tooplooox

Krystian Szczucki

18.09.2016

Read in the data.csv and analyse the basic statistics of the v(n) or n = 24; 72; 168.

```r
length(rowSums(!is.na(tcsv[(1:168)])))
```

```
## [1] 916
```

```r
quantile(tcsv$v168,c(0.1,0.5,0.9,0.95,0.99))
```

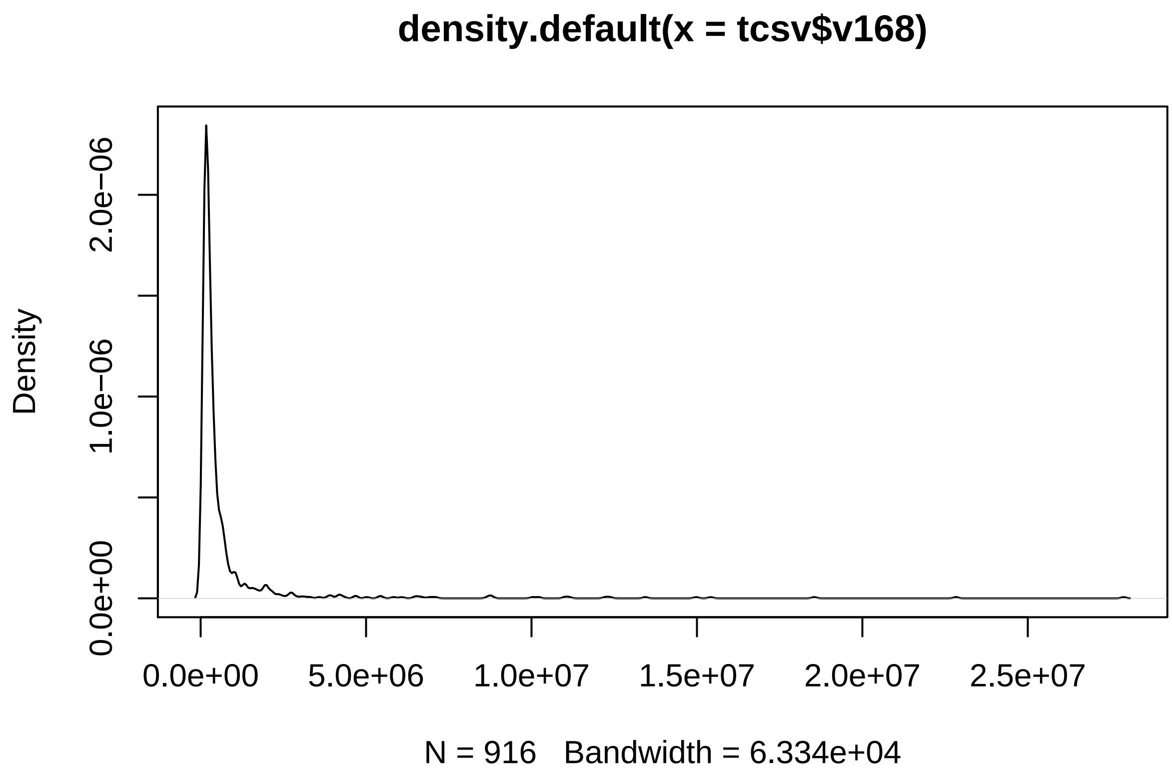
```
##      10%      50%      90%      95%      99%
##   104216   252287  1324281  2403244 10886372
```

```r
apply(df,2,summary)
```

```
##         Linear Regression Multiple-input Linear Regression reference_time
## Min.               0.7034                          0.01856           1.00
## 1st Qu.            1.7730                          0.27430           6.75
## Median             2.8110                          1.17100          12.50
## Mean               4.0380                          2.51500          12.50
## 3rd Qu.            5.1480                          2.64300          18.25
## Max.              14.5800                         14.58000          24.00
```

Plot the distribution of the v(168). How would you describe the distribution of the views?

