[Datatator2](https://www.kaggle.com/datatator2):

library(data.table)

trainData<-fread('../input/train.csv',drop = c(1,6,7),nrows=80000000)

[not use “ip”, click time, attributed time]

# Balanceamos clases

trainDataDwnSmpl<-downSample(trainData[,-5],as.factor(trainData$is\_attributed))

[down sample the training set(decease the amount of high frequency class value=> amount of lowest frequency class )]

endOfTrainData<-dim(trainDataDwnSmpl)[1]

# Descartamos las columnas "ip" (2) y "click time" (7)

testData<-fread('../input/test.csv',drop = c(2,7))

[use ip and click time as perdiction?]

# Todas las columnas de testData menos la primera ("click\_id")

z<-testData[,-1]

[Z is just about the click time]

# allData es para que no haya discrepancias en la variables binarias

# de train y test cuando hacemos one hot encode

allData<-rbind(trainDataDwnSmpl,z,fill =T)

# one hot encode app

apps<-as.factor(allData$app)

apps\_dummy<-Matrix::sparse.model.matrix(~0+apps)

# one hot encode devices

devices<-as.factor(allData$device)

devices\_dummy<-Matrix::sparse.model.matrix(~0+devices)

# one hot encode oss

oss<-as.factor(allData$os)

oss\_dummy<-Matrix::sparse.model.matrix(~0+oss)

[creat some sparse matrixs]

# one hot encode channels

channels<-as.factor(allData$channel)

channels\_dummy<-Matrix::sparse.model.matrix(~0+channels)

# Lo juntamos todo

allData\_dummified <- cbind(apps\_dummy,devices\_dummy,oss\_dummy,channels\_dummy)

# Recuperamos el conjunto de training

trainDataDwnSmpl<-xgb.DMatrix(data = allData\_dummified[1:endOfTrainData,],label = as.integer(as.character(trainDataDwnSmpl$Class)))

# Creamos el modelo

model <- xgboost(trainDataDwnSmpl, nrounds = 500, objective = "binary:logistic",eval\_metric="auc")

# Prediccion

res<-predict(model,allData\_dummified[-(1:endOfTrainData),])

submit<-cbind.data.frame(click\_id=testData$click\_id,is\_attributed=round(res,digits = 3))

write.csv(submit,"submit\_04\_abr\_2.csv",quote = F,row.names = F)

*Outcomes: down sampling ? sparse matrix? XGboost*

*Kxx:*

**\***Data.table package to save some time for data manipulation

fread();sample()

\*feature counts

\*discover the size of your dataset before look into it in detail

\*look at the most frequented value of categorical features

[most recorde is for device 1,and device 2] [two majority os takes about 66% of all the os]

 \*50% of all events are generated by 3 addresses. [ may want figure out the relation between those device and download activities(pairwise correlations)]

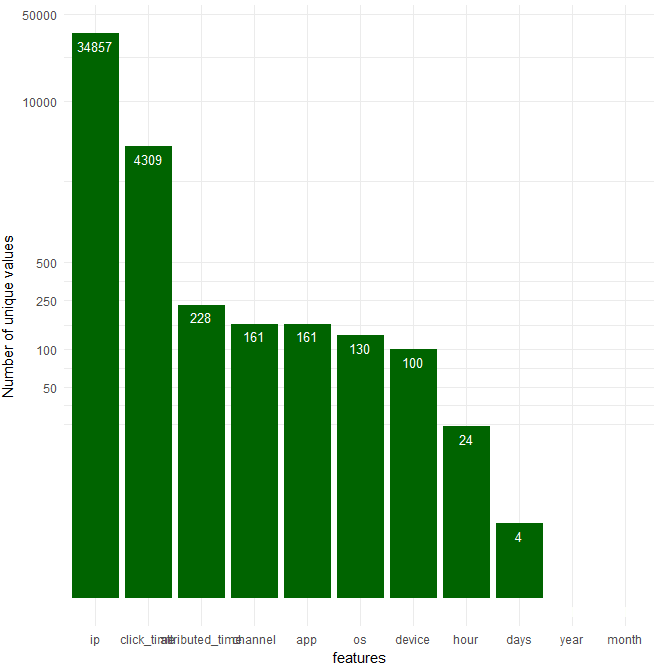
\*attributed\_time is not available in the test set

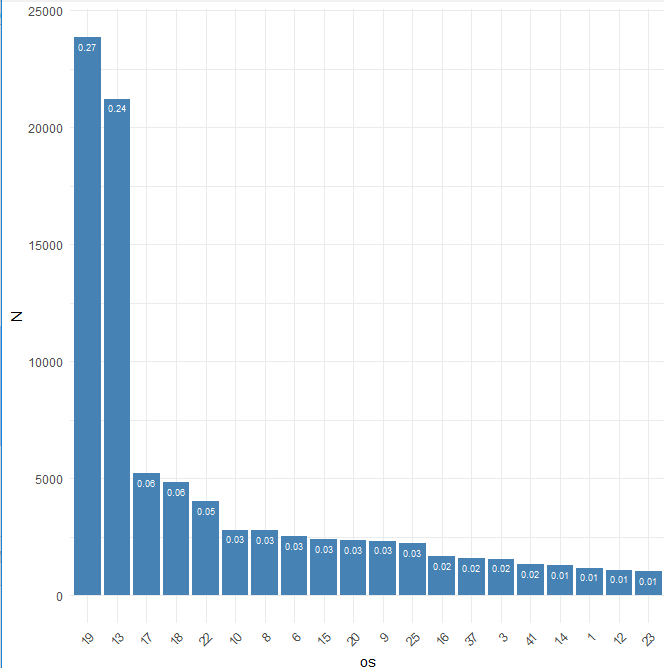
*Swamy S M*:

Convert time series data into factors:

train$year=year(train$click\_time) train$month=month(train$click\_time) train$days=weekdays(train$click\_time) [?]train$hour=hour(train$click\_time)

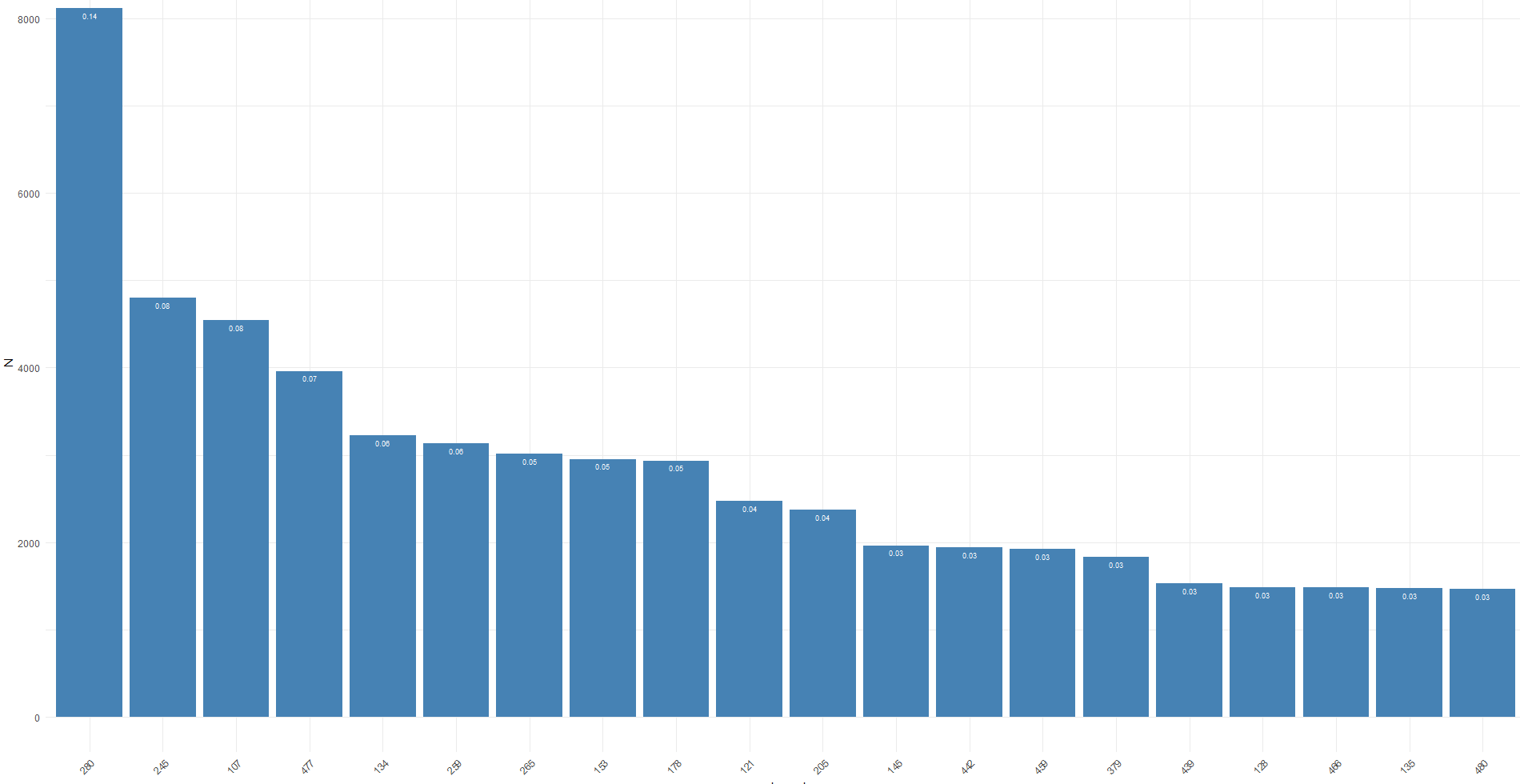
unique value for each attribute:



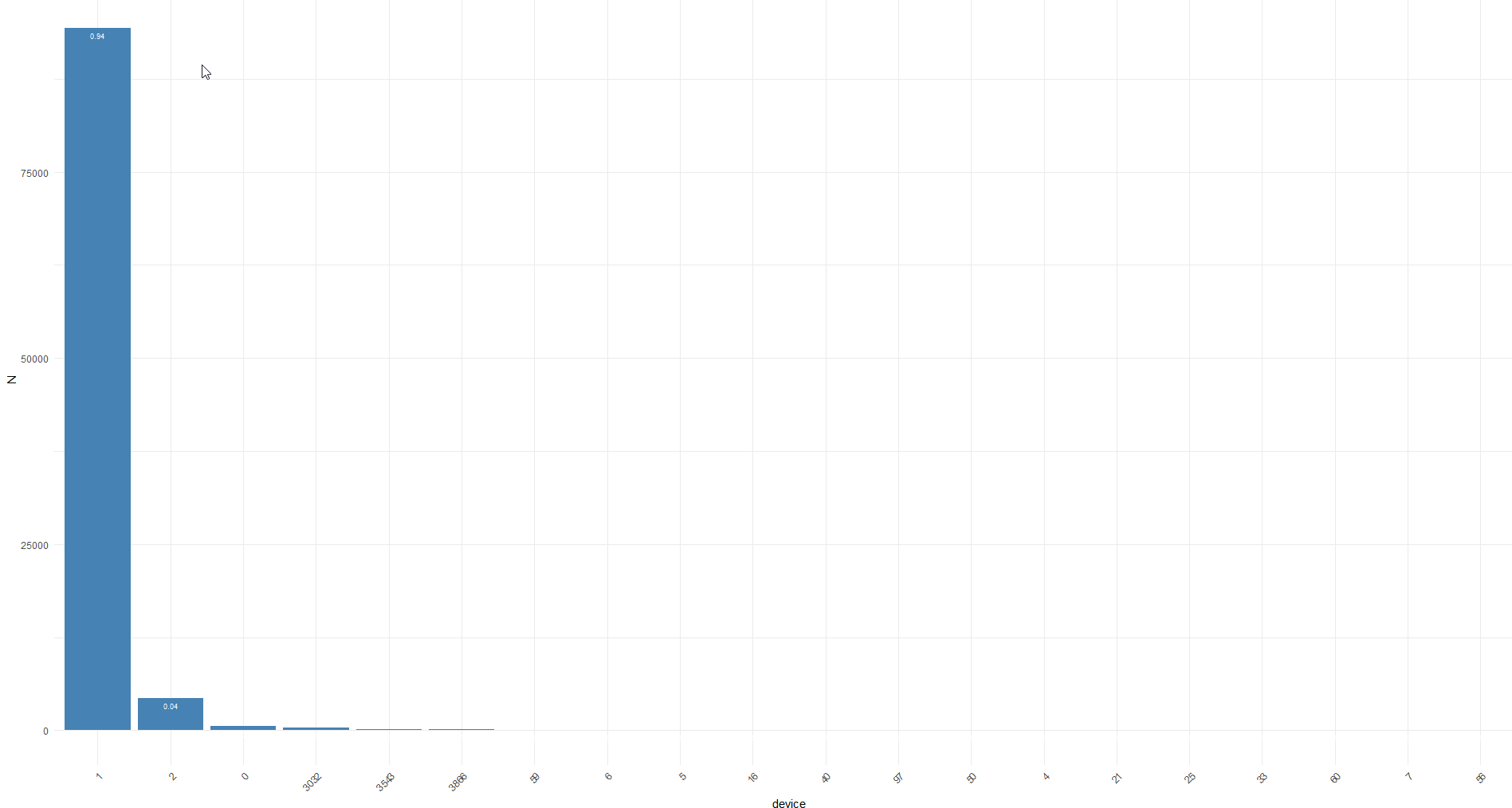
Thus the data is only for four days. 

The plot of frequency of operation system:

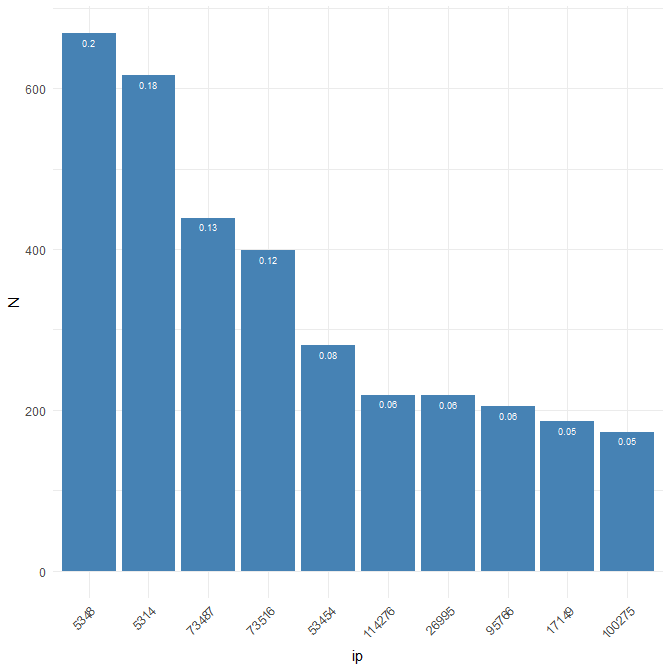
For different channel:



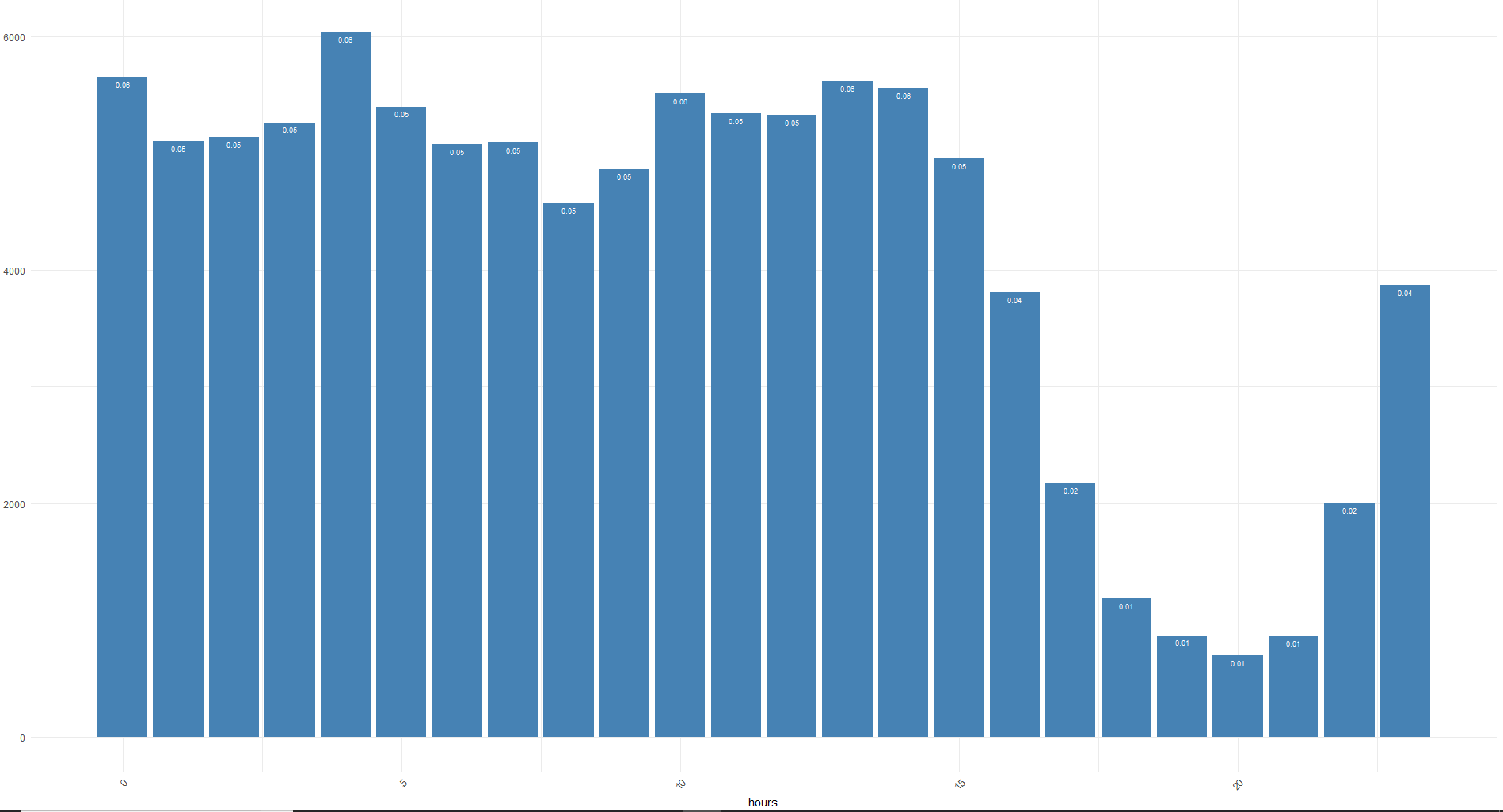
For device:



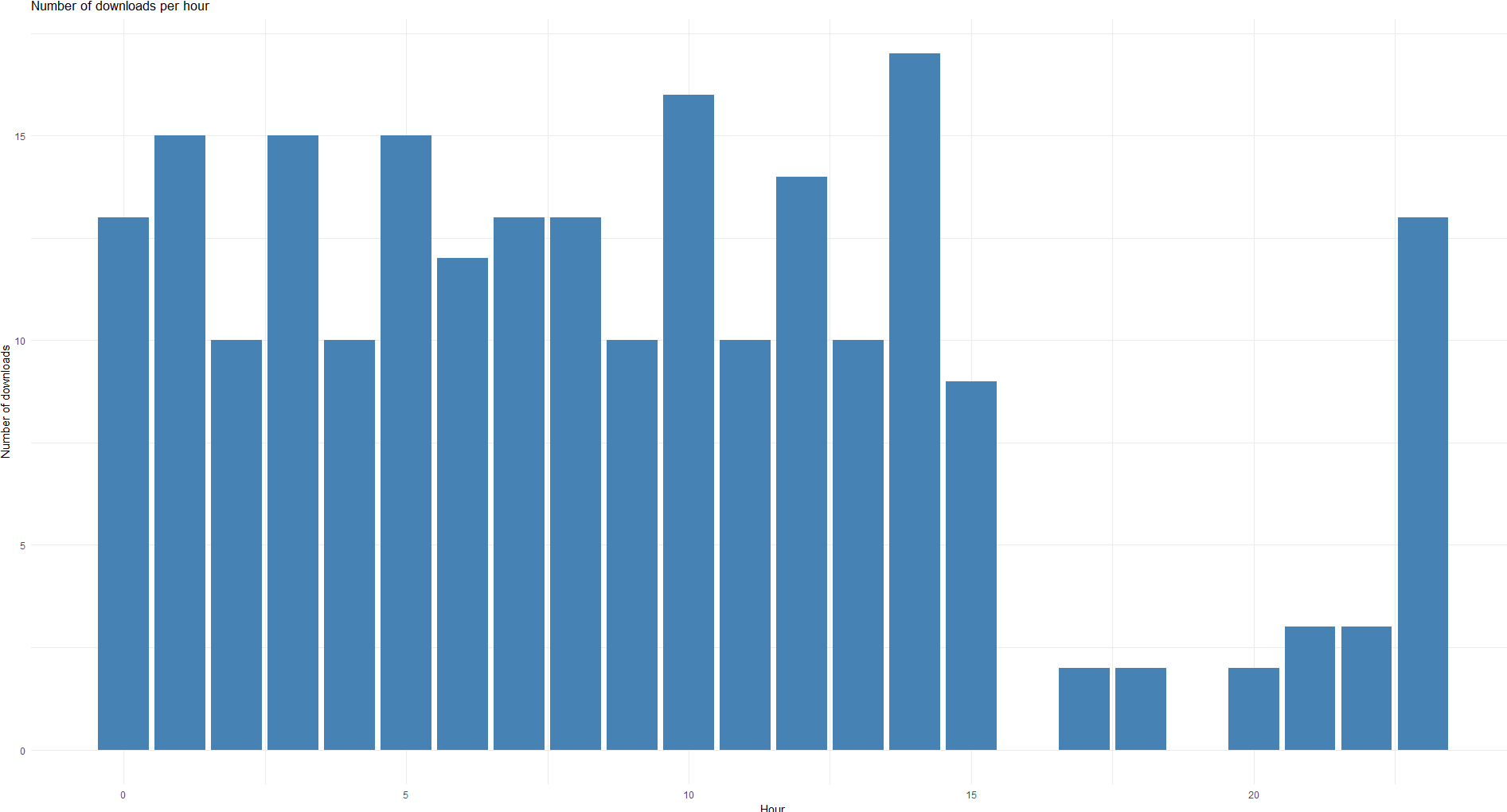
Ip vs click:



For hours:

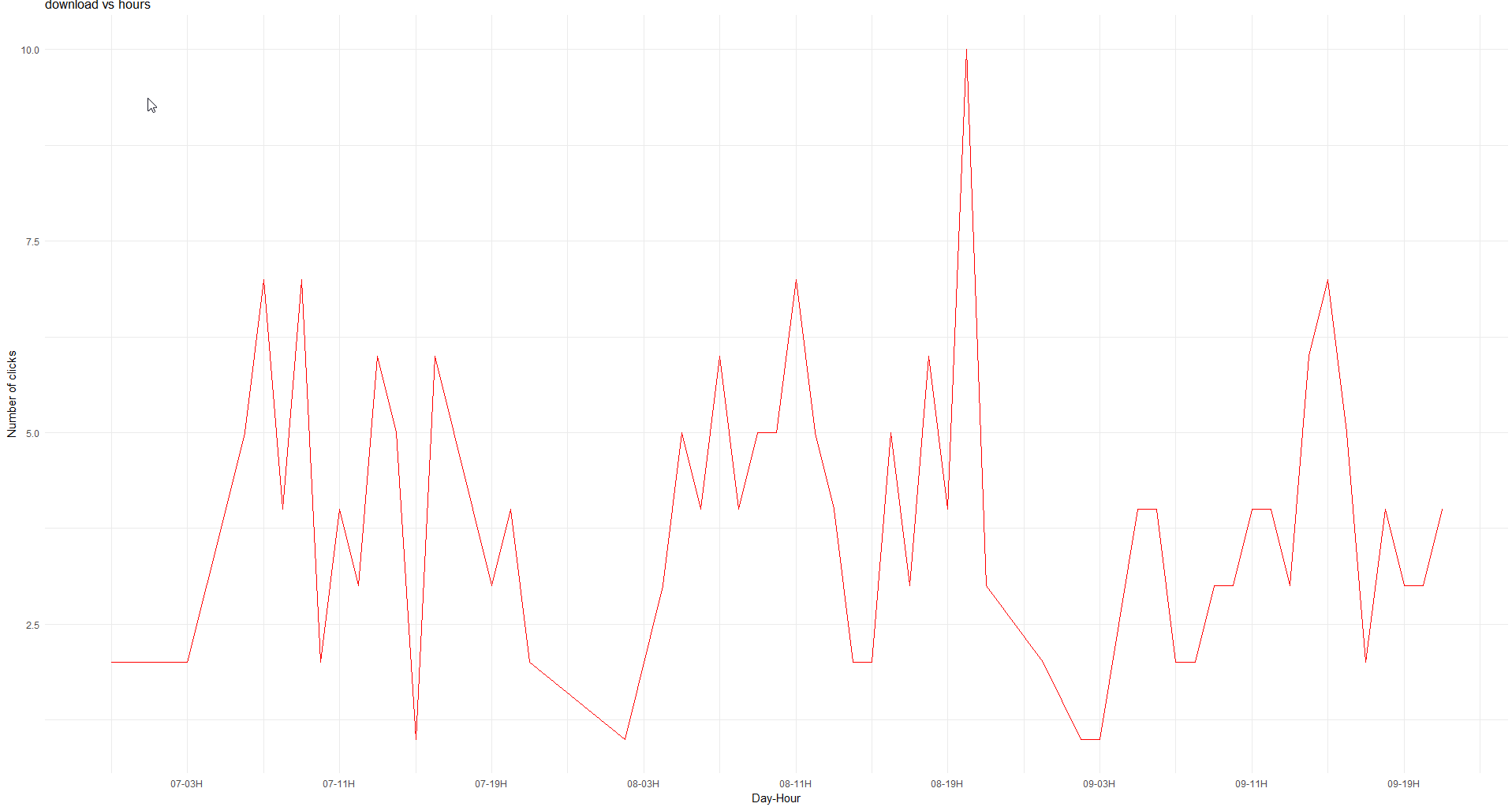


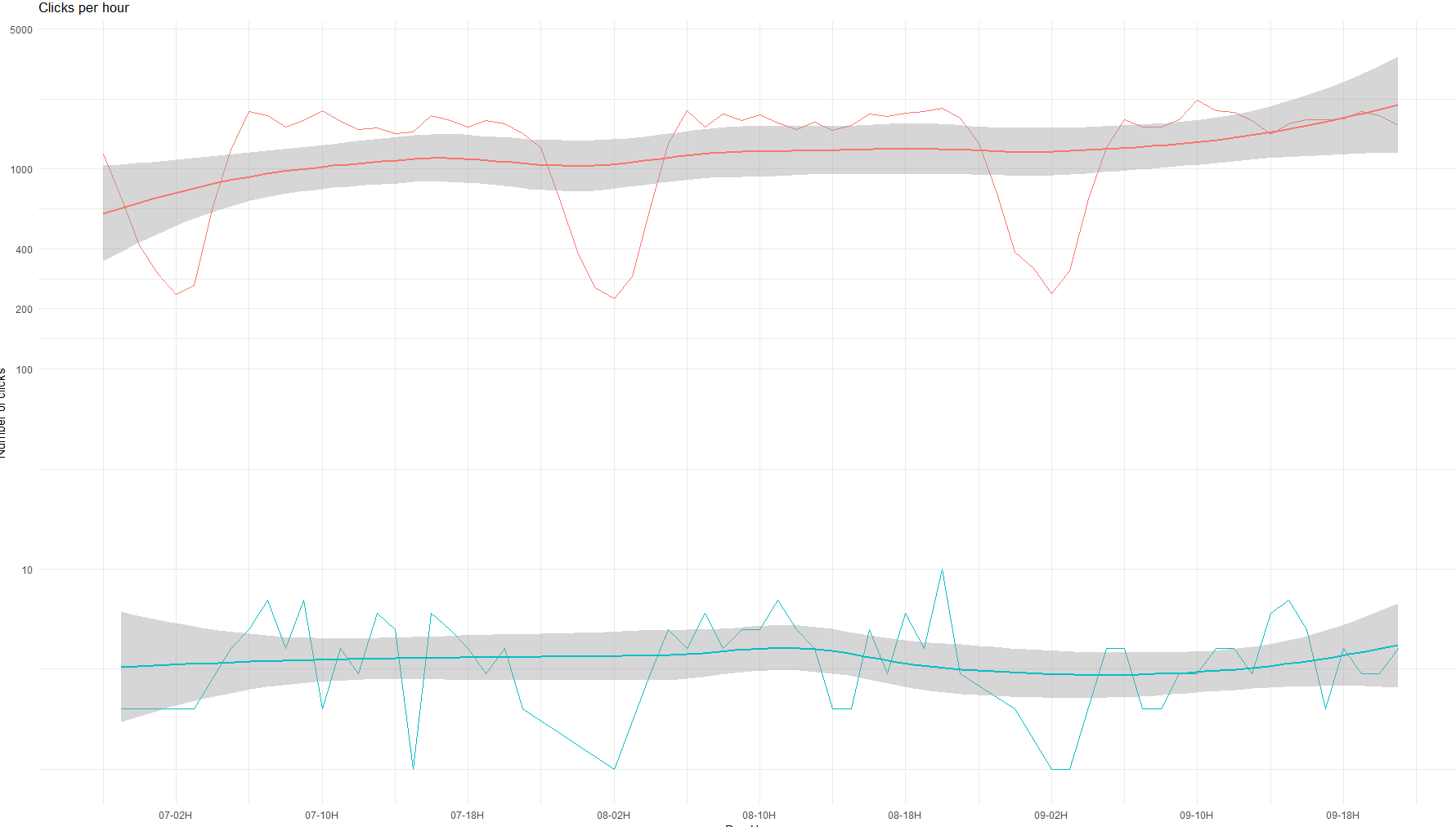
