Bike Analysis Poisson

2023-11-23

Load Packages

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readr)
library(lubridate)
library(dplyr)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(leaps)
library(ggplot2)
library(dplyr)
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.2
```

```
## corrplot 0.92 loaded
```

Overview

Objective: Create a poisson model to estimate the demand for bikes and test its accuracy and for overdispersion.

Response Variables:

- *Y* (Cnt): Total bikes rented by both casual & registered users together

Qualitative Predicting Variables:

Season: Season which the observation is made (1 = Winter, 2 = Spring, 3 = Summer, 4 = Fall) *Yr*: Year on which the observation is made *Mnth*: Month on which the observation is made *Hr*: Day on which the observation is made (0 through 23) *Holiday*: Indictor of a public holiday or not (1 = public holiday, 0 = not a public holiday) *Weekday*: Day of week (0 through 6) *Weathersit*: Weather condition (1 = Clear, Few clouds, Partly cloudy, Partly cloudy, 2 = Mist & Cloudy, Mist & Broken clouds, Mist & Few clouds, Mist, 3 = Snow, Rain, Thunderstorm & Scattered clouds, Ice Pallets & Fog)

Quantitative Predicting Variables:

Temp: Normalized temperature in Celsius *Atemp*: Normalized feeling temperature in Celsius *Hum*: Normalized humidity *Windspeed*: Normalized wind speed

Load Data

```
data <- read.csv("Bikes.csv", header=T)
head(data)
```

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

Preparing Data for Predicition

We have both qualititative and quantitive predicting variables, which means we have to change the qualitative to factors. We are also removing the columns we don’t need. This includes the record index, date, count of casual users, and count of registered users

```
# Remove irrelevant columns
clean_data = data[-c(1,2,9,15,16)]
# Convert the numerical categorical variables to predictors
clean_data$season = as.factor(clean_data$season)
clean_data$yr = as.factor(clean_data$yr)
clean_data$mnth = as.factor(clean_data$mnth)
clean_data$hr = as.factor(clean_data$hr)
clean_data$holiday = as.factor(clean_data$holiday)
clean_data$weekday = as.factor(clean_data$weekday)
clean_data$weathersit = as.factor(clean_data$weathersit)
```

Next set to training nad testing

```
# clean_data has categorical variables converted to factors , not necessary for doing separately on train and test
sample_size = floor(0.8*nrow(clean_data))
picked = sample(seq_len(nrow(clean_data)), size=sample_size)
train = clean_data[picked,]
test = clean_data[-picked,]
```

Poisson Regression Analysis

We do a poisson regression to the new clean set. We do poisson becasue the constant variance assumptin is violated when using MLR

