10-2012 | **Sandy** | 5 |
| 19-10-2012 | **Storm** | 5 |
| 04-07-2012 | **Washington DC fireworks** | 5 |
| 23-11-2012 | **Black Friday** | 4 |
| 24-12-2012 | **Christmas day** | 4 |
| 08-10-2012 | **Columbus memorial celebration** | 4 |
| 27-05-2012 | **Memorial day** | 4 |
| 22-11-2012 | **Annual Thanksgiving day** | 3 |
| 12-11-2012 | **Veterans day** | 2 |
| 16-04-2012 | **Tax day** | 1 |
| 23-03-2012 | National cherry blossom festival | 5 |
| 18-09-2012 | Heavy rain | 5 |
| 18-07-2012 | Severe thunderstorm | 5 |
| 01-06-2012 | Tornado | 4 |
| 04-12-2012 | Warm weather floods | 4 |
| 13-05-2012 | Bike DC | 3 |
| 11-02-2012 | Cupid Undie run 2012 | 3 |
| 23-01-2012 | March for life | 3 |
| 29-09-2012 | Green festival Washington DC | 3 |
| 25-11-2012 | The coldest morning of the season | 2 |
| 07-10-2012 | Unseasonably cool weather | 2 |
| 07-04-2012 | D.C. United vs. Seattle Sounders FC | 2 |
| 26-05-2012 | D.C. United vs. NE revolution | 2 |
| 21-05-2012 | Occasional showers and storms | 2 |
| 15-09-2012 | United vs. NE revolution | 2 |
| 11-10-2012 | D.C. Baseball v.s Tigers | 2 |
| 12-10-2012 | Hockey Capitals vs. NJ devils | 2 |
| 29-01-2012 | Occupy DC | 1 |
| 19-05-2012 | Survive DC 2012 | 1 |

Bold items are verified detected events (Events with $z$-score $\geq 2$) and non-bold items are those events that their $z$-score $\prec 2$. The numbers in third column is the impact rate (from 0 to 5) given by a human domain specialist for that date indicating the impact of event

Figure 38: red: events found by detector in WEKA with holiday information, yellow: events found by detector in WEKA without holiday information, green: events found by detector in R with holiday information

As we can see from the above table, our outlier detectors do help find some events

which influence the bike rental. We can eliminate these days found by the three outlier detectors. However, we are also removing points which might be useful since there might be a lot of false alarms on the days detectors found.

A better way to approach is separate data into day of events or not and train models separately on these two datasets. When a new data instance come, we first let the detector decide if it is event day, and pass it to correct model.

# 4 Supervised Learning

The most obvious supervised learning task is to predict `cnt`, which is a regression problem, since `cnt` is a continuous variable. This has practical applications, as any company managing a bicycle sharing system might be interested in having a predictive models that helps ensure the supply of bicycles is always appropriate.

## 4.1 Data Imputation

As discussed in section of clustering, we observed that some rows were missing in the hourly Bike Sharing datasets. In total, 165 observations were missing in 76 different days. We decided that it would be wise to impute the missing data for the supervised learning, since it could help build a better regression model. Moreover, having no missing data would make the R programming significantly easier since the dataset structure would become more regular and hence much easier to work with.

### 4.1.1 Imputation of Numeric Variables

We decided to use the `zoo` package to handle data imputation. The numeric variables (i.e. `temp`, `atemp`, `hum`, `windspeed`, `casual`, `registered`, `cnt`) were imputed using linear interpolation. This was very straightforward and produced results which we deemed acceptable for most variables such as `temp`, `atemp`, `hum` and `windspeed`. However, for `casual`, `registered` and `cnt`, the imputated data was clearly wrong in a few isolated cases. In the vast majority of the cases, only 1 or 2 observations were missing in a 24-hour period, in which case the linear interpolation was deemed appropriate. In fact, this was the case for 68 out of the 76 days with missing data. However, there were 8 days with a large number of missing observations (between 6 and 23 missing observations per 24-hour period). We decided to investigate visually the results of our imputation method for those days by constructing Figure 39 below.

