PH125.9x Choose Your Own submission

Lucas Laursen

2022-11-22

1. Introduction

I'm an avid cyclist and wanted to tackle a bike-related problem. I found a handy bike sharing dataset at one of the repositories recommended in our assignment: https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset.

My goal for this is to look at it from the perspective of an operator: how many bikes do I need at any one given time? That would let an operator choose a fleet size, schedule maintenance, and plan upgrades and expansion. In fact, there was a fair amount of debate in my city, Madrid, over maintenance and expansion of our public bike sharing system, and it ultimately led to the revocaton of a contract and city takeover. So if I do this right, I could consider adapting my model to use Madrid's public bike-sharing data (https://opendata.emtmadrid.es/Datos-estaticos/Datos-generales-(1)).

There is a growing academic literature on this and there are way more models than I have time to test (https://link.springer.com/article/10.1007/s11036-021-01737-1). One helpful overview is here: https://link.springer.com/chapter/10.1007/978-3-030-94751-4_25 and I even found a Madrid-specific writeup https://rpubs.com/tfm-bicimad/modelo_demanda.

Following those inspirations and the material from our class, I tried to build a stripped-down, simple model for predicting bike sharing usage. I used R version 4.0.1.

First, I downloaded the data and examined it with some simple summary commands and plots. This gave me an idea of how to organize the modeling. I began preparing the features and separating the data into a training and test set. Then, I applied two different modeling methods to the training data and examined their errors. I chose the one with the lower error to test against the test set. Finally, I commented on the meaning of my model and the limitations I knew of.

1.1 Installing packages and obtaining data

First, let's make sure R has the packages we'll need:

Then download the data.

```
dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/00275/Bike-Sharing-Dataset.zip
hour <- as.data.frame(read.csv(unzip(dl, "hour.csv")))
```

1.2 Data exploration

What variables are there and what data types are they?

```
str(hour)
```

```
## 'data.frame':    17379 obs. of  17 variables:
##  $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ dteday    : chr  "2011-01-01" "2011-01-01" "2011-01-01" "2011-01-01" ...
##  $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ hr        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday   : int  6 6 6 6 6 6 6 6 6 6 ...
##  $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weathersit: int  1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  0.24 0.22 0.22 0.24 0.24 0.24 0.22 0.2 0.24 0.32 ...
##  $ atemp     : num  0.288 0.273 0.273 0.288 0.288 ...
##  $ hum       : num  0.81 0.8 0.8 0.75 0.75 0.75 0.8 0.86 0.75 0.76 ...
##  $ windspeed : num  0 0 0 0 0 0.0896 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ cnt       : int  16 40 32 13 1 1 2 3 8 14 ...
```

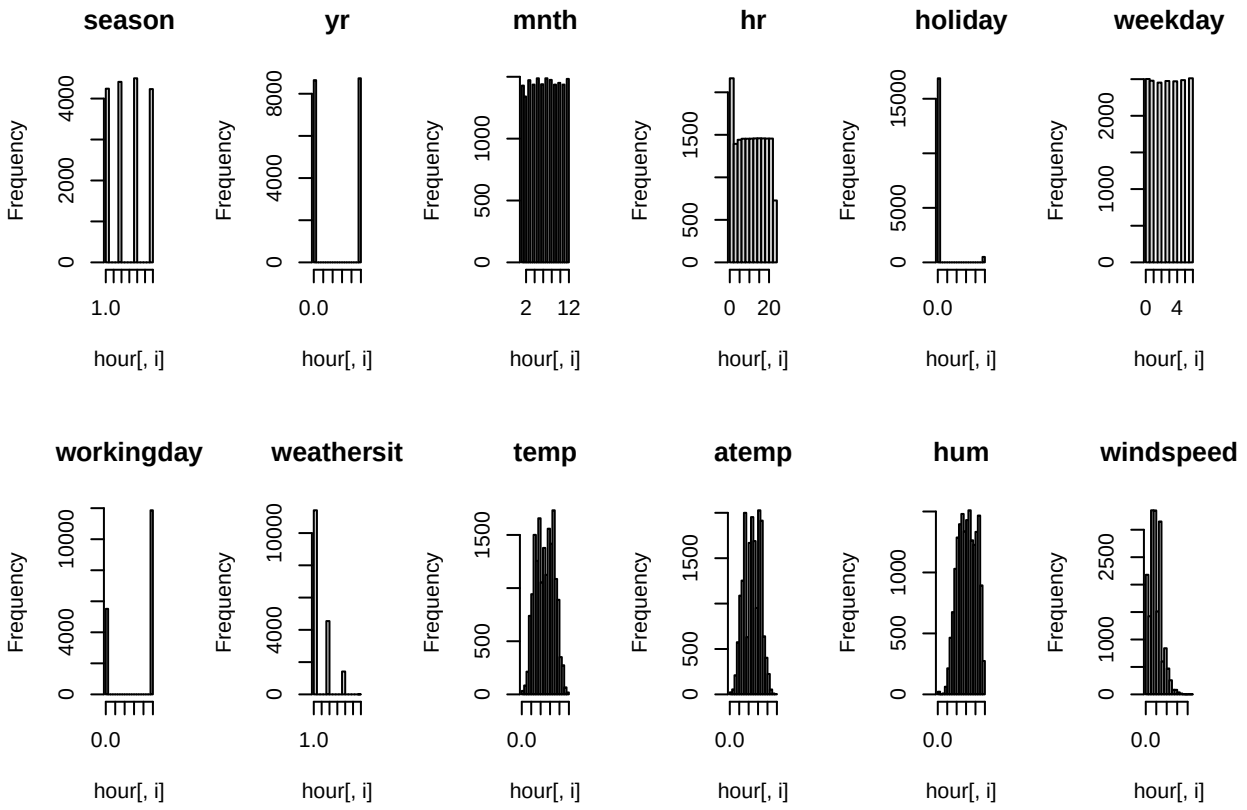
Are there missing values we should know about?

```
table(is.na(hour))
```

```
##
##   FALSE
## 295443
```

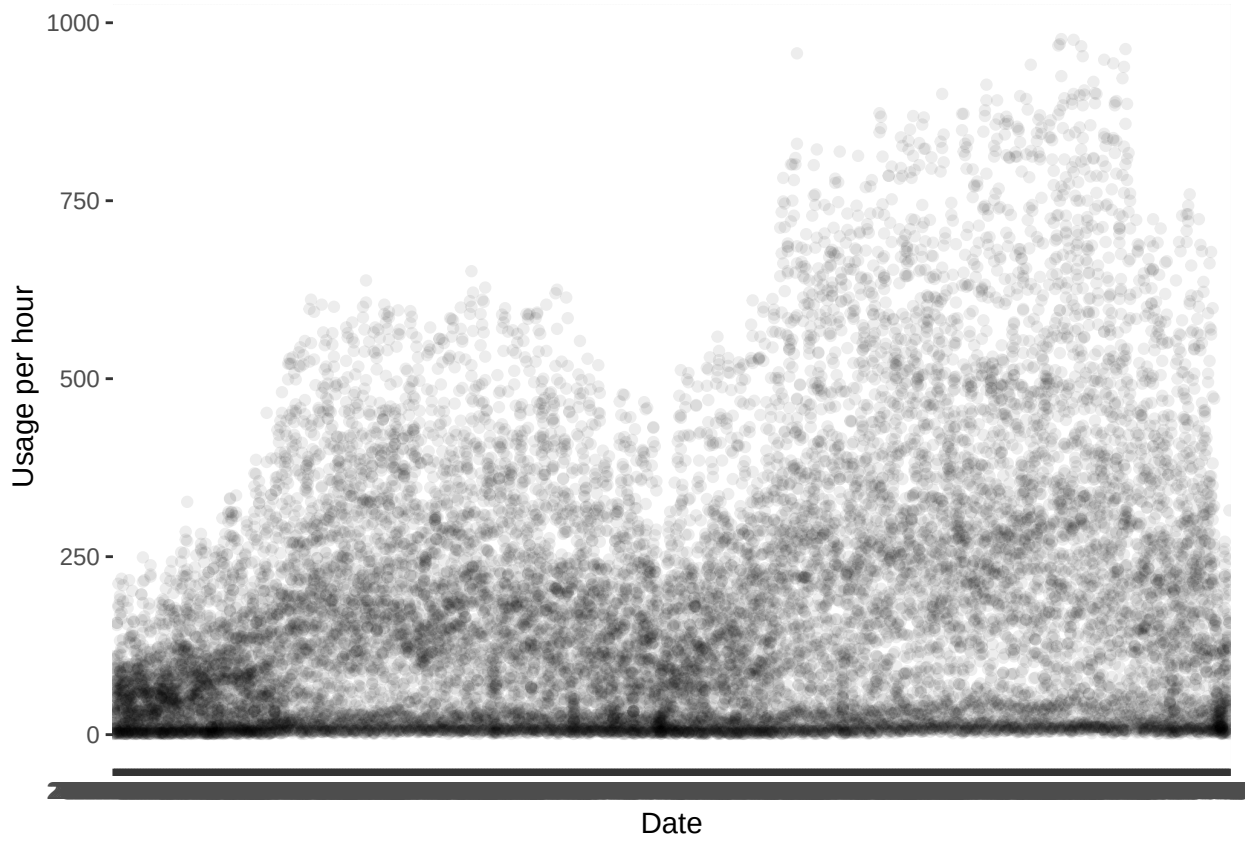
Let's visualize each feature and choose some to focus on in our subsequent modeling. A quick view:

```
par(mfrow=c(2,6))
for(i in 3:14) {
  hist(hour[,i], main = names(hour)[i])
}
```