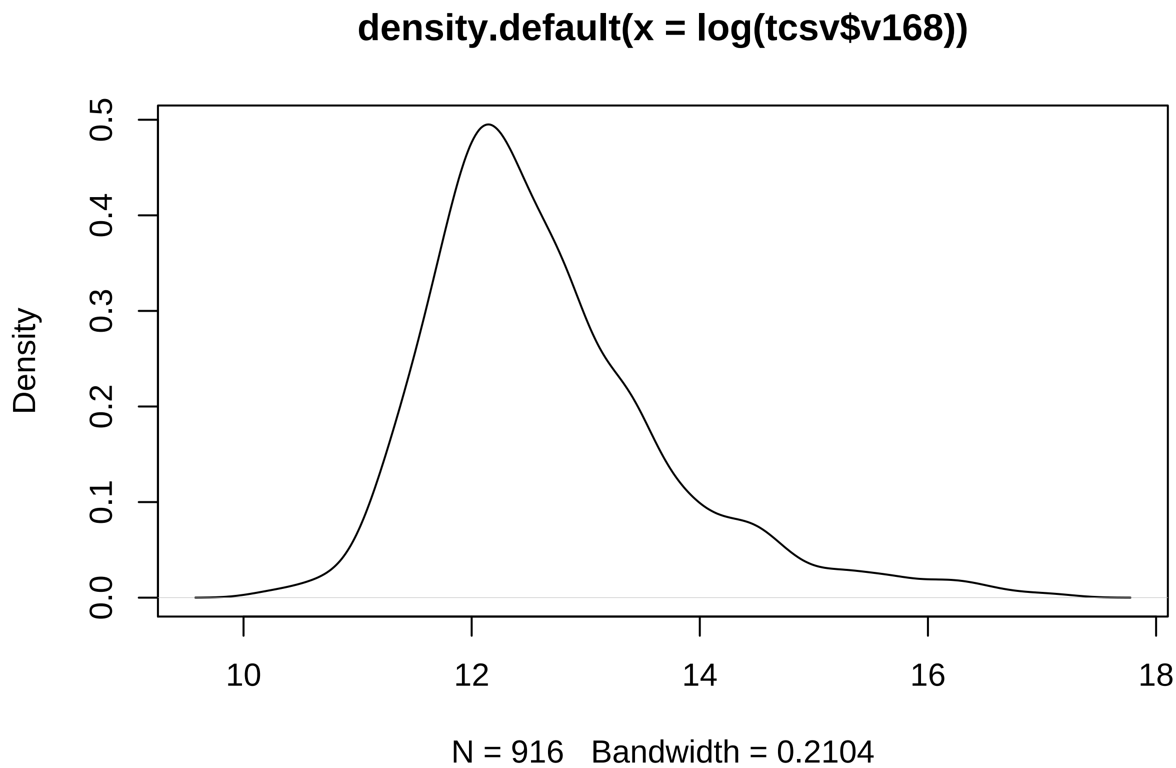
```r
plot(density(tcsv$v168))
```



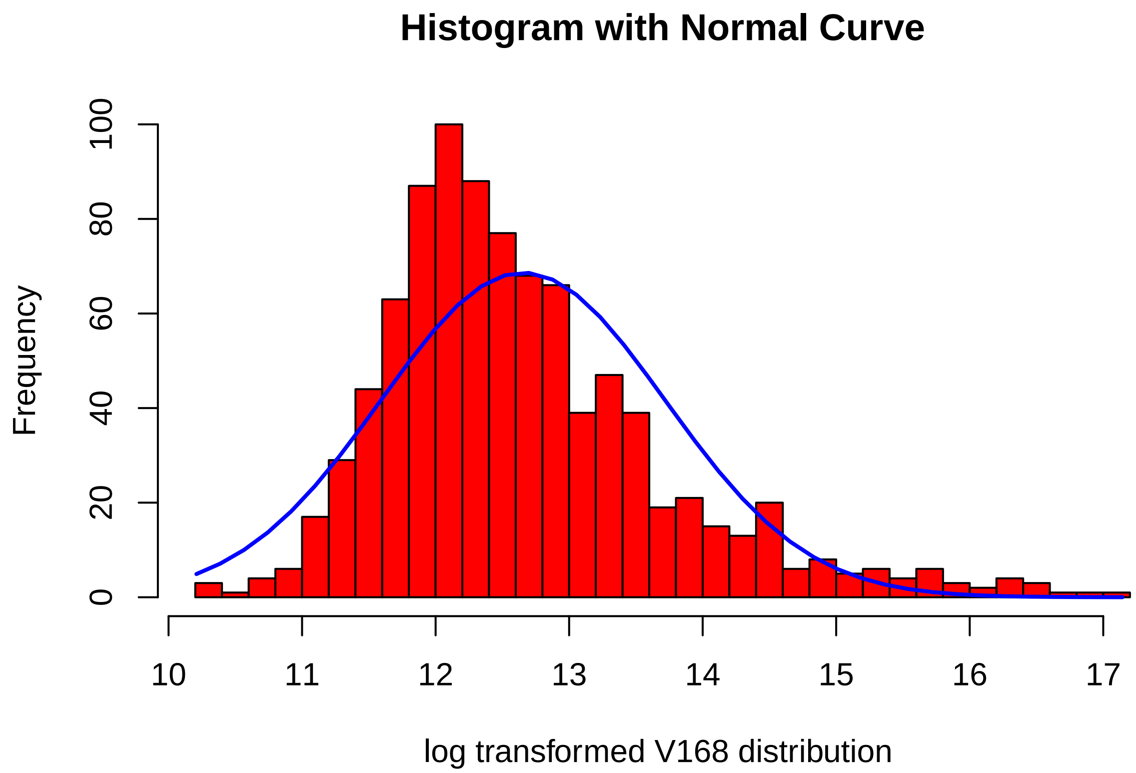
the distribution looks like right-sided skewed