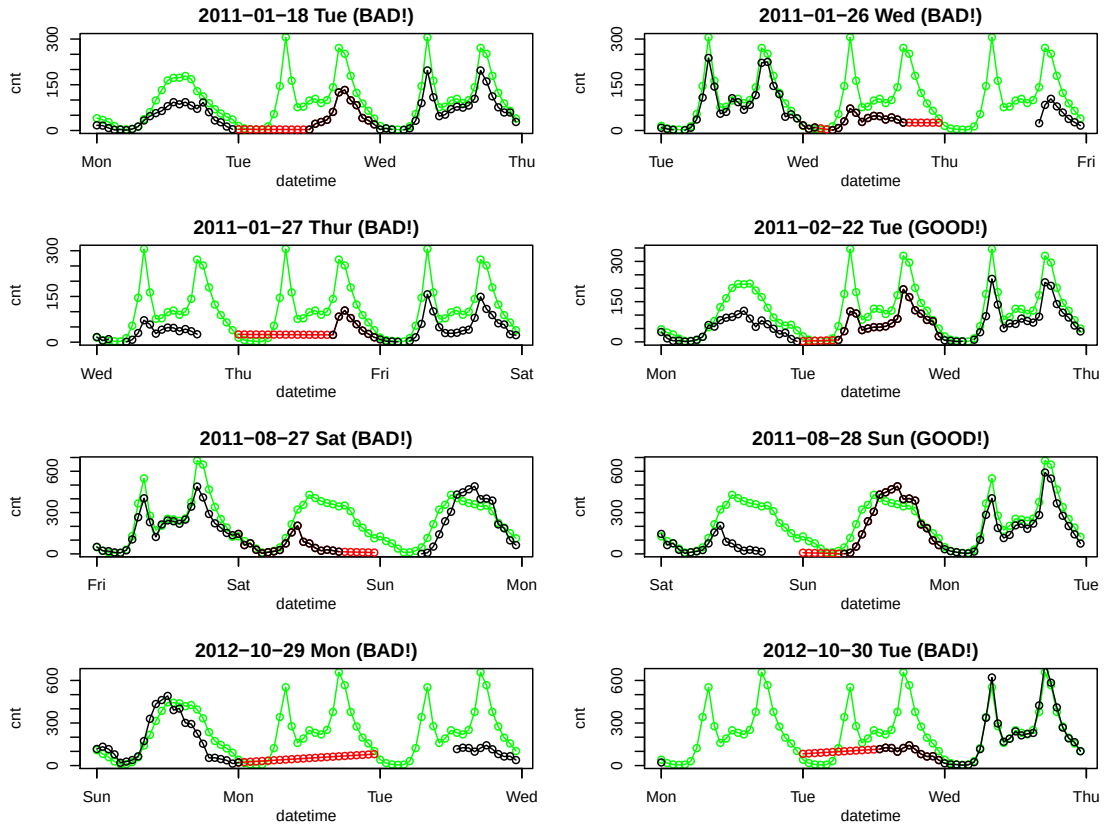



Figure 39: Imputation of missing observations of `cnt` for the 8 days with the highest number of missing data. Each day is plotted along with 24 hours before and 24 hours after, in order to see the continuity between successive days after data imputation. The **black** curve represents the actual values of `cnt` from the dataset. The **red** curve are the observations that were imputed using linear interpolation. The **green** curve represents the average value of `cnt` at the same hour on the same day of the week over the same month (the **green** curve therefore provides a rough idea of what a typical day looks like and helps us decide if the imputation is acceptable or not).

Without much surprise, linear interpolation of `cnt` produced poor results in 6 out of the 8 cases illustrated in Figure 39. However, the imputation on dates 2011-02-22 and 2011-08-28 produced good results. Indeed, the missing observations for those two days were during the hours of the night, during which `cnt` is fairly flat, hence linear interpolation produces acceptable results. On the other hand, the missing observations from the other 6 days appear during the morning, afternoon or evening, during which `cnt` is highly non-linear. It would still be possible to impute missing data using linear interpolation if very few successive observations (perhaps one or two) were missing, but this is not the case here, which explains the poor results for those 6 days.