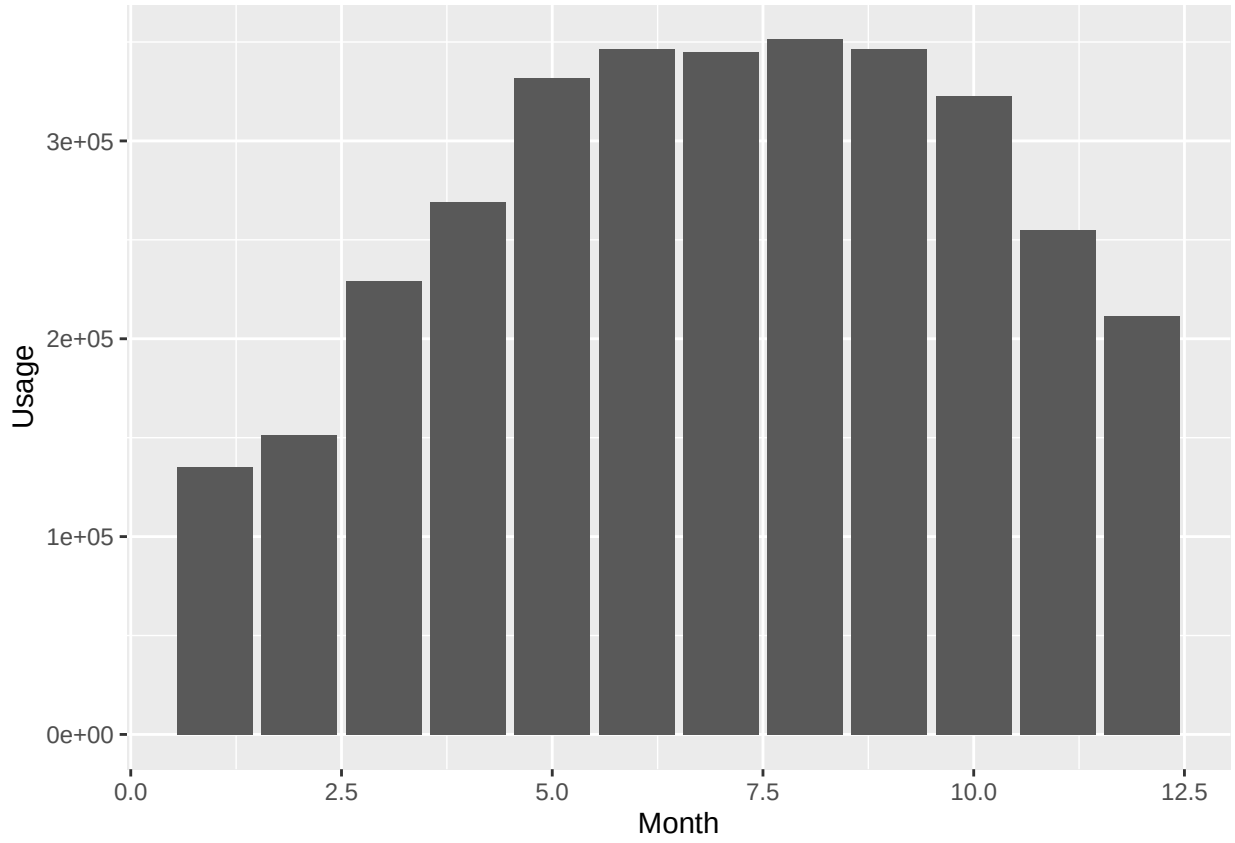


We get an initial sense of how ridership varies with each feature–some more, some less, some in simple patterns, some a little more confusing. Let's look at each feature in more detail.

```
# plot total counts per hour
cnt_dteday<-ggplot(hour,aes(x=dteday,y=cnt))+
  geom_point(alpha=0.07)+
  labs(x='Date', y= 'Usage per hour')
cnt_dteday
```

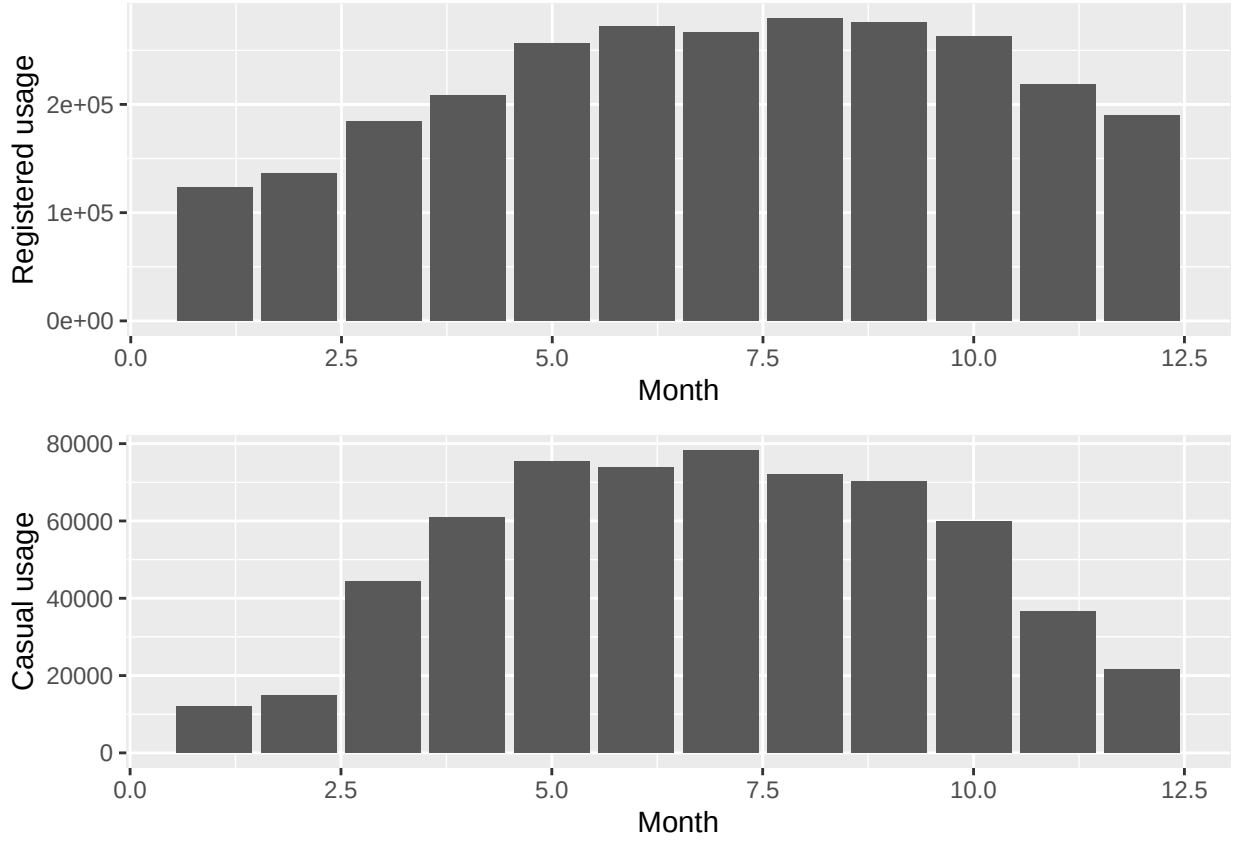


```
# plot total counts per month
cnt_mnth<-ggplot(hour,aes(x=mnth,y=cnt))+
  geom_col()+
  labs(x='Month', y= 'Usage')
cnt_mnth
```



