STOR 664 HW 2

Brian N. White

9/9/2020

Problem 20

```
library(data.table)
dmark <- fread('http://rls.sites.oasis.unc.edu/faculty/rs/source/Data/dmark.dat')
#pull in data from website
head(dmark)
```

```
##    V1    V2    V3
## 1:  1 1.153 1.529
## 2:  2 1.152 1.530
## 3:  3 1.161 1.490
## 4:  4 1.161 1.483
## 5:  5 1.159 1.457
## 6:  6 1.154 1.460
```

```
colnames(dmark)[] <- c("week", "mark", "pound")
#give variables descriptive names
colnames(dmark)
```

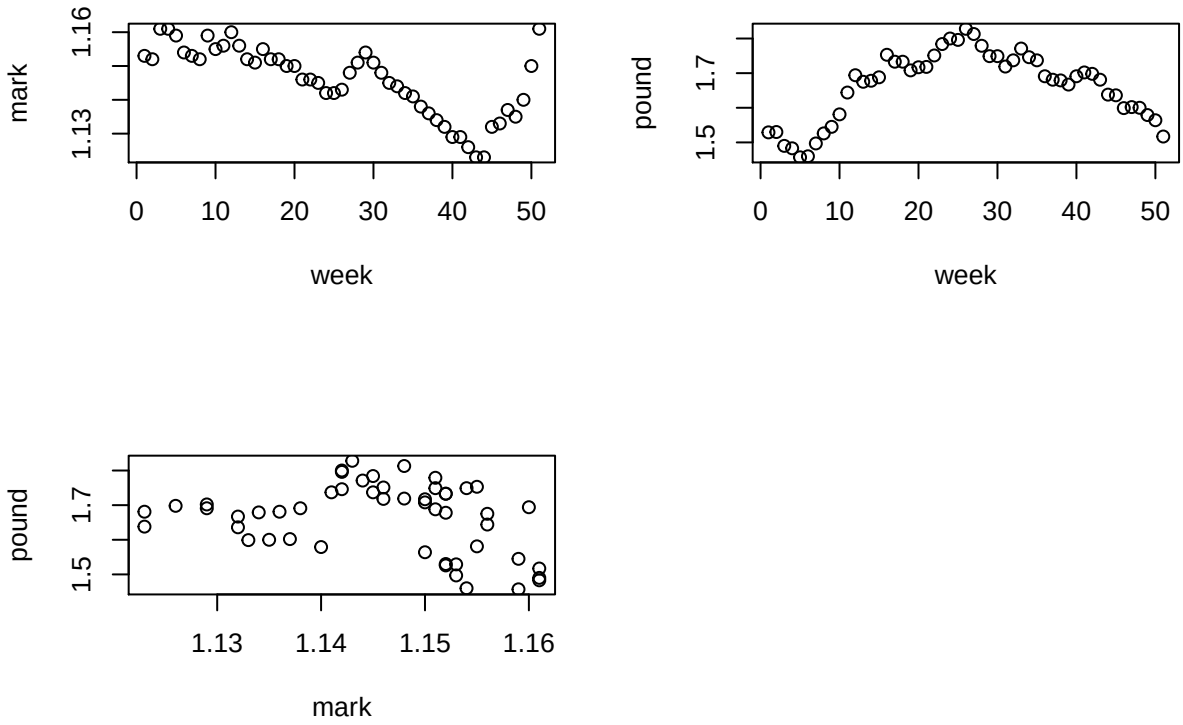
```
## [1] "week"  "mark"  "pound"
```

```
attach(dmark)
#column names of mydat recognized independently
```

(a)

Note, there is strong visual evidence of autocorrelation in the time series.

```
par(mfrow=c(2, 2))
plot(mark~week)
plot(pound~week)
plot(pound~mark)
```



(b)

Assume the individaul weekly observaions are independent. The linear regression equation $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ is computed via the code below. A point estimate for $\hat{\beta}_1$ is approximately -2.9 with a 90% confidence interval of $[-5.12, -0.69]$. Consider the following hypotheses: $H_0 : \beta_1 = 0$ vs $H_1 : \beta_1 \neq 0$ with $\alpha = 0.1$. Observe that 0 is not an element of $[-5.12, -0.69]$, the 90% confidence interval for β_1. Thus, H_0 is rejected; there is evidence to suggest that $\beta_1 \neq 0$.

```
mp_lm <- lm(pound~mark)
summary(mp_lm)
```