At this point, we had a choice to either (1) design more sophisticated imputation methods or to (2) simply discard those 6 days with poor imputation results. Since the vast majority of imputation was successful (70 out of 76 days), we decided that it was

best to go with option (2) and simply flag as "bad rows" those 6 days from Figure 39 for which imputation of `cnt` was poor. This allows us later to simply exclude those days completely in the training phase of supervised learning.

### 4.1.2 Imputation of Categorical Variables

At this point, we imputed all the numeric variables for all missing rows, but we are still missing all the categorical variables. Some of them such as `season`, `yr`, `mnth`, `hr`, `weekday` are very obvious. They can all be derived uniquely from the corresponding `datetime` variable in each new row.

The categorical variables that are less obvious are `holiday`, `workingday` and `weathersit`. We decided to impute those values by taking a majority vote using all the other existing observations in the same day. In the case of `holiday` and `workingday`, this will obviously give the correct result. In the case of `weathersit`, this might not be exact, but it should be a decent approximation most of the time and we decided that the simplicity outweighed the disadvantages.

## 4.2 Kaggle Competition

The UCI Bike Sharing Dataset is featured as a Kaggle problem called *Bike Sharing Demand*[1]. We decided to participate in the competition, so that we could compare the performance of our method with others.

The competition started on 28 May 2014 and ends on 29 May 2015. As of 13 April 2015, approximately 2900 people or team have participated in this competition, submitting more than 26000 entries.

The Bike Sharing dataset provided by Kaggle is essentially the same as UCI's – there are only minor differences in the format, such as the date and hour being specified by only one column named `datetime` and non-normalized units used for temperature, humidity and wind speed.

The data must be split into a training set and a testing set, where the training set is comprised of the first 19 days of every month and the testing set is comprised of the 20th day until the end of the month.

The objective of the competition is to train a classifier on the training data and to predict the testing data's `cnt` field with the lowest possible root-mean-squared-logarithmic error (RMSLE).

Users can submit their predictions online and the RMSLE will be calculated

Note that the testing data on Kaggle excludes the `cnt` field. As such, users must submit their predictions only to discover its RMSLE. However, the full data is present in the UCI's dataset and so, we can compute the RMSLE of our predictions by ourselves.

[1] www.kaggle.com/c/bike-sharing-demand

## 4.3 Experimental Design

In all the supervised learning models that follows, we used the dataset `bike.hifx`, which stands for Bike Hourly Imputated Feature eXtracted. In other words, we started with `bike.hourly`, applied data imputation (as described in Section 4.1) to obtain `bike.hi` and finally we performed feature engineering (Section **??**) to obtain `bike.hifx`.

In all cases below, we kept the following features to in the training and testing sets:

```
training.features <-  c("season","yr","mnth","hr","weekday","workingday",
"weathersit","atempdiff","hum","windspeed")
```

In each case, we split the training set and the dataset as specified in the Kaggle competition. That is, we defined the following variable

```
is.training.day <- as.POSIXlt(bike.hfx$date)$mday <= 19
```

and split the data into a training and a testing dataset according to `is.training.day`.

Finally, in each case (except for the random forest model), we actually built two models, identical in all respects, except that the first model was trained to perform a regression on the variable `casual`, while the second model performed a regression on the variable `registered`. Then, the predictions of each model were simply added to each other and the results became the predictions for `[cnt`.

We found early one that building two models based on `casual` and `registered` always produced better results. This makes sense, since some amount of information is lost when considering only `cnt`, which is the sum of `casual` and `registered`. Moreover, the patterns of bike rentals for `casual` and for `registered` are very different, as can be seen from Figure 40. Indeed, `registered` tend to be very regular and features shark spikes in the morning and after-noon in week days (rush hour), whereas `casual` is much less regular and more difficult to predict, as the variation in demand is much greater.