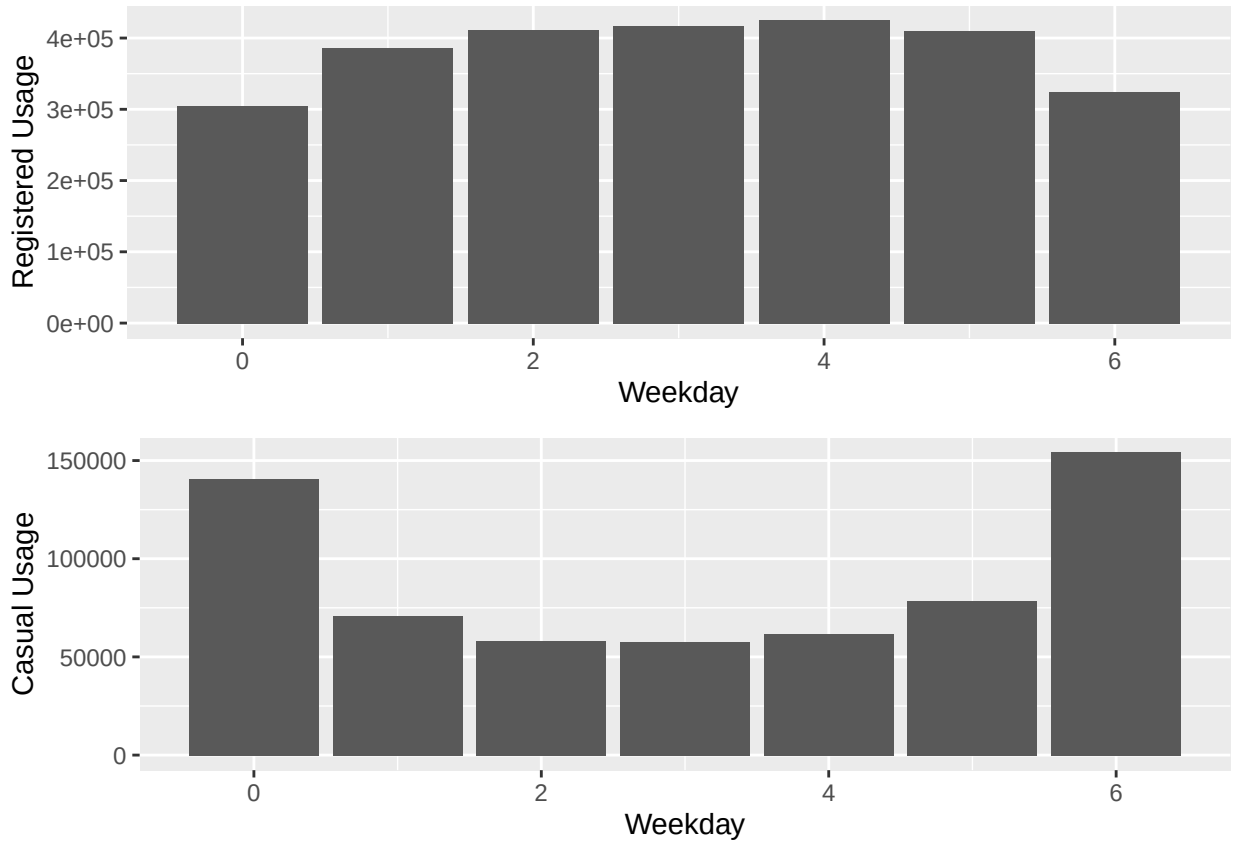
What about the difference between registered and casual users?

```
regcnt_mnth<-ggplot(hour,aes(x=mnth,y=registered))+
  geom_col()+
  labs(x='Month', y= 'Registered usage')
cascnt_mnth<-ggplot(hour,aes(x=mnth,y=casual))+
  geom_col()+
  labs(x='Month', y= 'Casual usage')
l_mnth <- list(regcnt_mnth, cascnt_mnth)
plot_grid(plotlist = l_mnth, nrow = 2)
```

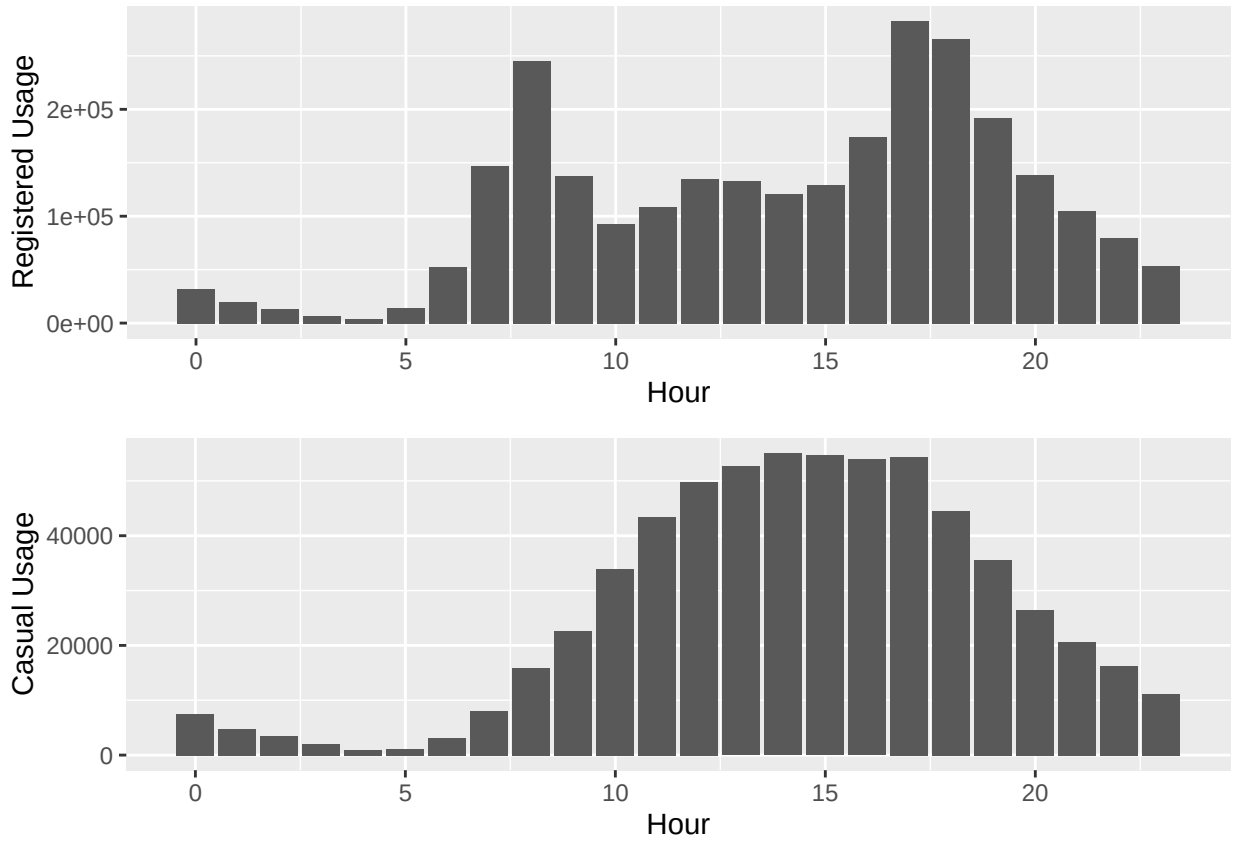


There's a small but noticeable difference, so we'll ask the same question of our other variables

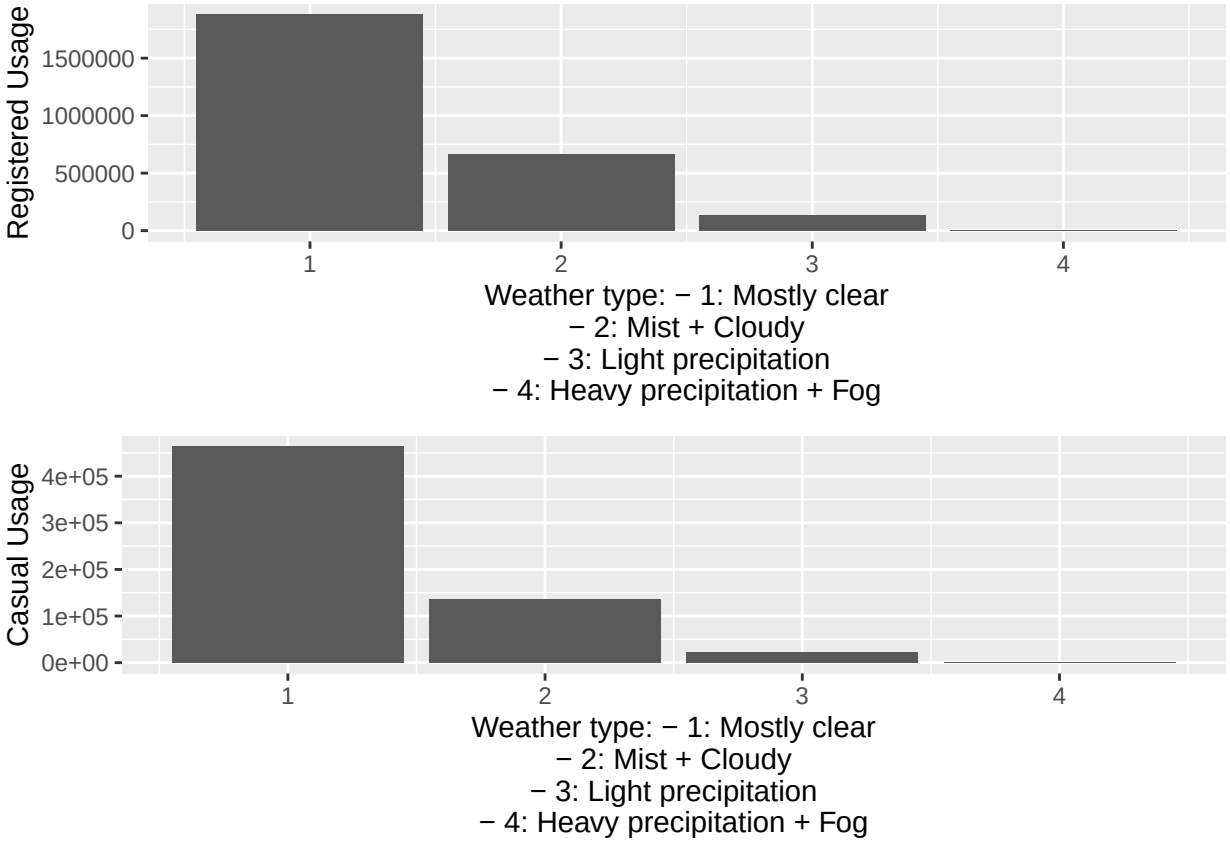
```
# plot separate registered count per weekday + casual count per weekday
regcnt_weekday<-ggplot(hour,aes(x=weekday,y=registered))+
  geom_col()+
  labs(x='Weekday', y= 'Registered Usage')
cascnt_weekday<-ggplot(hour,aes(x=weekday,y=casual))+
  geom_col()+
  labs(x='Weekday', y= 'Casual Usage')
l_weekday <- list(regcnt_weekday, cascnt_weekday)
plot_grid(plotlist = l_weekday, nrow = 2)
```



```
# plot counts per hour of day
regcnt_hour<-ggplot(hour,aes(x=hr,y=registered))+
  geom_col()+
  labs(x='Hour', y= 'Registered Usage')
cascnt_hour<-ggplot(hour,aes(x=hr,y=casual))+
  geom_col()+
  labs(x='Hour', y= 'Casual Usage')
l_weekday <- list(regcnt_hour, cascnt_hour)
plot_grid(plotlist = l_weekday, nrow = 2)
```