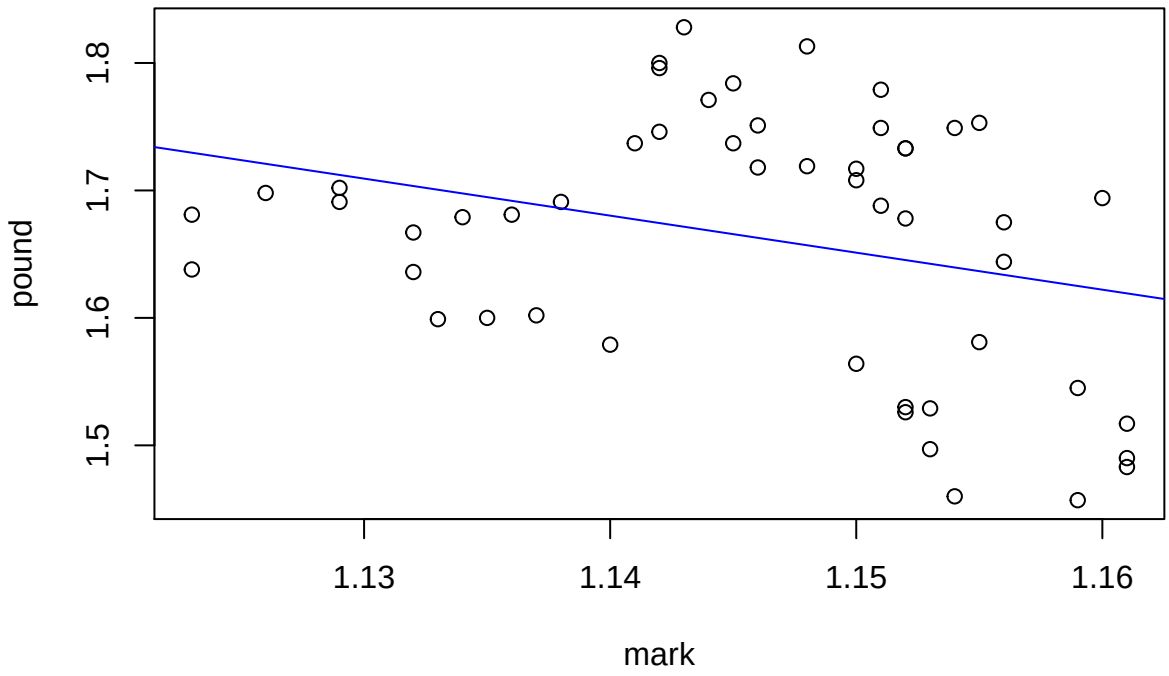
```
##
## Call:
## lm(formula = pound ~ mark)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.179567 -0.089384  0.004972  0.079741  0.156491
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.991      1.516   3.292  0.00185 **
## mark          -2.904      1.323  -2.195  0.03293 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09665 on 49 degrees of freedom
```

```
## Multiple R-squared:  0.08952,    Adjusted R-squared:  0.07094
## F-statistic: 4.818 on 1 and 49 DF,  p-value: 0.03293
```

```
confint(mp_lm, level=0.90)
```

```
##                   5 %       95 %
## (Intercept)  2.449006  7.5321114
## mark        -5.121743 -0.6858684
```