```
model1 = glm(cnt~., data=clean_data, family = "poisson")
summary(model1)
```

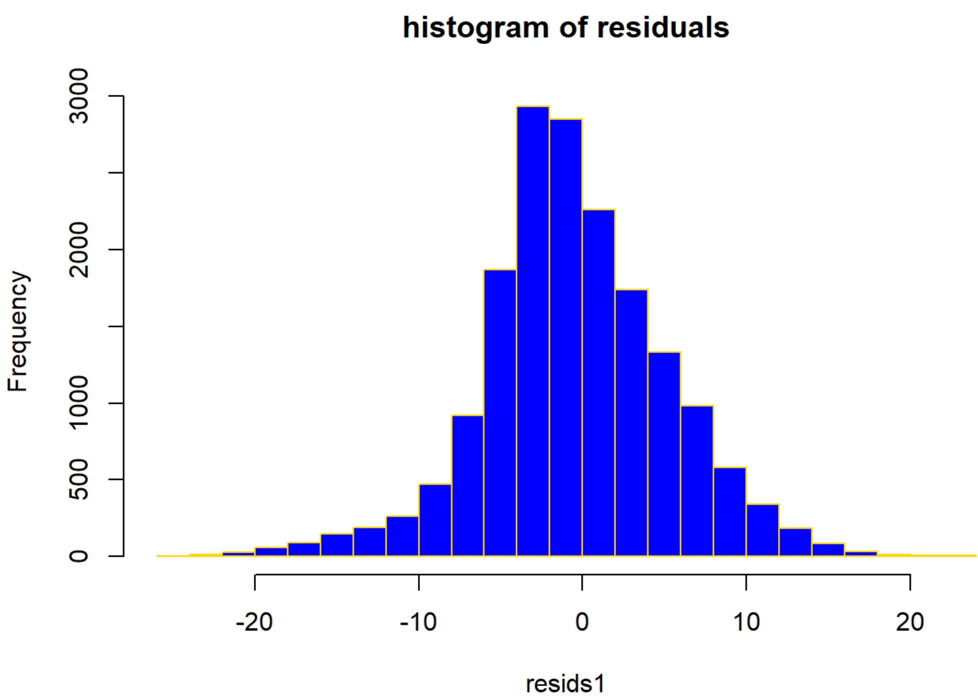
```
##
## Call:
## glm(formula = cnt ~ ., family = "poisson", data = clean_data)
##
## Coefficients:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept)  2.917366   0.006790  429.679   <2e-16 ***
## season2      0.274074   0.003680   74.486   <2e-16 ***
## season3      0.267229   0.004211   63.457   <2e-16 ***
## season4      0.457991   0.004081  112.212   <2e-16 ***
## yr1          0.468568   0.001151  407.084   <2e-16 ***
## mnth2        0.113477   0.003775   30.063   <2e-16 ***
## mnth3        0.223629   0.003935   56.827   <2e-16 ***
## mnth4        0.181256   0.005234   34.628   <2e-16 ***
## mnth5        0.244622   0.005476   44.669   <2e-16 ***
## mnth6        0.196331   0.005584   35.157   <2e-16 ***
## mnth7        0.098776   0.006063   16.291   <2e-16 ***
## mnth8        0.195068   0.005898   33.076   <2e-16 ***
## mnth9        0.270833   0.005426   49.916   <2e-16 ***
## mnth10       0.187673   0.005394   34.794   <2e-16 ***
## mnth11       0.061080   0.005302   11.519   <2e-16 ***
## mnth12       0.045320   0.004675    9.694   <2e-16 ***
## hr1         -0.466686   0.008182  -57.037   <2e-16 ***
## hr2         -0.839682   0.009313  -90.161   <2e-16 ***
## hr3         -1.507858   0.012163 -123.968   <2e-16 ***
## hr4         -2.110449   0.015858 -133.084   <2e-16 ***
## hr5         -0.956563   0.009787  -97.738   <2e-16 ***
## hr6          0.400500   0.006619   60.509   <2e-16 ***
## hr7          1.422873   0.005666  251.117   <2e-16 ***
## hr8          1.916567   0.005423  353.411   <2e-16 ***
## hr9          1.391884   0.005648  246.430   <2e-16 ***
## hr10         1.123196   0.005806  193.439   <2e-16 ***
## hr11         1.269600   0.005717  222.072   <2e-16 ***
## hr12         1.447488   0.005642  256.546   <2e-16 ***
## hr13         1.427095   0.005663  251.992   <2e-16 ***
## hr14         1.364778   0.005707  239.122   <2e-16 ***
## hr15         1.405028   0.005693  246.796   <2e-16 ***
## hr16         1.628131   0.005592  291.130   <2e-16 ***
## hr17         2.036237   0.005445  373.973   <2e-16 ***
## hr18         1.970314   0.005442  362.032   <2e-16 ***
## hr19         1.674867   0.005518  303.541   <2e-16 ***
## hr20         1.377558   0.005648  243.883   <2e-16 ***
## hr21         1.121996   0.005800  193.434   <2e-16 ***
## hr22         0.864956   0.006005  144.039   <2e-16 ***
## hr23         0.483910   0.006419   75.382   <2e-16 ***
## holiday1    -0.160986   0.003797  -42.401   <2e-16 ***
## weekday1     0.051215   0.002167   23.632   <2e-16 ***
## weekday2     0.060927   0.002103   28.971   <2e-16 ***
## weekday3     0.066412   0.002103   31.584   <2e-16 ***
## weekday4     0.067340   0.002089   32.229   <2e-16 ***
## weekday5     0.093582   0.002089   44.798   <2e-16 ***
## weekday6     0.079610   0.002091   38.064   <2e-16 ***
## weathersit2 -0.064258   0.001422  -45.177   <2e-16 ***
## weathersit3 -0.492933   0.002863 -172.188   <2e-16 ***
## temp         0.164379   0.019469    8.443   <2e-16 ***
## atemp        0.946853   0.020326   46.584   <2e-16 ***
## hum         -0.205704   0.004129  -49.823   <2e-16 ***
## windspeed   -0.109968   0.004869  -22.583   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2891591  on 17378  degrees of freedom
## Residual deviance:  572011  on 17327  degrees of freedom
```