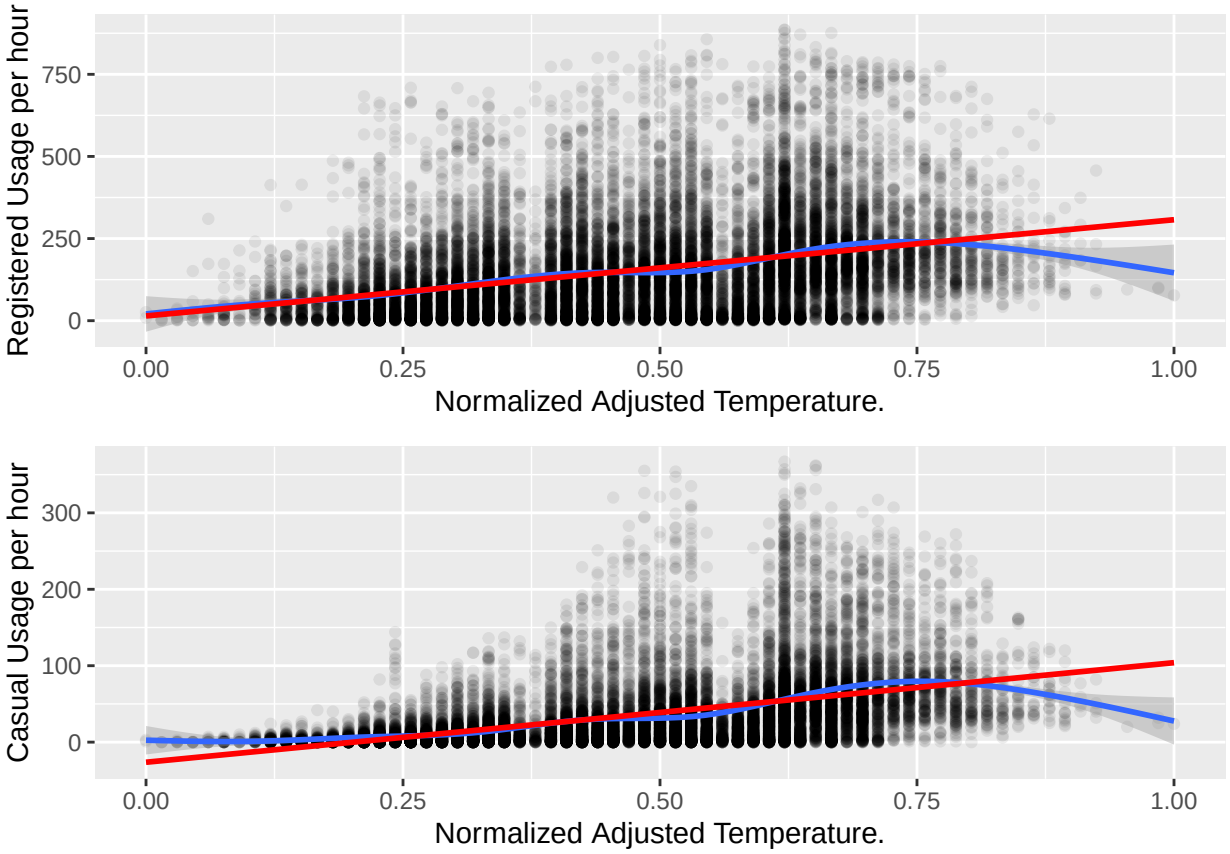


```
# plot counts per weather situation
regcnt_weather<-ggplot(hour,aes(x=weathersit,y=registered))+
  geom_col()+
  labs(x='Weather type: - 1: Mostly clear
- 2: Mist + Cloudy
- 3: Light precipitation
- 4: Heavy precipitation + Fog', y= 'Registered Usage')
cascnt_weather<-ggplot(hour,aes(x=weathersit,y=casual))+
  geom_col()+
  labs(x='Weather type: - 1: Mostly clear
- 2: Mist + Cloudy
- 3: Light precipitation
- 4: Heavy precipitation + Fog', y= 'Casual Usage')
l_weather <- list(regcnt_weather, cascnt_weather)
plot_grid(plotlist = l_weather, nrow = 2)
```



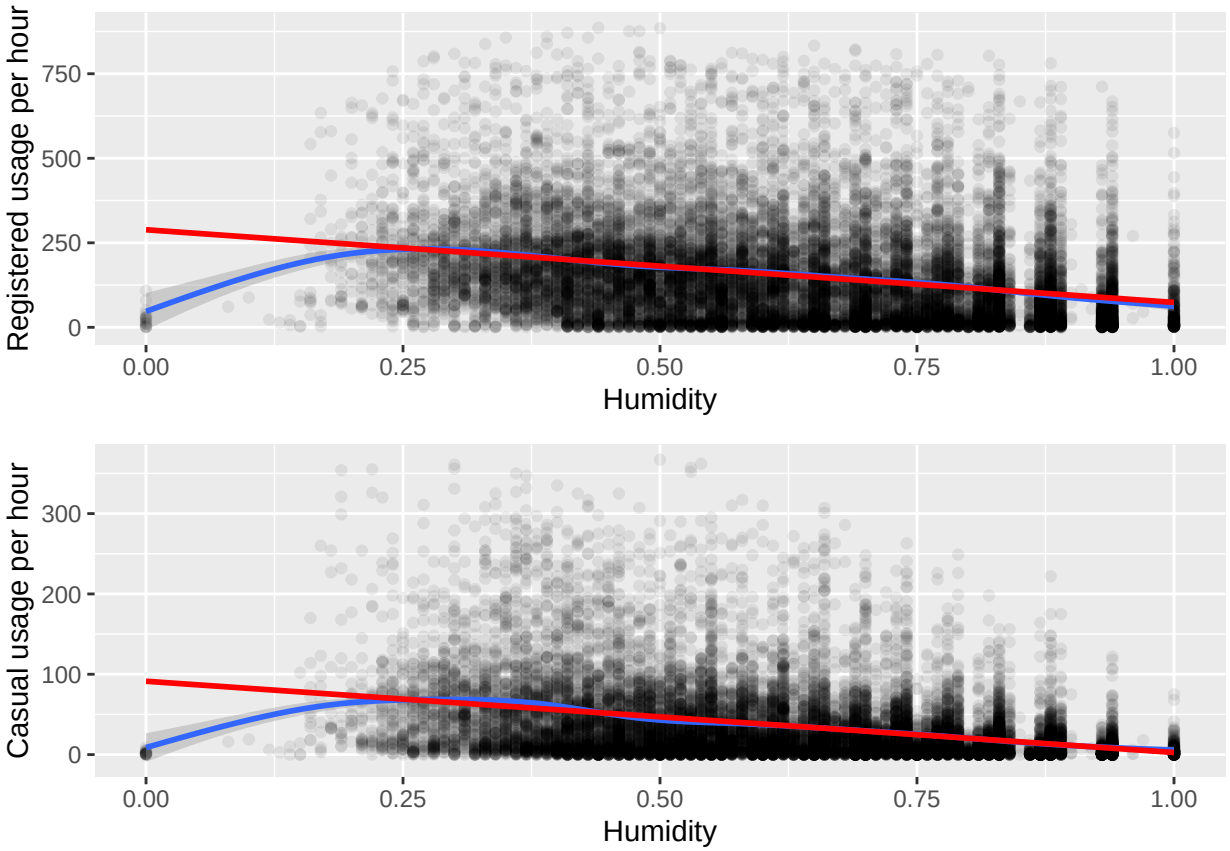
```
# plot count vs. atemp
# (note we are using this and ignoring the temp variable because they correlate)
regcnt_atemp<-ggplot(hour,aes(x=atemp,y=registered))+
  geom_point(alpha=0.07)+
  labs(x='Normalized Adjusted Temperature.', y= 'Registered Usage per hour')+
  geom_smooth(method='auto')+
  geom_smooth(method='lm',color= 'red')
cascnt_atemp<-ggplot(hour,aes(x=atemp,y=casual))+
  geom_point(alpha=0.07)+
  labs(x='Normalized Adjusted Temperature.', y= 'Casual Usage per hour')+
  geom_smooth(method='auto')+
  geom_smooth(method='lm',color= 'red')
l_atemp <- list(regcnt_atemp, cascnt_atemp)
plot_grid(plotlist = l_atemp, nrow = 2)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



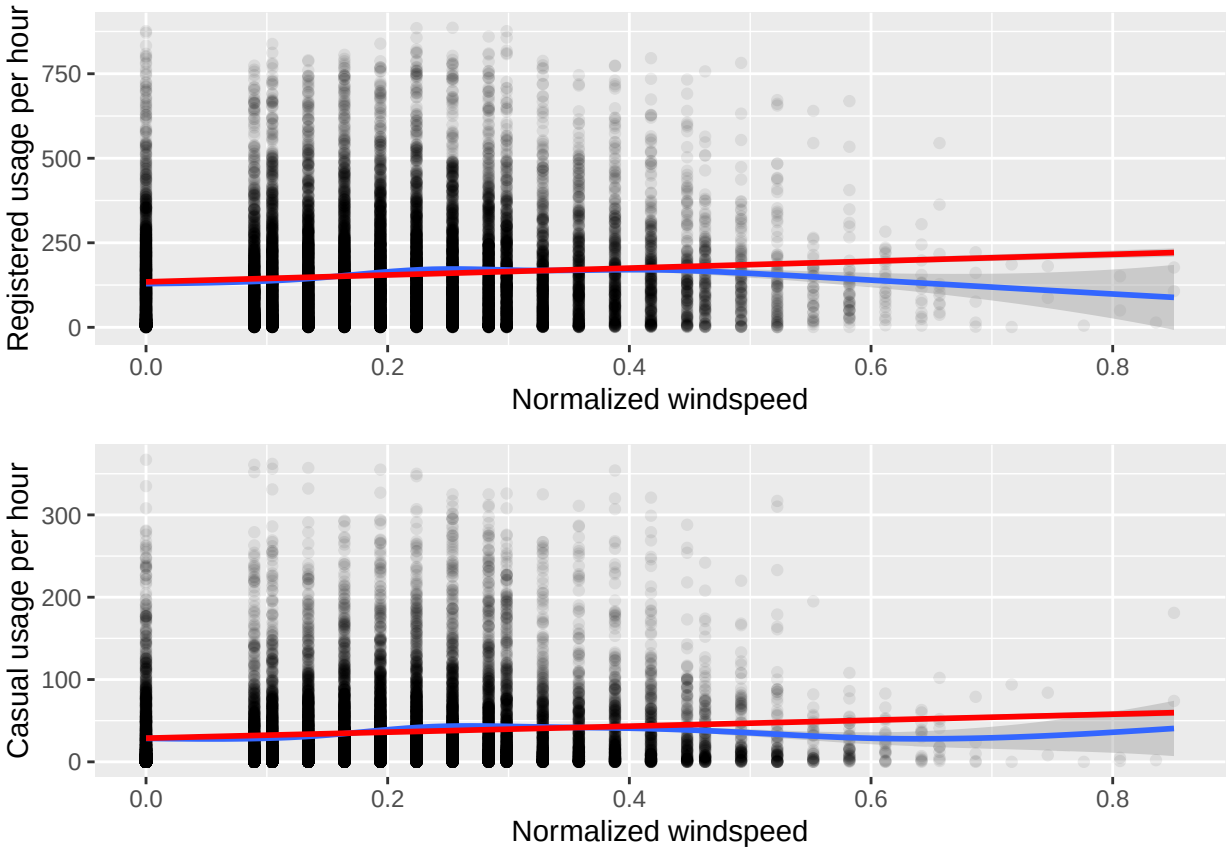
```
# plot count vs. humidity
regcnt_hum<-ggplot(hour,aes(x=hum,y=registered))+
  geom_point(alpha=0.07)+
  labs(x='Humidity', y= 'Registered usage per hour')+
  geom_smooth(method='auto')+
  geom_smooth(method='lm',color= 'red')
cascnt_hum<-ggplot(hour,aes(x=hum,y=casual))+
  geom_point(alpha=0.07)+
  labs(x='Humidity', y= 'Casual usage per hour')+
  geom_smooth(method='auto')+
  geom_smooth(method='lm',color= 'red')
l_hum <- list(regcnt_hum, cascnt_hum)
plot_grid(plotlist = l_hum, nrow = 2)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