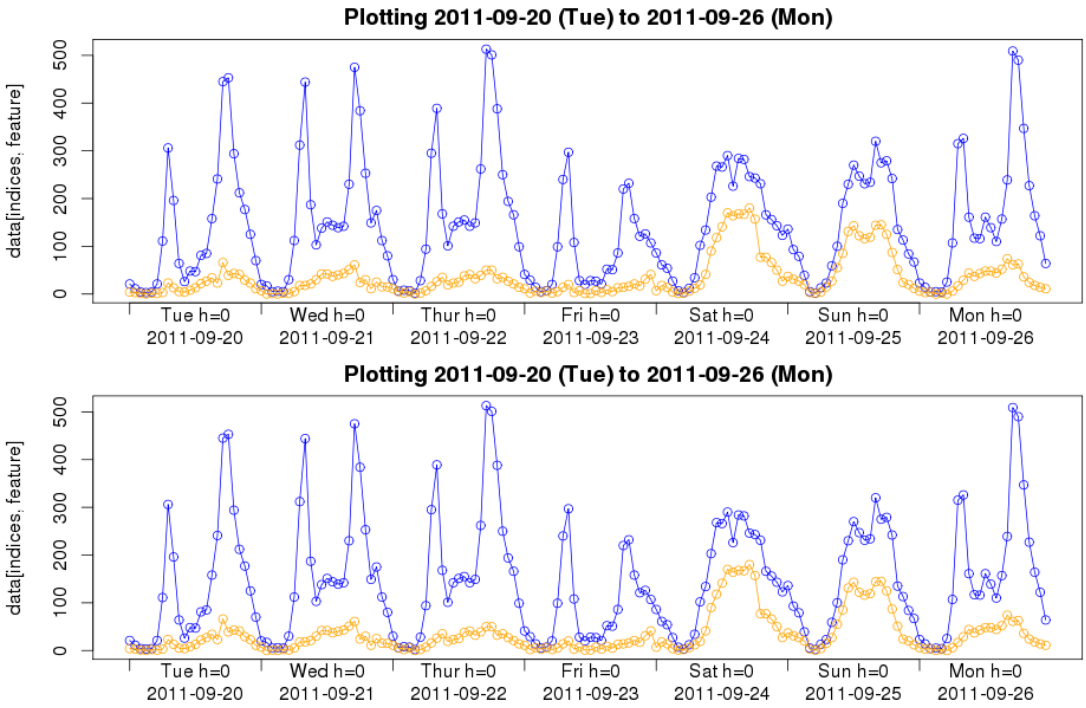



Figure 40: Comparison of `registered` (in blue) versus `casual` (in orange).

### 4.3.1 Random Forests Regression

We started by building a random forest classifier. We used `ntrees=250`. This was the simplest model we investigated and it ran fairly quickly and produced the following results

| Classifier | `htree` | Testing RMSLE |
|---|---|---|
| Random Forest | 250 | 0.7911 |

### 4.3.2 Neural Networks Regression

Next, we trained neural networks with the `nnet` package. We trained the classifier on a grid of parameters, defined by the following two variables:

```
params.h.size <- c(4,9,14,19,24,29,34,39)
params.maxit <- c(100, 500, 1000, 2000)
```

The `delay` used was 0.0001.

We printed the top 5 results, sorted by root-mean-squared logarithmic error (RMSLE) on the testing dataset.

| Time (seconds) | Hidden layer `size` | `maxit` | Testing RMSLE |
|---|---|---|---|
| 13.2 | 5 | 300 | 0.4293 |
| 50.1 | 20 | 300 | 0.4423 |
| 24.7 | 10 | 300 | 0.4443 |
| 138.1 | 40 | 300 | 0.4468 |
| 37.2 | 5 | 1000 | 0.4519 |

”

#### 4.3.3 Ensemble of Neural Networks Regression

We used the `avNNet` package, which builds a collection of neural networks with `nnet` in parallel. Then, it produces the average of all neural networks in its collection when predicting new data. We used a `delay` of 0.0001. See below for the results.

| Hidden layer `size` | `maxit` | `repeats` (i.e. ensemble size) | Testing RMSLE |
|---|---|---|---|
| 14 | 100 | 8 | 0.3984 |
| 4 | 2000 | 8 | 0.4028 |
| 9 | 1000 | 8 | 0.4036 |
| 9 | 500 | 8 | 0.4037 |
| 19 | 500 | 8 | 0.4040 |

This improved our results significantly. What might be surprising the most is that the best model was found in 100 iterations. However, this makes sense, because it is very easy for neural networks to overfit. Indeed, the results of the previous sections always indicate a better performance with smaller number of iterations.