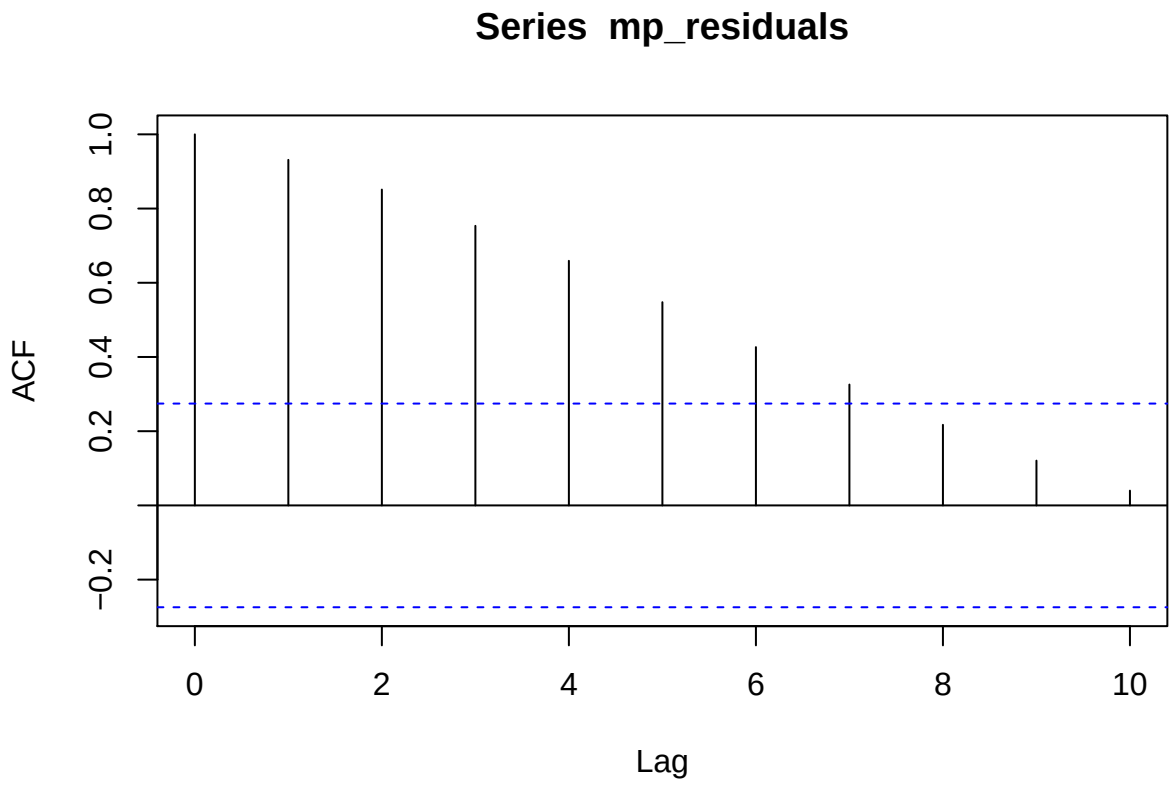
```
plot(pound~mark)
abline(mp_lm, col="blue")
```



(c)

Inspection of the first 10 autocorrelation coeffecients are computed from the residuals via the acf(.) command. Inspection of these, together with the heuristic $\frac{2}{\sqrt{n}} = \frac{2}{\sqrt{51}} \approx 0.28$, suggests that the first 8 autocorrelations are statistically significant. There is clearly evidence that autocorrelation is present. A more precise test, such as the Durbin-Watson could be performed to confirm this heuristic argument.

```
#the residuals of the linear model in question
mp_residuals <- mp_lm$residuals
#the first 10 serial correlations are computed with
#approximate 95% error bounds if the true time series is independent.
mp_ac <- acf(mp_residuals, lag.max=10)
```




```
mp_ac
```

```
##
## Autocorrelations of series 'mp_residuals', by lag
##
##     0     1     2     3     4     5     6     7     8     9    10
## 1.000 0.931 0.851 0.753 0.659 0.548 0.427 0.326 0.217 0.121 0.040
```

```
#the heursitic used to determine statistical significance of the autocorrelations
heuristic<-2/sqrt(51)
#the indices of the autocorrelations that are statistically significant.
which(abs(mp_ac$acf)>heuristic)
```

```
## [1] 1 2 3 4 5 6 7 8
```

```
#Code to compute the Durbin Watson test statistic
dw_num <- rep(0,50)
for(i in 2:51){
  dw_num[i] <- (mp_residuals[i]-mp_residuals[i-1])^2
}

D=sum(dw_num)/sum(mp_residuals^2)
D
```

```
## [1] 0.08659338
```