```
## AIC: 683016
##
## Number of Fisher Scoring iterations: 5
```

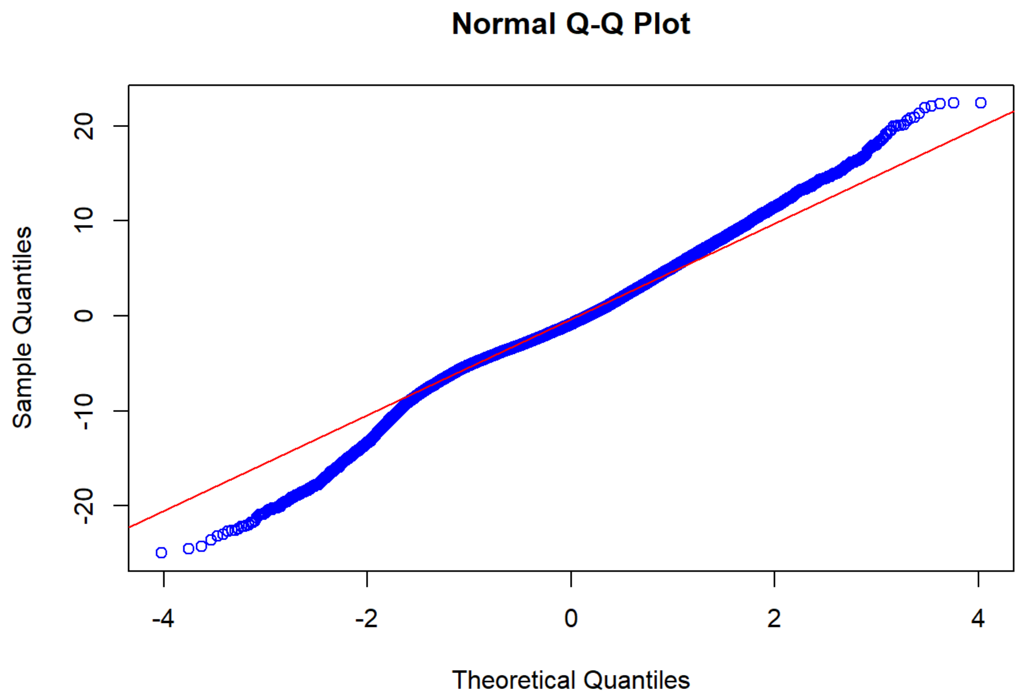
There are a lot of variables here, which can lead to inflated statistical significance.

Poisson GOF

```
resids1 <- resid(model1, type='deviance')
hist(resids1, nclass=20, col="blue", border="gold" , main = "histogram of residuals")
```




```
qqnorm(resids1, col="blue")
qqline(resids1, col = 'red')
```



The normality assumption looks ok from the histogram, but the qqnorm seems to be heavy-tailed.