For download in hours:



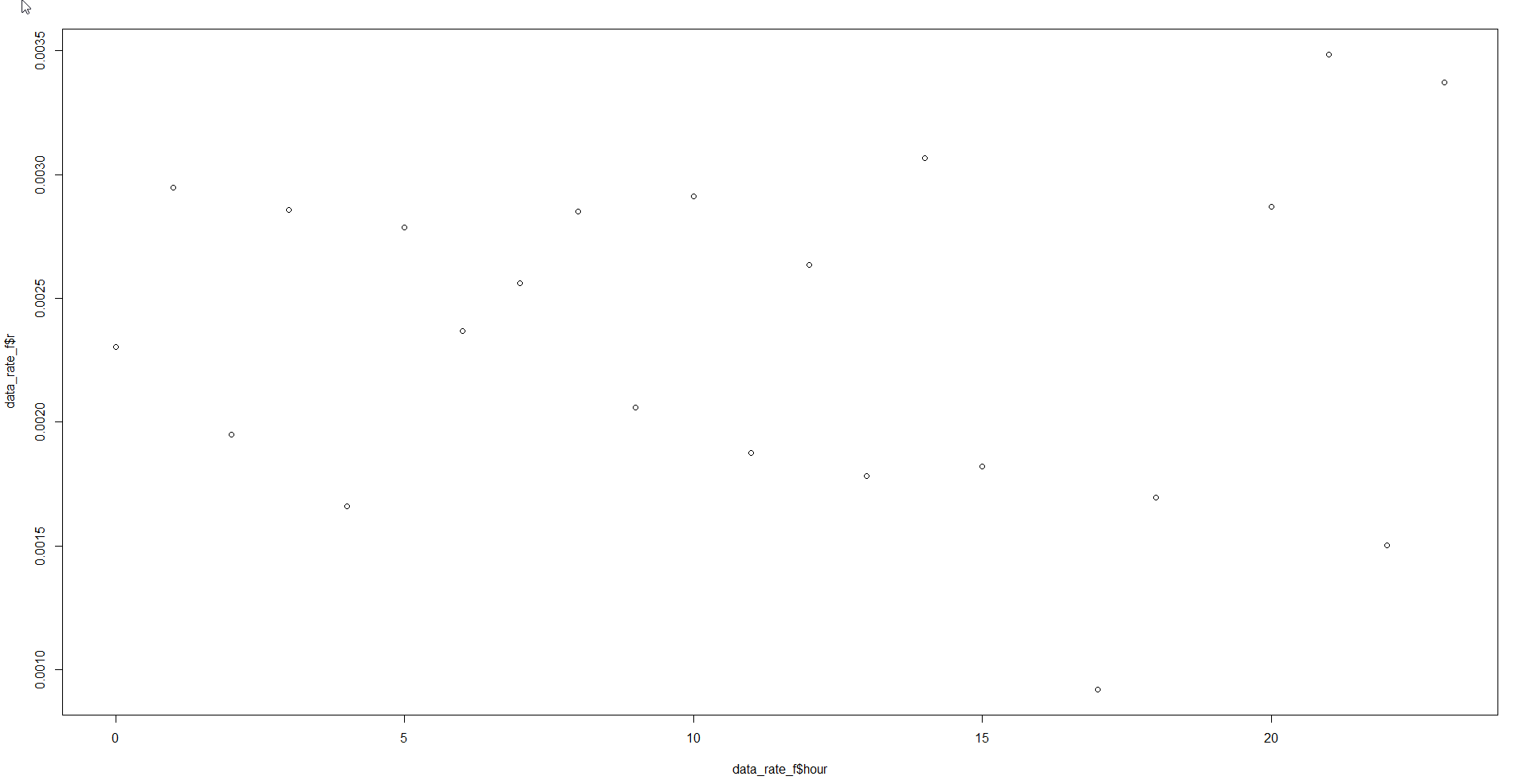
Clicks over time:  