Thus, it makes sense that ensembles of neural networks perform the best, because ensemble methods mitigate generalization errors of individual models.

#### 4.3.4 PCA + Ensemble of Neural Networks Regression

Finally, we were able to improve the results a little bit more by applying PCA and discarding some of the principal components accounting for the least variation in the data. In total, our dataset has 52 features (when accounting for dummy variables introduced by categorical features that have more than 2 categories). The best result was obtained by keeping the top 41 principal components.

| Hidden layer `size` | `maxit` | `repeats` | # of principal components | Testing RMSLE |
|---|---|---|---|---|
| 14 | 1000 | 8 | 41 | 0.3917 |
| 14 | 1000 | 8 | 38 | 0.3924 |
| 9 | 1000 | 8 | 41 | 0.3938 |
| 9 | 1000 | 8 | 38 | 0.4097 |
| 9 | 1000 | 8 | 33 | 0.4390 |

See Figure 41 below for a graph of two weeks of `cnt` data (in black) along with the predictions (in red) of our best model with RMSLE score 0.3917.

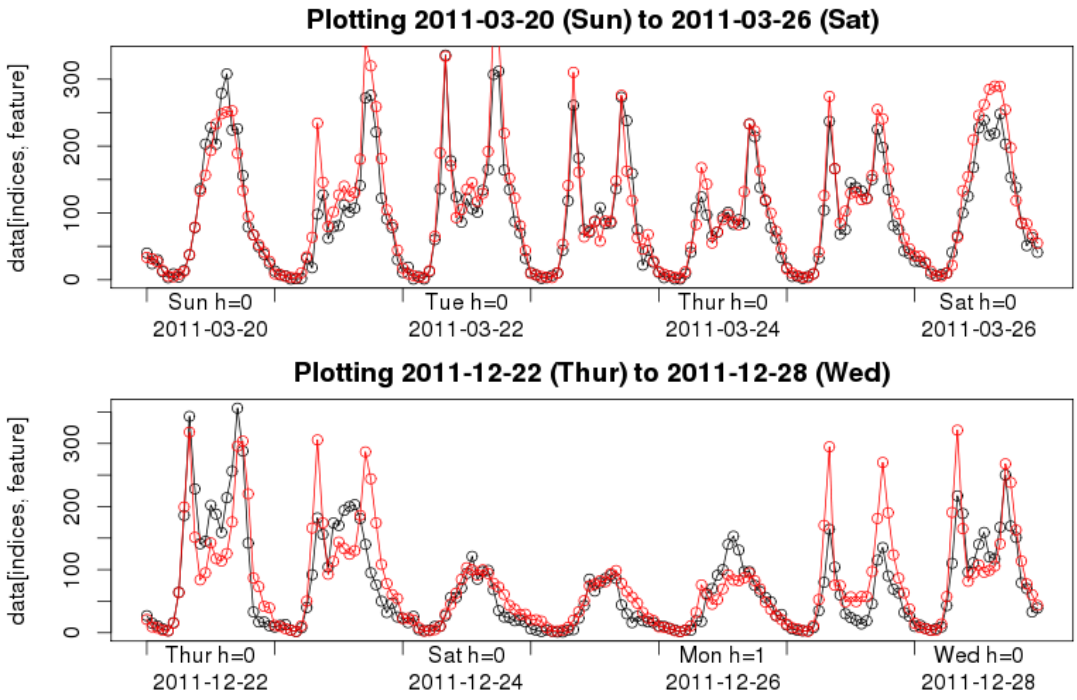



Figure 41: Regression using `avNNet` and PCA with 41 components (out of 52). The correct data is in black and the predictions obtained from the regression is in red.