(d) As autocorrelation is present, the standard deviation of the least squared estimates must be corrected. In particular, the corrected standard deviation of $\hat{\beta}_1$ is computed below for K=8. Note, this corrected value is about 3.77, in contrast to the original value of 1.32. With this new value, observe that the test statistic for the previously considerd hypotheses is $t = -\frac{2.195}{3.77} \approx -0.771$. The p-value associated with this test statistic, with respect to a t_{n-2} distribution, is greater than the 0.1 threshold. Thus, in contrast to the previous conclusion, the evidence does not support the rejection of H_0. In otherwords, there is not statistically significant evidence to suggest that $\beta_1 \neq 0$.

```
rse <- sum((mp_lm$residuals)^2)/49
beta1_se=sqrt(rse/sum((mark-mean(mark))^2))
beta1_se
```

```
## [1] 1.322917
```

```
#the denominator of r_x(k) for all k=1,..,8
k_denom <- vector()
  for(i in 1:51){
    k_denom[i] <- (mark[i]-mean(mark))^2
  }

#dummy vector to be used in the for loop below
rxk_terms <-list(vector(), vector(), vector(), vector(), vector(), vector(), vector(), vector())

#the lag
K=8

#this for loop generates the vectors that are stored in rxk_terms.
#Each one contains the terms of the sum in
#the numerator of the ratio that defines r_x(k)
for(i in 1:K){
  for(j in 1:(51-i)){
    rxk_terms[[i]][j] <- (mark[j]-mean(mark))*(mark[j+i]-mean(mark))
  }
}

#dummy vector to be used in the for loop below
a <- vector()

#this for loop fills the dummy vector a with the r_x(k)
#values where k=1,...,8; where the k'th entry corresponds to r_x(k).
for(i in 1:K){
  a[i] <-as.numeric(lapply(rxk_terms, sum)[i])/sum(k_denom)
}

#create vector of terms of the sum in the second term of
#the scaling factor for the corrected variance slope estimate
b <- vector()
for(i in 1:8){
  b[i] <- (mp_ac$acf[i])*a[i]
}

#the slope estimate standard deviation corrected for
#the presence of autocorrelation in the residuals.
```

```
beta1_se_corrected <- sqrt(((beta1_se)^2)*(1+2*sum(b)))
beta1_se_corrected
```

```
## [1] 3.765597
```

```
#t statistic with new standard error
t=(mp_lm$coefficients[2])/beta1_se_corrected
t
```

```
##       mark
## -0.7711408
```

```
qt(.05, 49)
```

```
## [1] -1.676551
```

Problem 21

First, the data set is imported and the columns given descriptive names.

```
marathon <- fread('http://rls.sites.oasis.unc.edu/faculty/rs/source/Data/marathon.dat')
#pull in data from website
head(marathon)
```

```
##          V1      V2
## 1: 1.000178  9402.5
## 2: 1.000178  9404.0
## 3: 1.000178  9402.0
## 4: 1.000178  9403.0
## 5: 0.000000 12163.5
## 6: 0.000000 14981.5
```