Downloads over time:

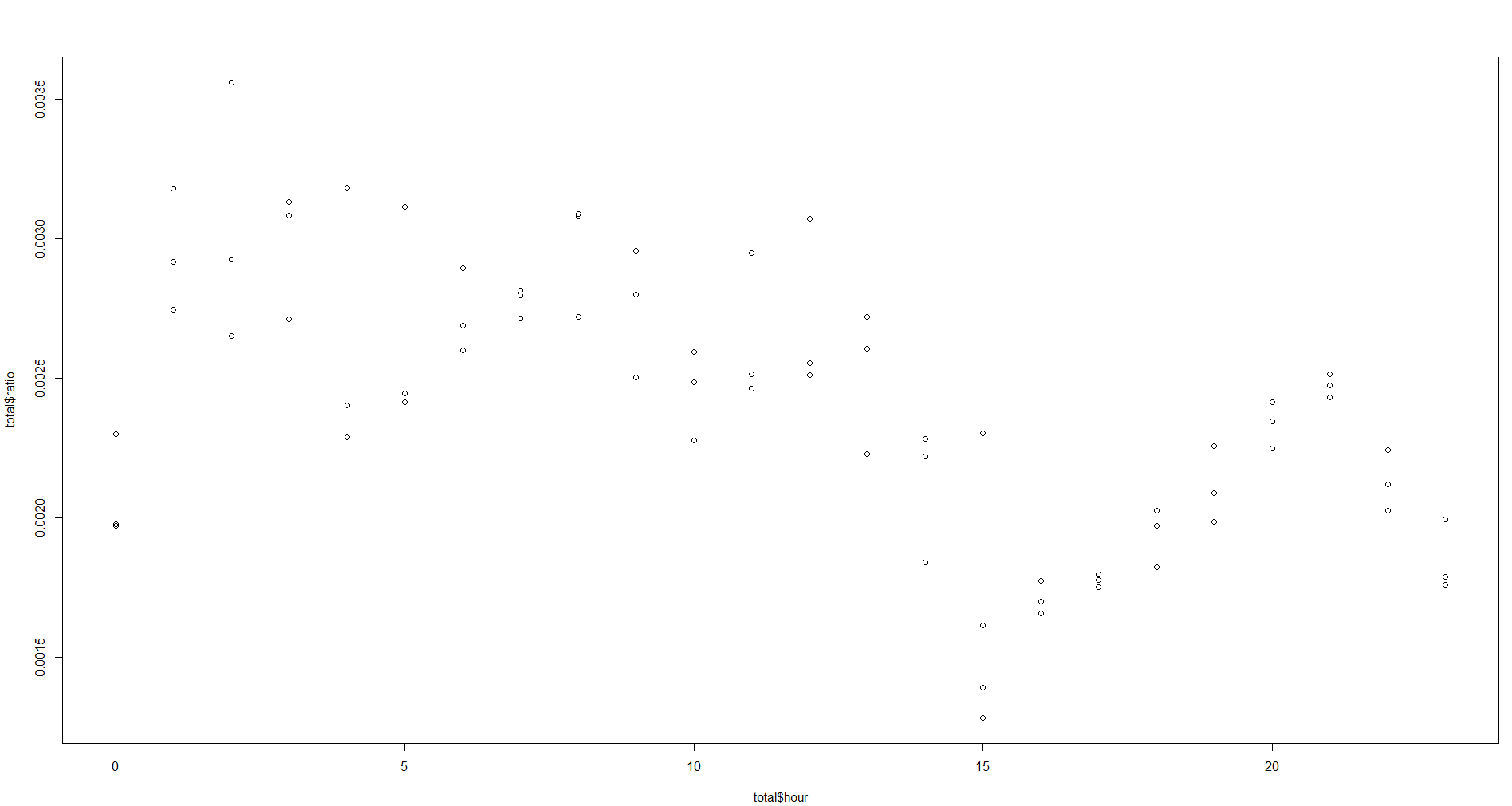




Download ratio:

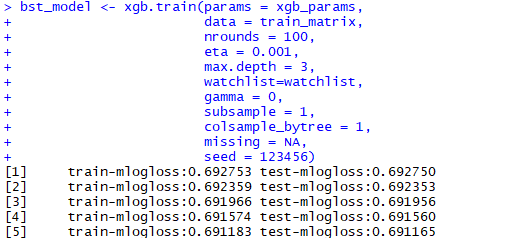


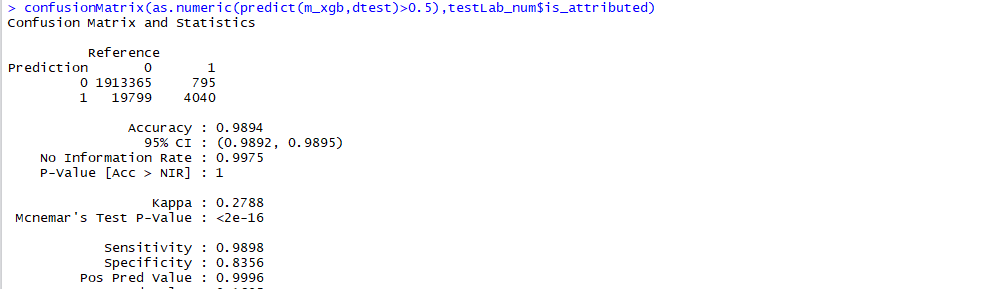
This plot is very close to random plot. Thus there is no difference as the hours changes



Thus we focus on only the third days of data.







> p2 <- list(objective = "binary:logistic",

+ booster = "gbtree",

+ eval\_metric = "auc",

+ nthread = 8,

+ eta = 0.09,

+ max\_depth = 3,

+ min\_child\_weight = 24,

+ gamma = 4,

+ subsample = 0.8,

+ lambda = 5,

+ max\_delta\_step = 6,

+ scale\_pos\_weight = 100,

+ nrounds = 1500)

>

> m2\_xgb <- xgb.train(p2, dtrain, p$nrounds,watchlist=watchlist, print\_every\_n = 20, early\_stopping\_rounds = 150)

[1] train-auc:0.802504 test-auc:0.807894

Multiple eval metrics are present. Will use test\_auc for early stopping.

Will train until test\_auc hasn't improved in 150 rounds.

[21] train-auc:0.932718 test-auc:0.934511

[41] train-auc:0.944631 test-auc:0.945273

[61] train-auc:0.950845 test-auc:0.950814

[81] train-auc:0.955282 test-auc:0.954921

[101] train-auc:0.958136 test-auc:0.957460

[121] train-auc:0.960262 test-auc:0.959157

[141] train-auc:0.961683 test-auc:0.960470

[161] train-auc:0.963009 test-auc:0.961545

[181] train-auc:0.964288 test-auc:0.962602

[200] train-auc:0.965092 test-auc:0.963200

>

> confusionMatrix(as.numeric(predict(m2\_xgb,dtest)>0.5),testLab\_num$is\_attributed)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 1910521 867

1 22643 3968

Accuracy : 0.9879

95% CI : (0.9877, 0.988)

No Information Rate : 0.9975

P-Value [Acc > NIR] : 1

Kappa : 0.2492

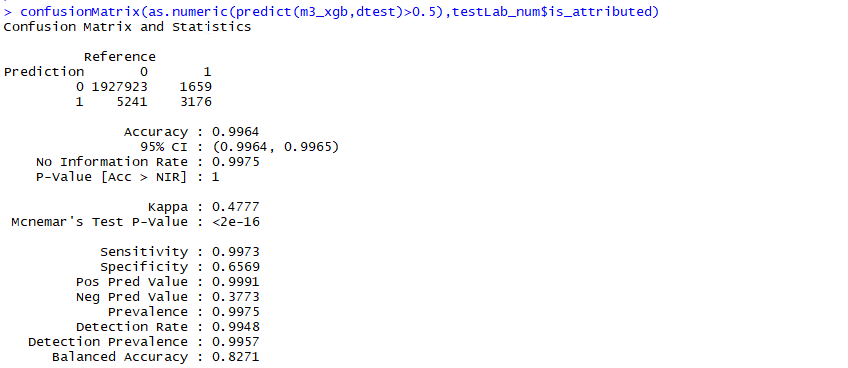
Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9883

Specificity : 0.8207

Pos Pred Value : 0.9995

When training without L2 regulizer:



> m4\_xgb <- xgb.train(p4, dtrain, p$nrounds,watchlist=watchlist, print\_every\_n = 20, early\_stopping\_rounds = 150)

[1] train-auc:0.802504 test-auc:0.807894

Multiple eval metrics are present. Will use test\_auc for early stopping.

Will train until test\_auc hasn't improved in 150 rounds.

[21] train-auc:0.856278 test-auc:0.863533

[41] train-auc:0.909589 test-auc:0.913104

[61] train-auc:0.931649 test-auc:0.933339

[81] train-auc:0.935234 test-auc:0.936748

[101] train-auc:0.940762 test-auc:0.942006

[121] train-auc:0.945629 test-auc:0.946648

[141] train-auc:0.948438 test-auc:0.949155

[161] train-auc:0.950818 test-auc:0.951138

[181] train-auc:0.951979 test-auc:0.952036

[200] train-auc:0.952868 test-auc:0.952838

>

> confusionMatrix(as.numeric(predict(m4\_xgb,dtest)>0.5),testLab\_num$is\_attributed)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 1913694 1100

1 19470 3735

Accuracy : 0.9894

95% CI : (0.9892, 0.9895)

No Information Rate : 0.9975

P-Value [Acc > NIR] : 1

Kappa : 0.2634

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.9899

Specificity : 0.7725

Pos Pred Value : 0.9994

Neg Pred Value : 0.1610

Prevalence : 0.9975

Detection Rate : 0.9875

Detection Prevalence : 0.9880

Balanced Accuracy : 0.8812

'Positive' Class : 0

##r\_time\_4=end\_time4-start\_time4

> r\_time\_4

Time difference of 5.471132 mins

##