We submitted our best regression result on Kaggle and obtained the following rank:

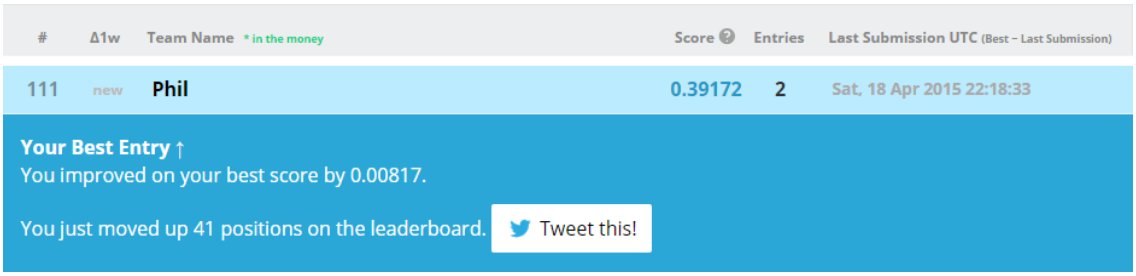



Figure 42: Our best submission on Kaggle.

### 4.3.5 Recurrent Neural Networks Regression

Some recurrent neural networks are particularly well-suited to our task. Indeed, recurrent neural networks are neural networks where connection between neurons are allowed to go backwards. In particular, *Elman neural networks* have all of the neurons in their hidden layer connected pair-wise to a set of *context units*. The context units are also connected pair-wise back to the same hidden neurons, forming a collection of cycles – one for each neuron in the hidden layer. This allows the Elman neural network to "remember" the state of the output units from previous data.

| Hidden layer `size` | `maxit` | Testing RMSLE |
|---|---|---|
| 48 | 3000 | 0.4326 |
| 24 | 3000 | 0.4336 |
| 36 | 3000 | 0.4348 |
| 18 | 500 | 0.4414 |
| 18 | 3000 | 0.4543 |

Table 4: Regression results for the Elman recurrent neural network.

The results were decent, although not as good as in our other models. One of the reasons that this model might not perform as well as we wanted is that Elman networks essentially only "remember" the last state, whereas we might need them to remember at least the last 18-36 states (i.e. the last 18 to 36 hours ago) to really benefit from knowledge about short-term variation in bike demand.

# 5 Conclusion

Through our project, we first tried visualizing the data to gain a general knowledge about the dataset. We have identified temperature, hour, workingday, and month as the most important features for predicting the bike rental demand though the data visualization. To explore more about the dataset, we did association rule mining and found the feature year is also a strong predictor for the bike rental demand.

We then tried unsupervised learning for dimensionality reduction and data reduction. For the dimensionality reduction, we used the PCA to reduce the number of features. For the data reduction, we tried k-means to group the similar instances together, and obtained from the DBI that 8 clusters separate the data very well. Another approach we tried for the data reduction is the outlier detection. compare the outcome of the three outliers we used to the result in Gama's paper about the event detection [1], our detectors can help to find almost half of the events listed in the table from Gama's paper. Overall, by using unsupervised learning, we can find a better representation of the data by summarizing and explaining key features of the data.

Finally, we used supervised learning to predict the total usage of the Capital Bikeshare system on an hourly basis. We built separate models to predict `registered` and `casual`, since we observed that the two patterns were so different. The modeling approaches we used were random forests regression, neural networks regression, ensemble of neural networks regression, and recurrent neural networks regression. To compare the performance of our approaches to that of other people, we participated in the Kaggle competition. Our final and best model ()PCA + ensemble of neural nets) ranked 111 (out of 2696 participating teams) at the time of submission on the Kaggle leaderboard.

# References

[1] 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.

[2] Brandon Harris. A simple model for kaggle bike sharing, 2014.

[3] Jimmy Du, Rolland He, Zhivko Zhechev. Forecasting Bike Rental Demand.

[4] Romain Giot, Raphael Cherrier. Predicting Bikeshare System Usage Up to One Day Ahead. IEEE Symposium Series in Computational Intelligence 2014 (SSCI 2014). Workshop on Computational Intelligence in Vehicles and Transportation Systems (CIVTS 2014), Dec 2014, France. pp.1-8.

[5] Arnab Kumar Datta. Predicting bike-share usage patterns with machine learning. Master's Thesis.