Plot the distribution of the log transformed v(168). Does it ring a bell?

```r
plot(density(log(tcsv$v168)))
```




```
x <- log(tcsv$v168)
h<-hist(x, breaks=30, col="red", xlab="log transformed V168 distribution ",
        main="Histogram with Normal Curve")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```



almost perfect normal distribution - after removing outliers it should look even more bell shaped

removing outliers

```
t_mean = mean(log(tcsv$v168))
t_sd = sd(log(tcsv$v168))
new_t <- filter(tcsv,log(v168) < (t_mean + 3*sd) & log(v168) > (t_mean - 3*sd))
# 15 wartoĹ›ci odpadĹ‚o
```

Compute correlation coefficients between the log-transformed v(n) for n = 1; 2; . . . 24 and v(168).

```
#trochÄ 0 jest w v1 - moĹĽna np daÄ‡ tam Ĺ›redniÄ… z kolumny? bo inaczej nie da rady cor zrobiÄ‡ z v1
new_t[2][new_t[2] == 0] <- mean(new_t$v2)

log_n <- log(new_t[2:25])
cor(x=log_n,y=log(new_t$v168))
```

```
##          [,1]
## v1  0.6618722
## v2  0.7847666
```

```
## v3  0.8397695
## v4  0.8557074
## v5  0.8683973
## v6  0.8802344
## v7  0.8908777
## v8  0.9010290
## v9  0.9099138
## v10 0.9166807
## v11 0.9218252
## v12 0.9265142
## v13 0.9305554
## v14 0.9342969
## v15 0.9379937
## v16 0.9414161
## v17 0.9447434
## v18 0.9475137
## v19 0.9499097
## v20 0.9521537
## v21 0.9542765
## v22 0.9562581
## v23 0.9580274
## v24 0.9596839
```

```
#silna korelacja liniowa - z kaĹĽdÄ... kolejnÄ... godzinÄ... wiÄksza
```

Randomly split the log-transformed dataset into training and test sets (10% of the dataset should be used for testing, rest for training).

```
set.seed(12)
library(caret)
new_t_log <- log(new_t[2:169])
inTrain <- createDataPartition(y = new_t_log$v168,p = 0.9, list = FALSE)
training <- new_t_log[inTrain,]
testing <- new_t_log[-inTrain,]
```

Using log-transformed training data, find linear regression model that minimizes OrdinaryLeast Squares (OLS) error function. It should take as the input v(n) and output v(168).

```
u <-0
for(i in names(training[1:167]))
{
        u[i]<-summary(train(v168~training[[i]],data = training,method = "lm"))$coef[2,2]
}
```

```
sort(u,decreasing = F)[2]
```

```
##         v167
## 5.027955e-05
```

```r
fit1 <-lm(v168~v167,data=training)
summary(fit1)
```

```
##
## Call:
## lm(formula = v168 ~ v167, data = training)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0021838 -0.0005620 -0.0002844  0.0000727  0.0120556
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -6.777e-03  6.350e-04   -10.67   <2e-16 ***
## v167         1.001e+00  5.028e-05 19900.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.001367 on 811 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.96e+08 on 1 and 811 DF,  p-value: < 2.2e-16
```

Extend the above linear regression model with multiple inputs, that is it for a given time n the model should take an array of view counts preceding time

```r
set.seed(12345)
fit_all <- train(v168~.,data=training,method = "lm")
summary(fit_all)
fit2 <- train(v168~v72+v73+v74+v75+v85+v103+v114+v115+v108+v96+v97+v91+v82+v128+v129+v132+v166+v167,data
summary(fit2)
fit3 <- train(v168~v73+v74+v75+v166+v167,data=training,method = "lm")
```

```r
summary(fit3)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0047355 -0.0000688 -0.0000108  0.0000460  0.0066371
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0004343  0.0002286  -1.900   0.0578 .
## v73          0.1579535  0.0158479   9.967   <2e-16 ***
## v74         -0.3107470  0.0314125  -9.892   <2e-16 ***
## v75          0.1522072  0.0157859   9.642   <2e-16 ***
## v166        -0.8765328  0.0156268 -56.091   <2e-16 ***
## v167         1.8771562  0.0154800 121.263   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0004256 on 807 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 8.062e+08 on 5 and 807 DF,  p-value: < 2.2e-16
```

To evaluate the proposed predictors, compute mean Relative Squared Error (mRSE),