```
with(model1, cbind(res.deviance = deviance, df = df.residual, p = pchisq(deviance, df.residual, lower.tail=FALSE)))
```

```
##      res.deviance    df p
## [1,]     572011.4 17327 0
```

Prediction Accuracy Measures

This is simliar to what we used for MLR in teh previous module. We use the same set of functions, and all together aggregated.

```
# Mean Squared Prediction Error (MSPE)
mse_fun <- function(pred,dat){mean((pred-dat)^2)}
# Mean Absolute Prediction Error (MAE)
mae_fun <- function(pred,dat){mean(abs(pred-dat))}
# Mean Absolute Percentage Error (MAPE)
mape_fun <- function(pred,dat){mean(abs(pred-dat)/abs(dat))}
# Precision Measure (PM)
pm_fun <- function(pred,dat){sum((pred-dat)^2)/sum((dat-mean(dat))^2)}
## Aggregate Prediction Function
pred_fun <- function(model,test){
  pred = predict(model, test, type="response")
  test.pred = pred
  mse_model = mse_fun(test.pred,test$cnt)
  mae_model = mae_fun(test.pred,test$cnt)
  mape_model = mape_fun(test.pred,test$cnt)
  pm_model = pm_fun(test.pred,test$cnt)
  pred_meas = c(mse_model,mae_model, mape_model, pm_model)
  return(pred_meas)
}
```

prediction accuracy: Poisson with Test/Train

We can measure the accuracy once or 100 times.

```
## Accuracy measures for 1 iteration (Poisson Regression)
set.seed(0)
sample_size = floor(0.8*nrow(clean_data))
picked = sample(seq_len(nrow(clean_data)), size=sample_size)
train = clean_data[picked,];
test = clean_data[-picked,]
model1.train = glm(cnt~.,data=train,family="poisson")
pred_fun(model1.train,test)
```

```
## [1] 8045.0726478   59.7703829    0.8475516    0.2449583
```

These four are MSPE, MAE, MAPE, and PM. PM is t0.245, meaning about 25% of model is explained.

```
## Accuracy measures for 100 iteration (Poisson Regression)
set.seed(0)
pred1_meas = matrix(0,4,100)
for(i in 1:100){
  sample_size = floor(0.8*nrow(clean_data))
  picked = sample(seq_len(nrow(clean_data)), size=sample_size)
  train = clean_data[picked,]; test = clean_data[-picked,]
  model1.train = glm(cnt~.,data=train,family="poisson")
  pred1_meas[,i] = pred_fun(model1.train,test)
}
model1_ave = round(apply(pred1_meas,1,mean),4)
model1_ave
```

```
## [1] 8147.6160   60.3805    0.8198    0.2477
```

For 100x, it doesn’t seem to be all that much better

P-value and inflated significance, subsampling

instead of all data, we do 20% with 100 repetitions and apply the poisson regression **Tunning Parameter**: percent sub-sample