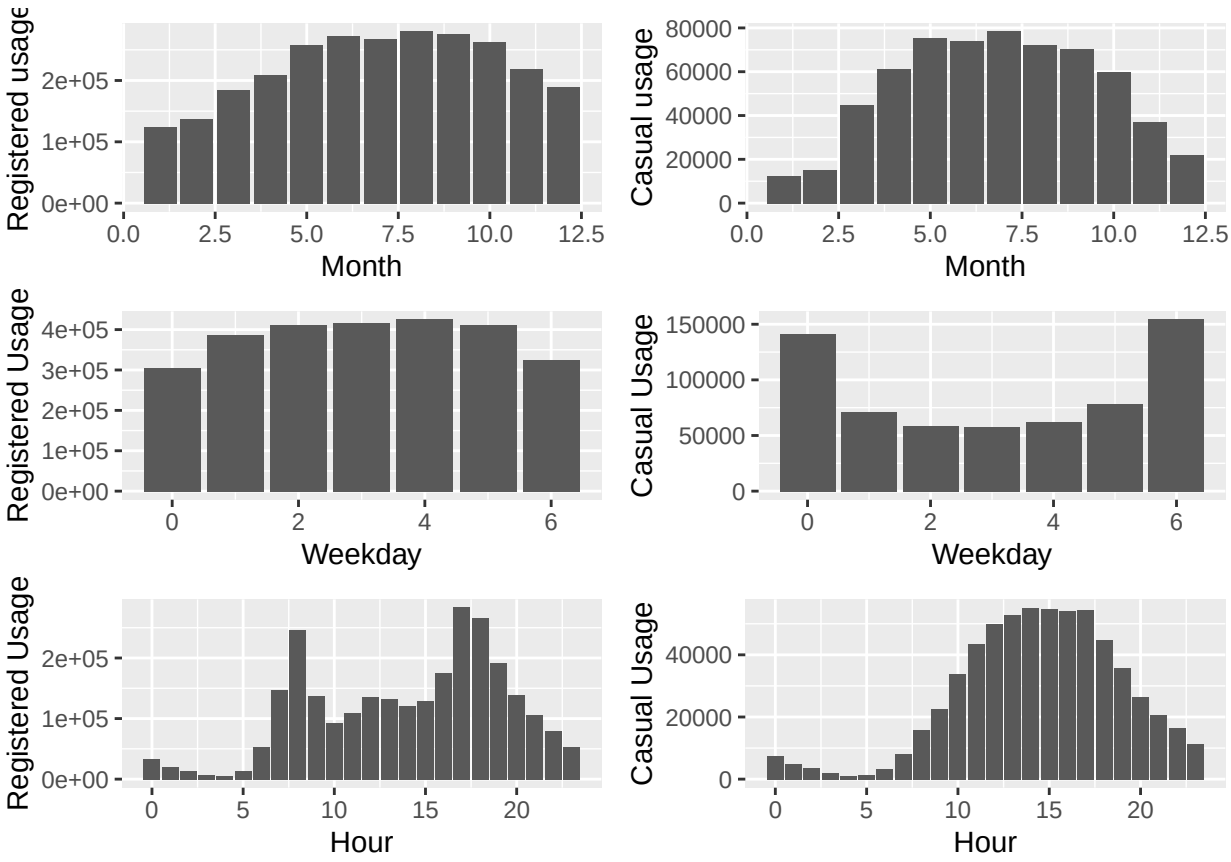
```
# plot count vs. windspeed
regcnt_wind<-ggplot(hour,aes(x=windspeed,y=registered))+
  geom_point(alpha=0.07)+
  labs(x='Normalized windspeed', y= 'Registered usage per hour')+
  geom_smooth(method='auto')+
  geom_smooth(method='lm',color= 'red')
cascnt_wind<-ggplot(hour,aes(x=windspeed,y=casual))+
  geom_point(alpha=0.07)+
  labs(x='Normalized windspeed', y= 'Casual usage per hour')+
  geom_smooth(method='auto')+
  geom_smooth(method='lm',color= 'red')
l_wind <- list(regcnt_wind, cascnt_wind)
plot_grid(plotlist = l_wind, nrow = 2)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



So, to review, there are time dependencies:

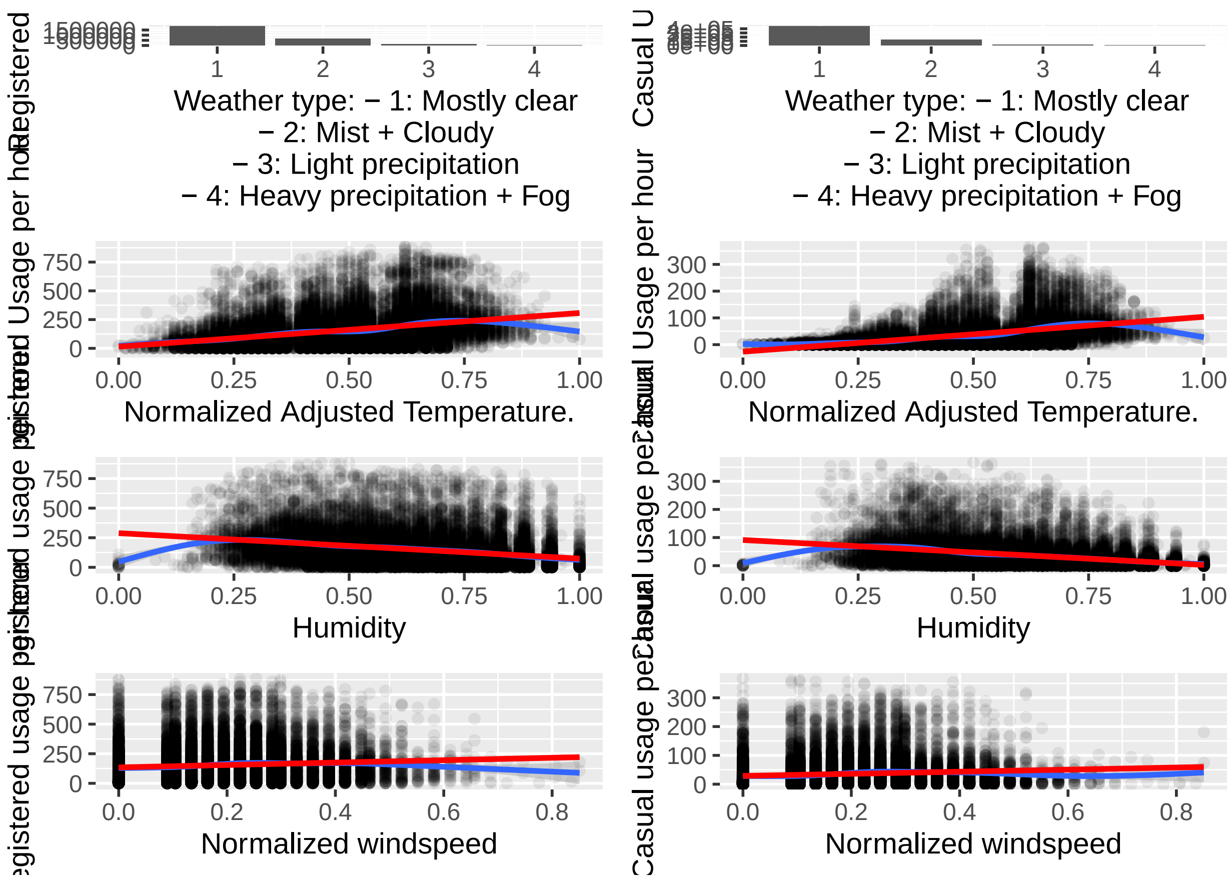
```
l_time<-list(regcnt_mnth, cascnt_mnth, regcnt_weekday, cascnt_weekday, regcnt_hour, cascnt_hour)
plot_grid(plotlist = l_time, nrow = 3)
```



and there are weather dependencies:

```
l_wx<-list(regcnt_weather, cascnt_weather, regcnt_atemp, cascnt_atemp, regcnt_hum, cascnt_hum, regcnt_w
plot_grid(plotlist = l_wx, nrow = 4)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



Looks like we'll want to build separate models for registered and causal users, though there are so many more casual users than registered users that it may not make a huge difference.

We clearly need to take into account the hour of the day, the day of the week, and the day of the year. Those are cyclical.

We'll also want to take weather into account, via the temp & humidity variables at least, because of their larger effects. Windspeed seems like less of a priority.

2. Methods/Analysis

2.1 Data wrangling

We'll start with a little wrangling to get things ready for the analysis/modeling.

First, discard things we don't plan to use, such as the index column.

```
hour <- select(hour,-1, -11)
```

Then prepare some of the things we will use, such as converting the dteday column into R-parsable dates...

```
hour$dteday <- as_datetime(hour$dteday)
```

...and extracting the day and week of the year from the dteday column and storing them.

```
hour <- mutate(hour, day = yday(dteday))
hour <- mutate(hour, week = week(dteday))
```

The next step is dividing our data into training and test sets. Unlike the Movielens project, we need to do this one chronologically, since the data is time-dependent. We will make a prediction for the last 10% of the time series based on training on the preceding 90%.

```
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
train_index <- c(1:round(nrow(hour)*.9))
train_set <- hour[train_index,]
test_set <- hour[-train_index,]
```

2.2 Data analysis

We can begin with a linear regression, one of the simplest things we can do.

First define x and y data for regression. x is all the features minus the counts, y is the count

```
yTrain <- data.matrix(train_set[c(15)])
xTrain <- data.matrix(train_set[-c(13,14,15)])
```

Make a linear model for the count and measure its mean squared residuals (MSR) as a measure of accuracy– the smaller the better.

```
fit <- lm(yTrain ~ xTrain, data = train_set)
mean(residuals(fit)^2)
```

```
## [1] 19592.4
```

Does modeling casual and registered users separately do any better?

```
yTrain <- data.matrix(train_set[c(13)]) #casual count is in column 13
xTrain <- data.matrix(train_set[-c(13,14,15)])
fit_lmcasual <- lm(yTrain ~ xTrain, data = train_set)
mean(residuals(fit_lmcasual)^2)
```

```
## [1] 1323.211
```

```
yTrain <- data.matrix(train_set[c(14)]) #registered count is in column 14
xTrain <- data.matrix(train_set[-c(13,14,15)])
fit_lmreg <- lm(yTrain ~ xTrain, data = train_set)
mean(residuals(fit_lmreg)^2)
```

```
## [1] 14673.53
```

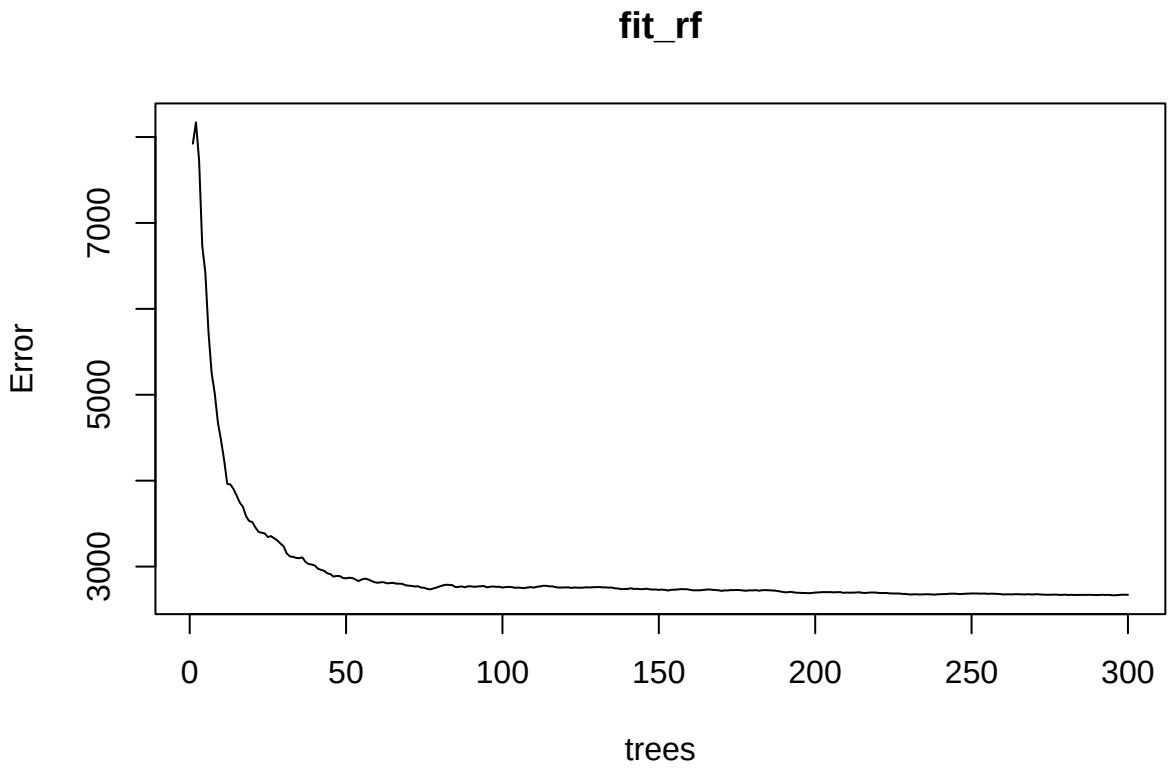
Both models are better than the overall count model and the casual prediction is an order of magnitude better than the registered prediction.

Based on my literature review and the fact that we've worked with them before, I'll try a random forest model next.

```
set.seed(415)
fit_rf <- randomForest(train_set$cnt ~ dteday + season + yr + mnth + week + day + hr + holiday + workin
```

Does it look like we are minimizing the error?

```
plot(fit_rf)
```



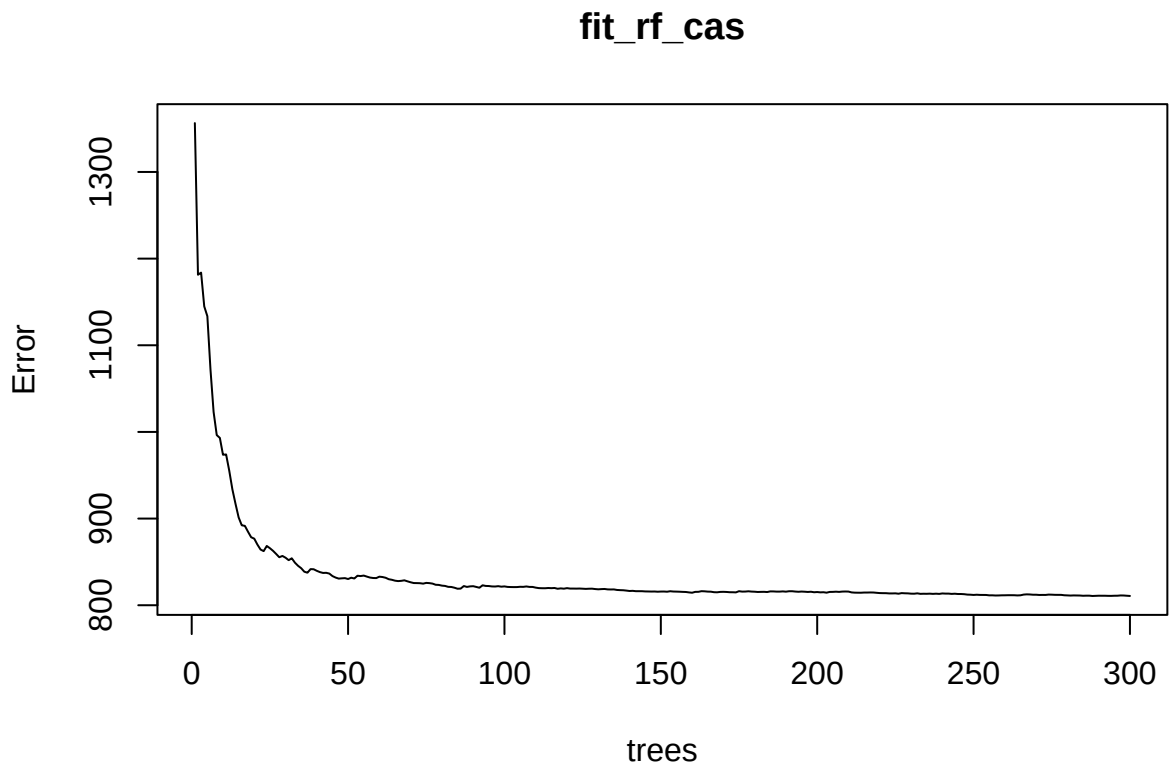
It does seem to level off somewhere around 150 trees after a couple of iterations of trying different numbers. Let's look at the results of the model.

```
fit_rf
```

```
##
## Call:
##  randomForest(formula = train_set$cnt ~ dteday + season + yr +      mnth + week + day + hr + holiday
##                Type of random forest: regression
##                      Number of trees: 300
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 2671.893
##                     % Var explained: 91.81
```