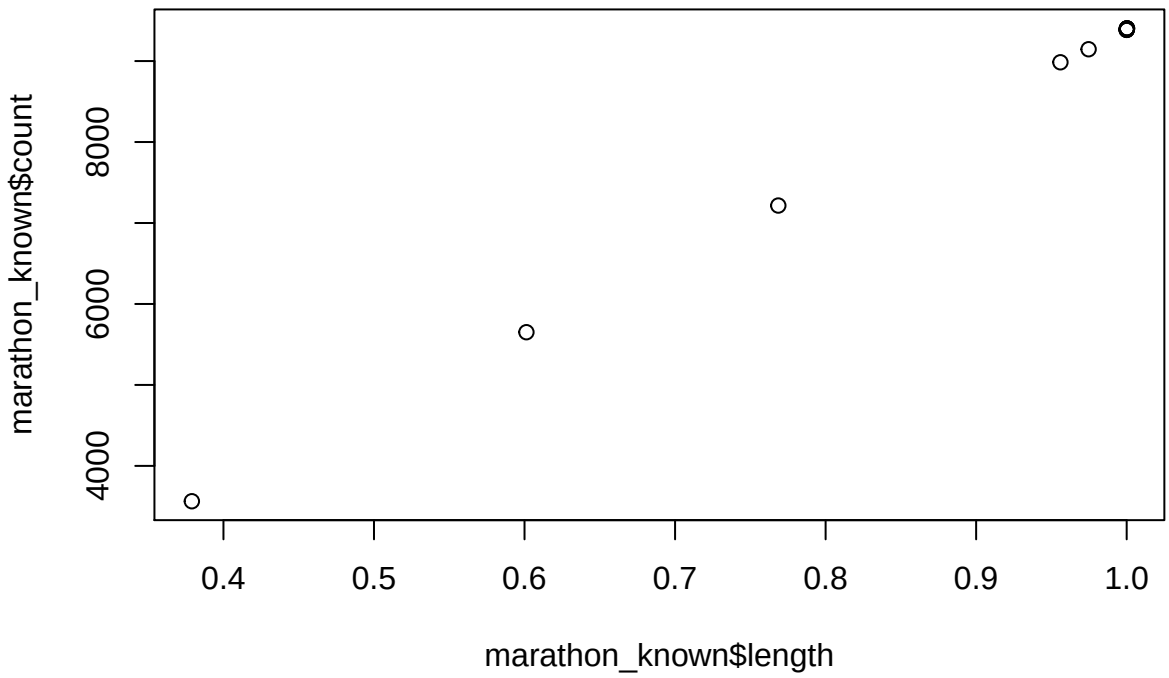
```
colnames(marathon)[] <- c("length", "count")
#give variables descriptive names
colnames(marathon)
```

```
## [1] "length" "count"
```

Next, approximate inverse regression is performed. Note, a zero in the length column corresponds to a missing value. I will use inverse regression to predict these unknown length values from the corresponding known count value. Further, I will produce standard errors for these estimates. The predicted values, along with their corresponding y values and standard errors, are compiled in the data frame named 'results_df'. This data frame is output below.

```
#remove rows with unkown x values from data
marathon_known <- marathon[-c(which(marathon$length==0)),]

#a scatterplot of the data in question
plot(marathon_known$count~marathon_known$length)
```




```
#make variable names usable without reference to data set
attach(marathon_known)

#perform linear regression on the known data
lm_marathon <- lm(count~length)

#create data vectors for new unknown x and known y values
new_y <- marathon$count[which(marathon$length==0)]
new_x <- vector()

#compute inverse regression estimates of
#the unknown length values using the corresponding known y values
for(i in 1:13){
  new_x[i] <- mean(length)+(new_y[i]-lm_marathon$coefficients[1])/lm_marathon$coefficients[2]
}

#the predicted x values
new_x
```

```
##  [1] 2.185045 2.485169 4.466438 5.127020 2.808511 3.440232 5.157214 2.922842
##  [9] 3.668254 6.194922 1.500499 1.465141 1.058460
```

```
#compute the prediction standard errors of the length estimates
summary(lm_marathon)
```

```
##
```

```
## Call:
## lm(formula = count ~ length)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.772 -7.727  1.230  7.063  8.938
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.95      10.33   0.382     0.71
## length       9389.44      11.34 828.238   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.652 on 10 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 6.86e+05 on 1 and 10 DF,  p-value: < 2.2e-16
```

```
resid_se=7.652


#compute the sum of the squared known centered length values
length_c <- vector()
for(i in 1:12){
  length_c[i] <- length[i]-mean(length)
}

d=sum(length_c^2)

#compute the standard errors of the new length estimates
se_x <- vector()
for(i in 1:13){
  se_x[i] <- (resid_se/lm_marathon$coefficients[2])*sqrt((1/12)+(((new_x[i]-mean(length))^2)/d)+1)
}

#standard errors of the predicted x values
se_x
```

```
##  [1] 0.0017788184 0.0021044290 0.0044005232 0.0051854019 0.0024667237
##  [6] 0.0031937081 0.0052213685 0.0025967770 0.0034598964 0.0064608123
## [11] 0.0011237280 0.0010962006 0.0008722732
```

```
results_df <- data.frame(new_x=new_x, y=marathon$count[which(marathon$length==0)], se_new_x=se_x)
results_df
```

```
##       new_x       y     se_new_x
## 1  2.185045 12163.5 0.0017788184
## 2  2.485169 14981.5 0.0021044290
## 3  4.466438 33584.5 0.0044005232
## 4  5.127020 39787.0 0.0051854019
## 5  2.808511 18017.5 0.0024667237
## 6  3.440232 23949.0 0.0031937081
## 7  5.157214 40070.5 0.0052213685
```

```
## 8  2.922842 19091.0 0.0025967770
## 9  3.668254 26090.0 0.0034598964
## 10 6.194922 49814.0 0.0064608123
## 11 1.500499  5736.0 0.0011237280
## 12 1.465141  5404.0 0.0010962006
## 13 1.058460  1585.5 0.0008722732
```