```
## Approach: Subsample 20% of the initial data sample & repeat 100 times
count = 1
n = nrow(clean_data)
B = 100 #repetitions
ncoef = dim(summary(model1)$coeff)[1] #no of coefficients
pv_matrix = matrix(0,nrow = ncoef,ncol = B)
while (count <= B){
  subsample = sample(n, floor(n*0.2), replace=FALSE) #sample 20%
  subdata = clean_data[subsample,] #sample that from original set
  # Fit the poisson regression for each subsample
  submod = glm(cnt~.,data=subdata,family="poisson")
  ## Count pvalues smaller than 0.01 across the 100 (sub)models
  pv_matrix[,count] = summary(submod)$coeff[,4]
  count = count + 1
}
alpha = 0.01
pv_significant = rowSums(pv_matrix < alpha)
```

Identifying Statistical Significance

The above gave us 100 values and we are testing them against a sig level of95% larger than sig level.

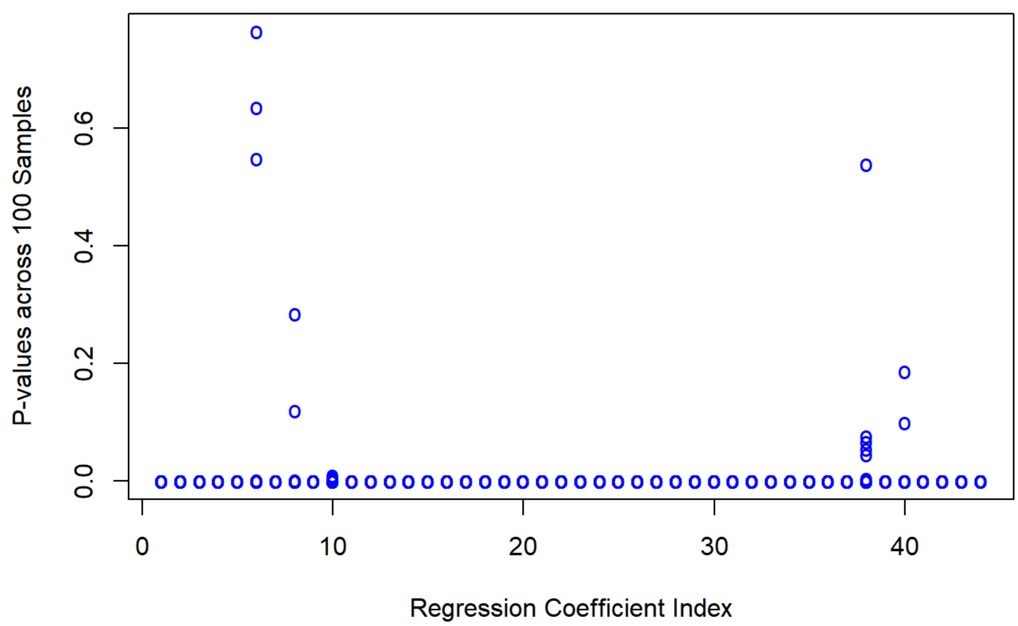
```
# Identify variables which have p-values less<alpha on more than 95% iterations
idx_scoef = which(pv_significant>=95)
length(idx_scoef)
```

```
## [1] 44
```

This means 44/51 p-values are small across the sub-samples

We can also plot this

```
matplot(pv_matrix[idx_scoef,],
        xlab="Regression Coefficient Index",
        ylab="P-values across 100 Samples",
        type="p",
        pch="o",
        col="blue")
```



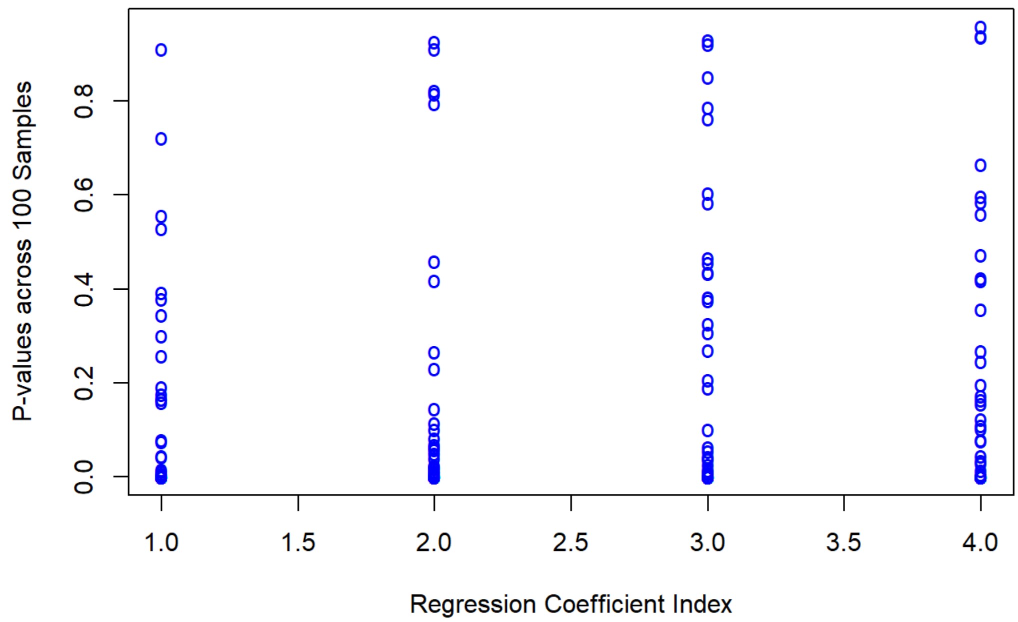
Lack Statistical Significance

We want to ind which coeficients are not statistically signficant, which would mean a pv less than 85