Mean Squared Residual (MSR) is ~2673. What about casual and registered users separately?

```
set.seed(415)
fit_rf_cas <- randomForest(train_set$casual ~ season + yr + workingday + holiday + atemp + hum + windsp
                       data = train_set, importance = TRUE, ntree = 300)
plot(fit_rf_cas)
```



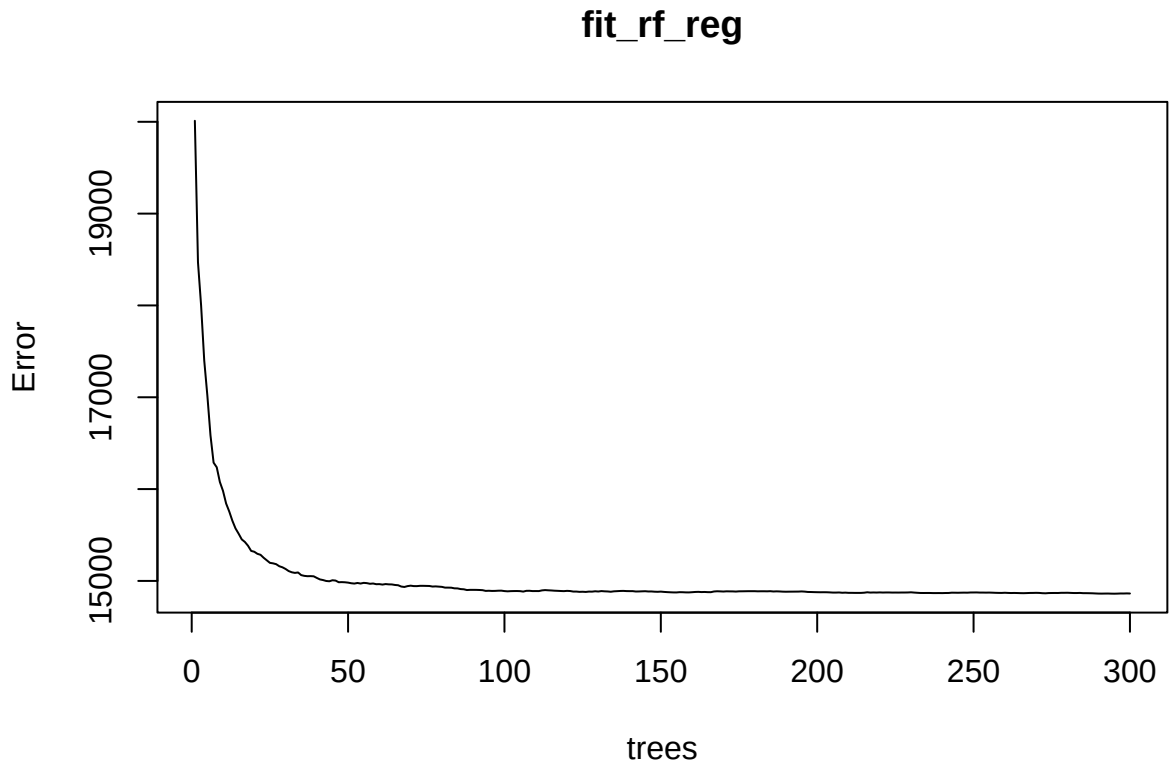
```
fit_rf_cas
```

```
##
## Call:
##  randomForest(formula = train_set$casual ~ season + yr + workingday +      holiday + atemp + hum + wi
##                Type of random forest: regression
##                      Number of trees: 300
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 810.6907
##                     % Var explained: 67.21
```

MSR is ~810, much lower than combined.

```
fit_rf_reg <- randomForest(train_set$registered ~ season + yr + workingday + holiday + atemp + hum + wi
                           data = train_set, importance = TRUE, ntree = 300)
```

```
plot(fit_rf_reg)
```




```
fit_rf_reg
```

```
##
## Call:
```

```
##  randomForest(formula = train_set$registered ~ season + yr + workingday +      holiday + atemp + hum
##                Type of random forest: regression
##                      Number of trees: 300
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 14862.84
##                     % Var explained: 33.61
```

MSR for registered users is ~14,863, way higher than the ~2672 for the combined model! So much for that idea.

Those analyses used only the training data (I specified 'train_set' to create those fits) so if we want to evaluate the models more meaningfully we need to do so with the test dataset, the 10% that we held back higher up.

So we'll make predictions using the three random forest models and compare them to the actual counts in the test set. The way we did that in the demonstration MovieLens project earlier in this class was with the predict() tool and then we calculated the root mean square error (RMSE), so I'll do that here again.

First, define a function to calculate the root mean square error:

```
RMSE <- function(true_cnt, predicted_cnt){
  sqrt(mean((true_cnt - predicted_cnt)^2))}
```

Then use each model to make a prediction for the test set and calculate and store the RMSEs.

```
#Overall count
rfpredictions <-  predict(fit_rf,newdata = test_set)
rf_RMSE <- RMSE(rfpredictions, test_set$cnt)
#Casual riders
rf_caspredictions <-  predict(fit_rf_cas,newdata = test_set)
rf_cas_RMSE <- RMSE(rf_caspredictions, test_set$casual)
#Registered riders
rf_regpredictions <- predict(fit_rf_reg,newdata = test_set)
rf_reg_RMSE <- RMSE(rf_regpredictions, test_set$registered)

results <- tibble(rf_RMSE, rf_cas_RMSE, rf_reg_RMSE)
```

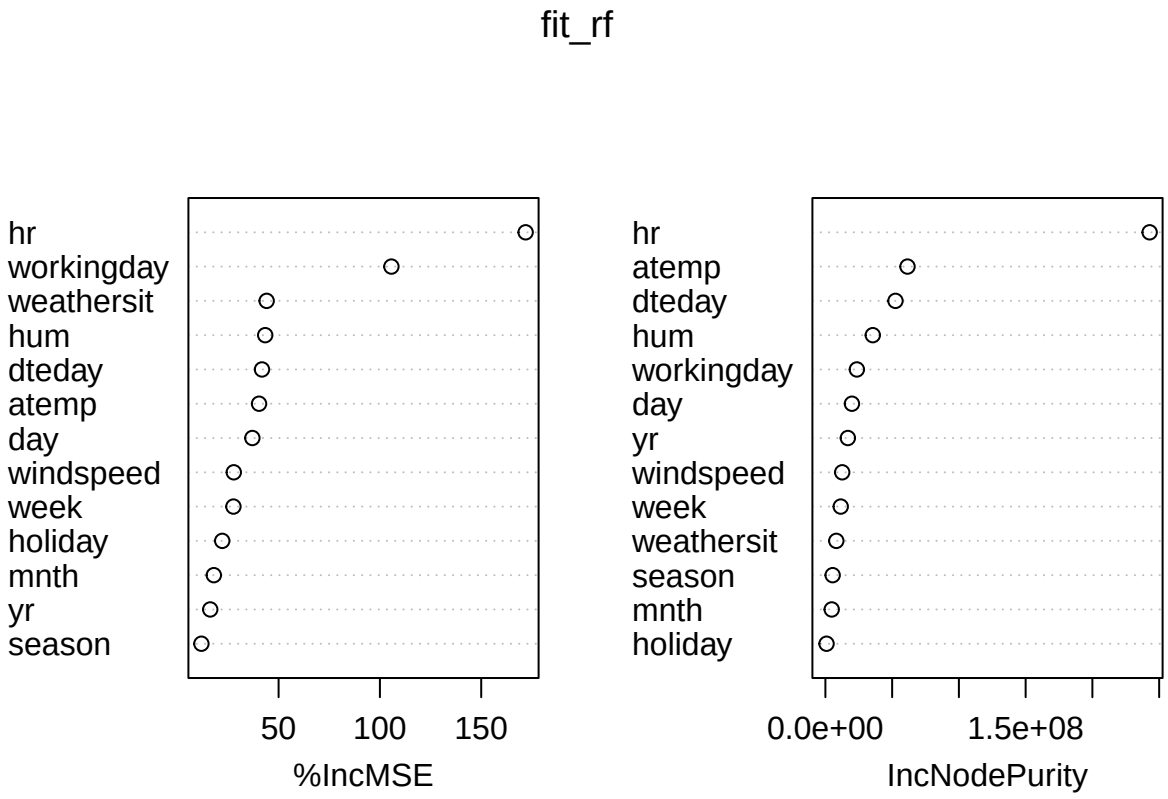
3. Results

```
results
```

```
## # A tibble: 1 x 3
##   rf_RMSE rf_cas_RMSE rf_reg_RMSE
##     <dbl>       <dbl>       <dbl>
## 1    137.        31.1        147.
```