Next, I compute, via both approximate and exact methods, a 95% prediction interval for the total length of the predicted x values (i.e. the total length of the measurements which did not have a steel tape measurement). I will do so with the assumption that there is an independent error with variance σ^2 for each section. Further, I contend that the sum of the errors over the 13 unmeasured sections should have variance $13\sigma^2$. The formulae to be used to compute the 95% prediction interval can be found on page 80, for the approximate method, and page 85, for the exact method, of Smith and Young. To find the approximate prediction interval see the output of my_approx_conf below. For the exact prediction interval see the handwritten notes. Note, that the traditional addition of 0.1% to the length of the road course is reasonable in light of the approxaminate prediction interval.

```
x_bar <- sum(new_x)

#lower bound of prediction interval
conf_approx_up <- x_bar + pt(12, .975)*(13*resid_se/lm_marathon$coefficients[2])*sqrt(1/12
                                        + ((x_bar-mean(marathon_known$length))^2/d) + 1)
#upper bound of prediction interval
conf_approx_low <- x_bar - pt(12, .975)*(13*resid_se/lm_marathon$coefficients[2])*sqrt(1/12
                                       + ((x_bar-mean(marathon_known$length))^2/d) + 1)

my_approx_conf <- data.frame(lower=conf_approx_low, x_bar=x_bar, upper=conf_approx_up)
my_approx_conf
```

```
##           lower    x_bar    upper
## length 41.84518 42.47975 43.11431
```

```
#a 0.1% correction to the length of road course
x_bar*1.001
```

```
## [1] 42.52223
```

A last task, I will re-compute the predicted x values and their corresponding standard errors under the assupmtion that the true slope is 0 (i.e. this is reasonble considering the scatterplot of the data). Note, that the change in the values is negligable as the assumption that the slope is 0 is well founded (i.e. the least squared estimate for the slope was near 0 to begin with).

```
#slope 0 (s0)
new_x_s0 <- vector()

#compute inverse regression estimates of
#the unknown length values using the corresponding known y values where beta0 is assumed to be 0.
for(i in 1:13){
  new_x_s0[i] <- mean(length)+(new_y[i]/lm_marathon$coefficients[2])
}

new_x_s0
```

```
##  [1] 2.185466 2.485590 4.466858 5.127441 2.808932 3.440652 5.157634 2.923263
##  [9] 3.668675 6.195343 1.500920 1.465561 1.058881
```

```
#compute the standard errors of the new length estimates
se_x_s0 <- vector()
for(i in 1:13){
  se_x_s0[i] <-
    (resid_se/lm_marathon$coefficients[2])*sqrt((1/12)+(((new_x_s0[i]-mean(length))^2)/d)+1)
}


results_df_s0 <-
  data.frame(new_x=new_x_s0, y=marathon$count[which(marathon$length==0)], se_new_x_s0=se_x_s0)
results_df_s0
```

```
##       new_x       y  se_new_x_s0
## 1  2.185466 12163.5 0.0017792649
## 2  2.485590 14981.5 0.0021048939
## 3  4.466858 33584.5 0.0044010217
## 4  5.127441 39787.0 0.0051859030
## 5  2.808932 18017.5 0.0024672007
## 6  3.440652 23949.0 0.0031941978
## 7  5.157634 40070.5 0.0052218697
## 8  2.923263 19091.0 0.0025972571
## 9  3.668675 26090.0 0.0034603888
## 10 6.195343 49814.0 0.0064613159
## 11 1.500920  5736.0 0.0011240613
## 12 1.465561  5404.0 0.0010965225
## 13 1.058881  1585.5 0.0008723918
```