```
idx_icoef = which(pv_significant<85)
# Show the p-values of the non-significant coefficients in model2
cbind(summary(model1)$coeff[idx_icoef,c(1,4)],
      Freq=pv_significant[idx_icoef])
```

```
##           Estimate     Pr(>|z|) Freq
## mnth7  0.09877559 1.147263e-59   80
## mnth11 0.06107955 1.054946e-30   74
## mnth12 0.04532047 3.196587e-22   73
## temp   0.16437898 3.093212e-17   74
```

```
# Plot the 100 p-values of the non-significant coefficients
matplot(pv_matrix[idx_icoef,],
        xlab="Regression Coefficient Index",
        ylab="P-values across 100 Samples",
        type="p",
        pch="o",
        col="blue")
```



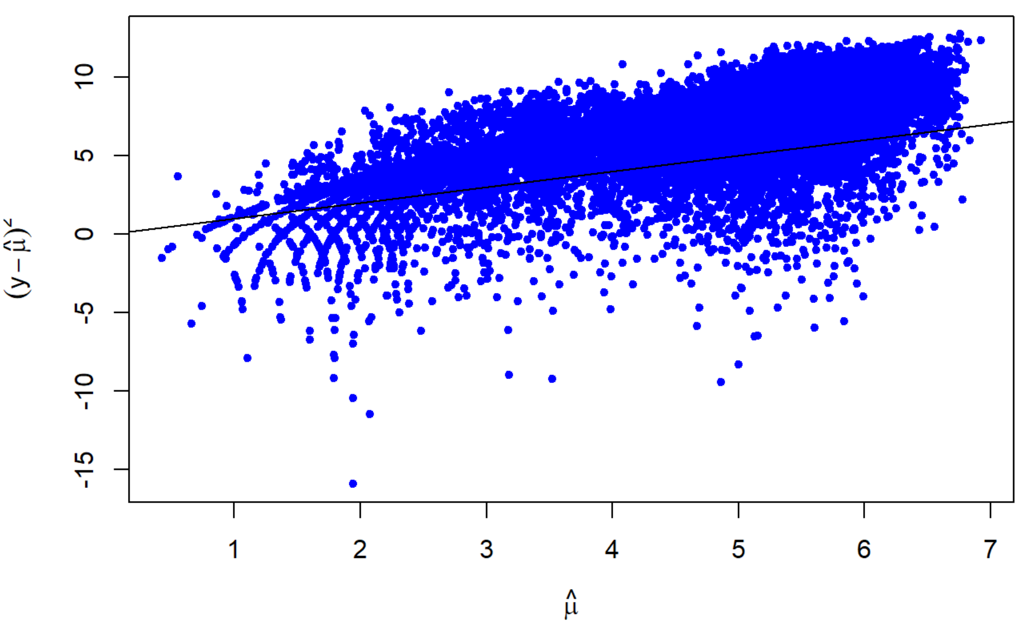
We can see from the table which

values were not sinificant because 80 of p-values smalle rthan significance the distribution of p-values is approx uniform

Overdispersion

Like the other models we can have overdispersion. We can find this by plotting the log of fitted values against log of squared differences. With Overdispersion, we can try to correct with quasi poisson or negative binomial.