```r
set.seed(12345)
prediction <- predict(object = fit3,newdata = testing)
mRSE <- (sum(prediction/testing$v168 - 1)^2)/length(prediction)
mRSE
```

```
## [1] 1.183062e-11
```

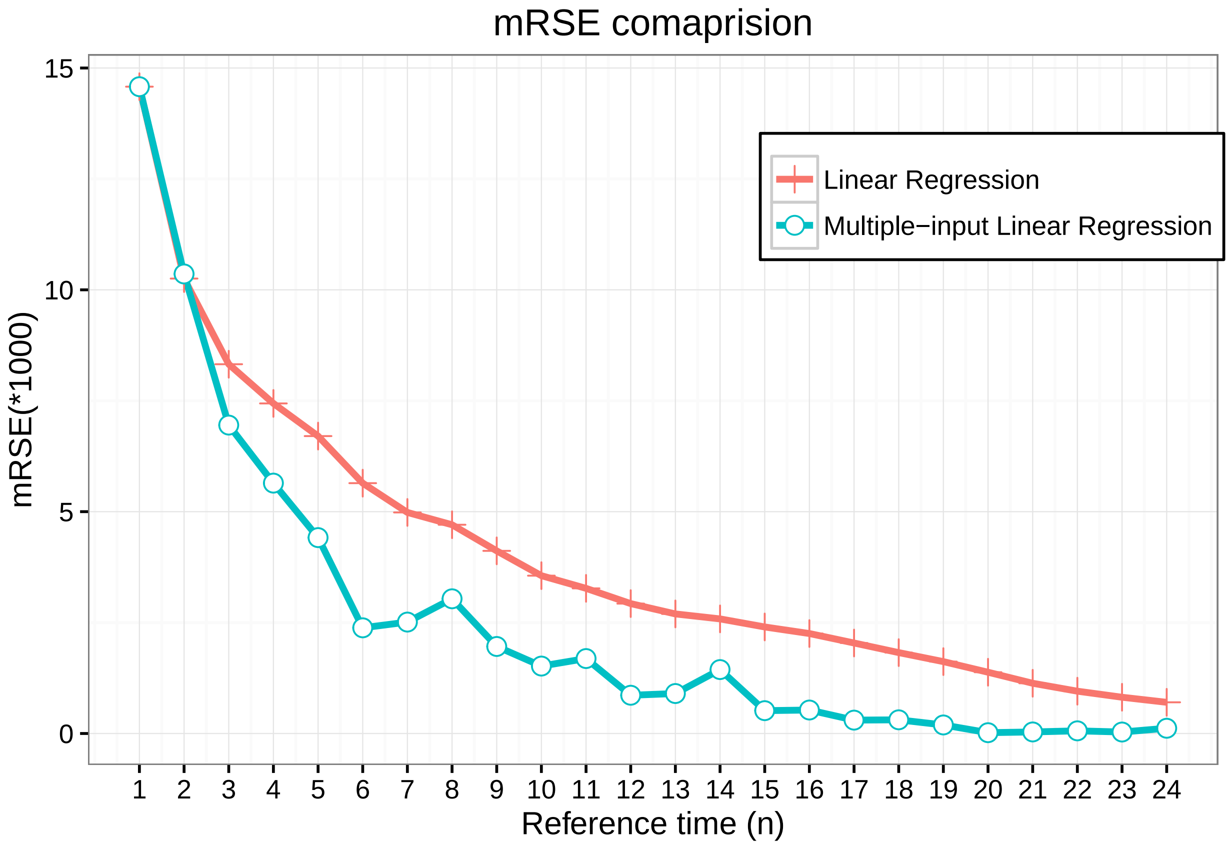
Plot the mRSE values for n in (1:24) computed on the test dataset.

```r
mRSE <- 0
for(i in 1:24)
{
        xnam <- paste("v", i, sep="")
        fmla <- as.formula(paste("v168 ~ ", xnam))
        fit <- train(fmla,data=training, method = "lm")
        prediction <- predict(object = fit,newdata = testing)
        mRSE <- c(mRSE,(sum(prediction/testing$v168 - 1)^2)/length(prediction)*1000)

}
mRSE <- mRSE[2:25]

mRSE2 <- 0
for(i in 1:24){
        xnam <- paste("v", 1:i, sep="")
        fmla <- as.formula(paste("v168 ~ ", paste(xnam, collapse= "+")))
        fit <- train(fmla, data = training,method="lm")
        prediction <- predict(object = fit,newdata = testing)
        mRSE2 <- c(mRSE2,(sum(prediction/testing$v168 - 1)^2)/length(prediction)*1000)
}
mRSE2 <- mRSE2[2:25]
```

plot



mRSE comaprision
Linear Regression
Multiple−input Linear Regression
15
10
5
0
mRSE(*1000)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Reference time (n)