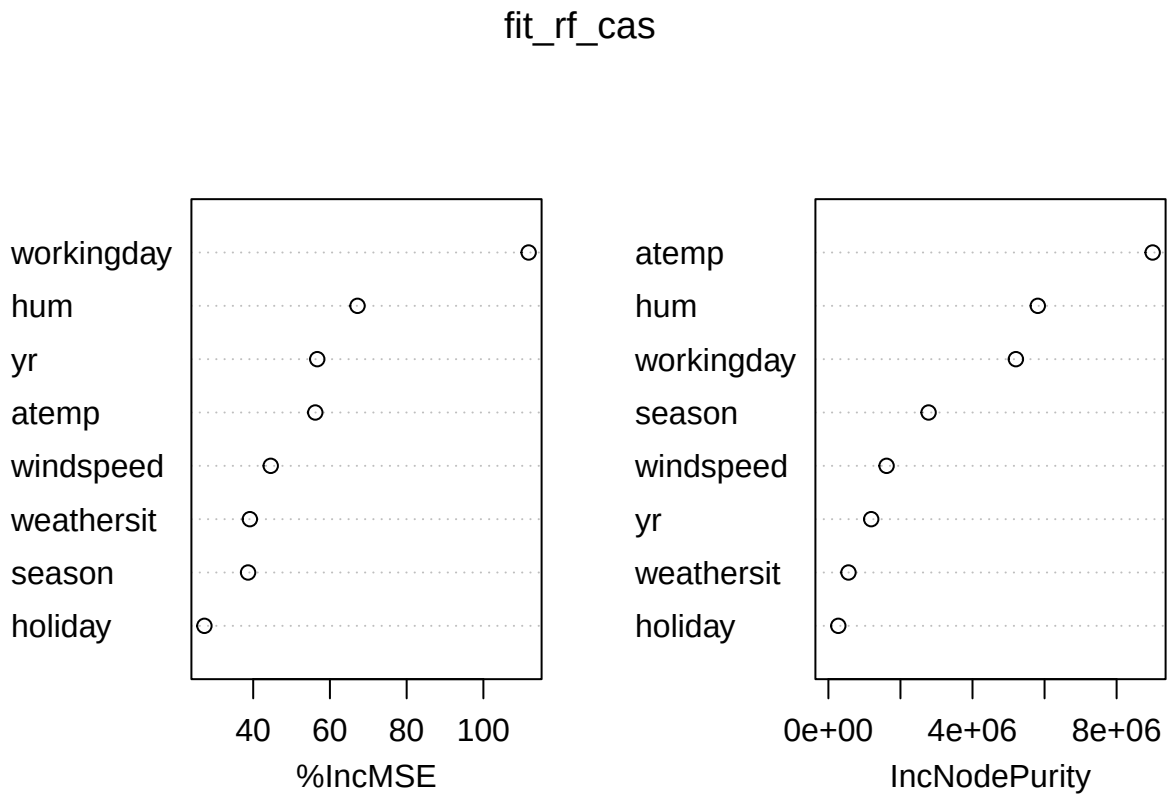
What does it all mean? Let's examine the importance of each feature in the random forest models.

```
varImpPlot(fit_rf)
```



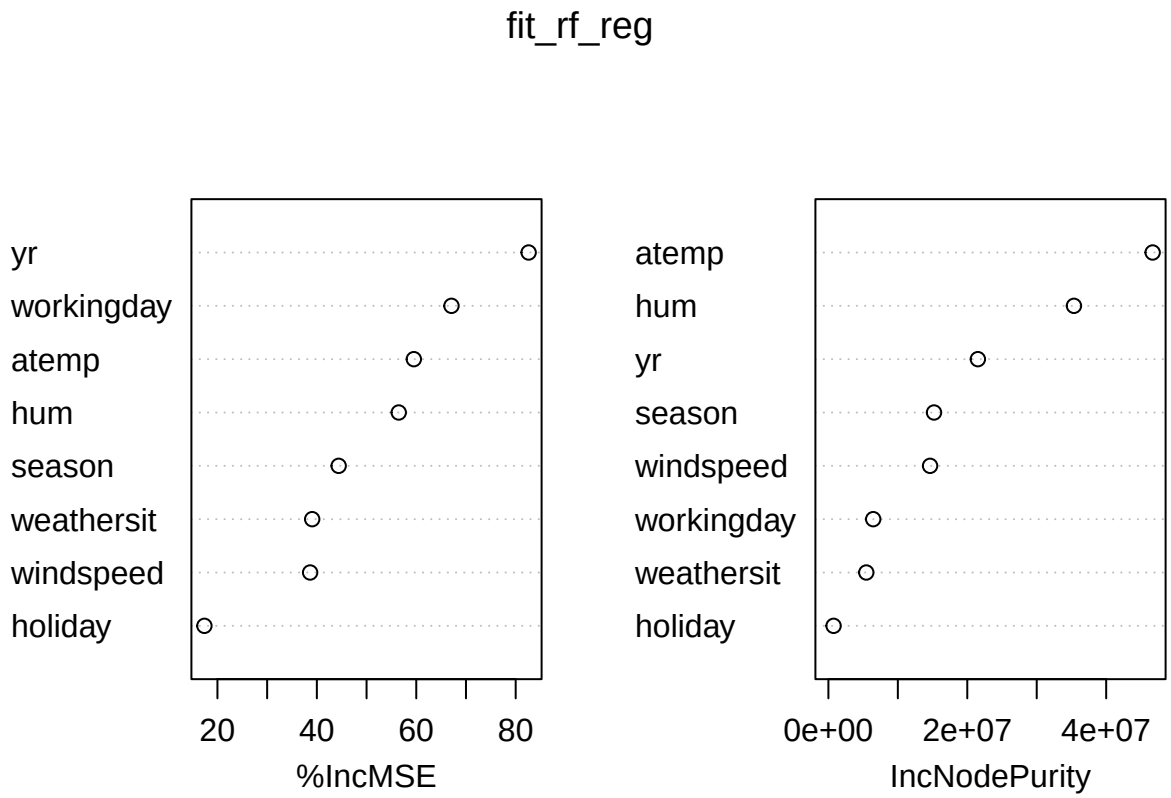
The lefthand plot is the relevant one for us, because we are working with a time series whose order we can't randomize. Looks like the time of day is the most important variable for the whole dataset, followed by the workingday feature, which specifies whether a given day is a workday or not, then followed at a distance by the overall weather situation and humidity and apparent temperature. After the day of the year, the variables drop noticeably in importance.

```
varImpPlot(fit_rf_cas)
```



Casual riders care much more about whether it's a workday or not, followed by humidity. The year seems to matter even more than the apparent temperature. The next variables matter less.

```
varImpPlot(fit_rf_reg)
```



Registered riders, on the other hand, seem to be more affected by the year. The data covered two years, and there was significant growth from the first to the second year, so this could be a sign that the program was going through a growth boom, perhaps even converting some of the first year's casual users into registered users.

The next major features are the working day, apparent temperature, and humidity. These are the same features as the casual riders, but a slightly different order. Rain is less important–maybe commuters pack a rain jacket.

4. Conclusion

This analysis gives us some starting points for understanding this bike sharing program. For one thing, we ought to be careful about any longer-term forecasts given the year-to-year trend and the likelihood that there are bigger issues not present in our data. For example, the bike sharing program might have lowered its prices or run an advertising campaign or the city might have limited personal car access to the center. The data are older, but we know from the news that the pandemic had a huge impact on bike usage and that doesn't appear in weather data or calendars, the main sources of our variable features.

Some of the other features do seem more reliable on the short to medium timescales, such as whether it's a workday or wet outside or hot or cold.

My literature search showed a much longer list of potential factors, including pollution and disease outbreaks. So while this project is a simple initial demonstration any thorough analysis would need to examine those and others and focus on the most useful.

So much for data. On to methods.

This analysis involves just two simple methods, but again, my literature search turned up more than half a dozen models that researchers have used to explore bike sharing data, including decision trees, knn, conditional inference tree, boosted regression, and neural networks. It would certainly be within the scope of this class, but not within the time remaining before the deadline, to try a couple more of those for comparison.

Finally, the question and further questions. I sought to predict usage over the last 10% of the available timeframe in the dataset based on training my models on the first 90%. However, there could be plenty of more specific questions worth exploring. A bike sharing system operator might want to know more about the differences between registered and casual usage, for example. They would surely want to explore station-level data. We are using aggregate system-wide data but that doens't help an operator allocate their bicycle-ferrying trucks. They would also want to explore maintenance data in order to allocate mechanics.

In fact, a fascinating next step for a project like this would be to interview people who actually operate bike sharing systems in order to learn the questions they consider worthwhile. A natural complement would be the actual users. Those, after all, are who a data scientist ought to be thinking about and working for.