```
# Overdispersion (a probable cause of inflated significance)
plot(log(fitted(model1)),log((data$cnt-fitted(model1))^2),xlab=expression(hat(mu)),ylab=expression((y-hat(mu))^2),pch=20,c
ol="blue")
abline(0,1) ## 'variance = mean' line
```



We can see that the overdispersion occurs above the line here, which means that most observations, variance is higher than the mean.

Overdispersion parameter

```
#overdispersion parameter

dp = sum(residuals(model1,type ="pearson")^2)/model1$df.residual
dp
```

```
## [1] 32.2539
```

It's above 32, which is WAY more than the threshold of 2 or 4.

```
# see how the coefficients are affected owing to overdispersion
cbind(original_estimates=summary(model1)$coeff[,4],dispersion_estimates=summary(model1,dispersion=dp)$coeff[,4])
```

```
##             original_estimates dispersion_estimates
## (Intercept)       0.000000e+00         0.000000e+00
## season2           0.000000e+00         2.683827e-39
## season3           0.000000e+00         5.495361e-29
## season4           0.000000e+00         6.817077e-87
## yr1               0.000000e+00         0.000000e+00
## mnth2            1.491440e-198         1.200495e-07
## mnth3             0.000000e+00         1.432651e-23
## mnth4            9.513063e-263         1.078637e-09
## mnth5             0.000000e+00         3.681519e-15
## mnth6            9.213358e-271         6.002722e-10
## mnth7             1.147263e-59         4.124545e-03
## mnth8            6.520489e-240         5.743655e-09
## mnth9             0.000000e+00         1.505359e-18
## mnth10           2.950093e-265         8.978114e-10
## mnth11            1.054946e-30         4.252875e-02
## mnth12            3.196587e-22         8.783683e-02
## hr1               0.000000e+00         9.859233e-24
## hr2               0.000000e+00         9.377586e-57
## hr3               0.000000e+00        1.254081e-105
## hr4               0.000000e+00        1.952626e-121
## hr5               0.000000e+00         2.247980e-66
## hr6               0.000000e+00         1.664041e-26
## hr7               0.000000e+00         0.000000e+00
## hr8               0.000000e+00         0.000000e+00
## hr9               0.000000e+00         0.000000e+00
## hr10              0.000000e+00        2.824836e-254
## hr11              0.000000e+00         0.000000e+00
## hr12              0.000000e+00         0.000000e+00
## hr13              0.000000e+00         0.000000e+00
## hr14              0.000000e+00         0.000000e+00
## hr15              0.000000e+00         0.000000e+00
## hr16              0.000000e+00         0.000000e+00
## hr17              0.000000e+00         0.000000e+00
## hr18              0.000000e+00         0.000000e+00
## hr19              0.000000e+00         0.000000e+00
## hr20              0.000000e+00         0.000000e+00
## hr21              0.000000e+00        2.911101e-254
## hr22              0.000000e+00        6.558723e-142
## hr23              0.000000e+00         3.306879e-40
## holiday1          0.000000e+00         8.275242e-14
## weekday1         1.822328e-123         3.167849e-05
## weekday2         1.514348e-184         3.374194e-07
## weekday3         6.170875e-219         2.678436e-08
## weekday4         6.955188e-228         1.388133e-08
## weekday5          0.000000e+00         3.069305e-15
## weekday6          0.000000e+00         2.052563e-11
## weathersit2       0.000000e+00         1.794705e-15
## weathersit3       0.000000e+00        6.465900e-202
## temp              3.093212e-17         1.371105e-01
## atemp             0.000000e+00         2.355479e-16
## hum               0.000000e+00         1.743287e-18
## windspeed        6.302919e-113         6.994173e-05
```

```
# effect of overdispersion on model coefficients

length(which(summary(model1,dispersion=dp)$coeff[,3]<0.025))
```

```
## [1] 10
```

```
length(which(summary(model1)$coeff[,3]<0.025))
```

```
## [1] 10
```