732A90 Computational Statistics

Lab 5

Johannes Hedström & Mikael Montén

STIMA
Institutionen för datavetenskap
Linköpings universitet

2023-12-05

Contents

1 Question 1: Hypothesis testing

In 1970, the US Congress instituted a random selection process for the military draft. All 366 possible birth dates were placed in plastic capsules in a rotating drum and were selected one by one. The first date drawn from the drum received draft number one, the second date drawn received draft number two, etc. Then, eligible men were drafted in the order given by the draft number of their birth date. In a truly random lottery there should be no relationship between the date and the draft number. Your task is to investigate whether there can be doubts concerning the randomness of the selection of the draft numbers. The draft numbers (Y=Draft No) sorted by day of year (X=Day of year) are given in the file lottery.xls. The data was originally published by the U.S. Government, and most conveniently made available online at http:// jse.amstat.org/jse_data_archive.htm (see also Starr Norton (1997) Nonrandom Risk: The 1970 Draft Lottery, Journal of Statistics Education, 5:2, DOI: 10.1080/10691898.1997.11910534)

1.1 1

Create a scatterplot of Y versus X, are any patterns visible?

```
ggplot(lott, aes(x=Day_of_year,y=Draft_No)) + geom_point() + theme_bw()
```

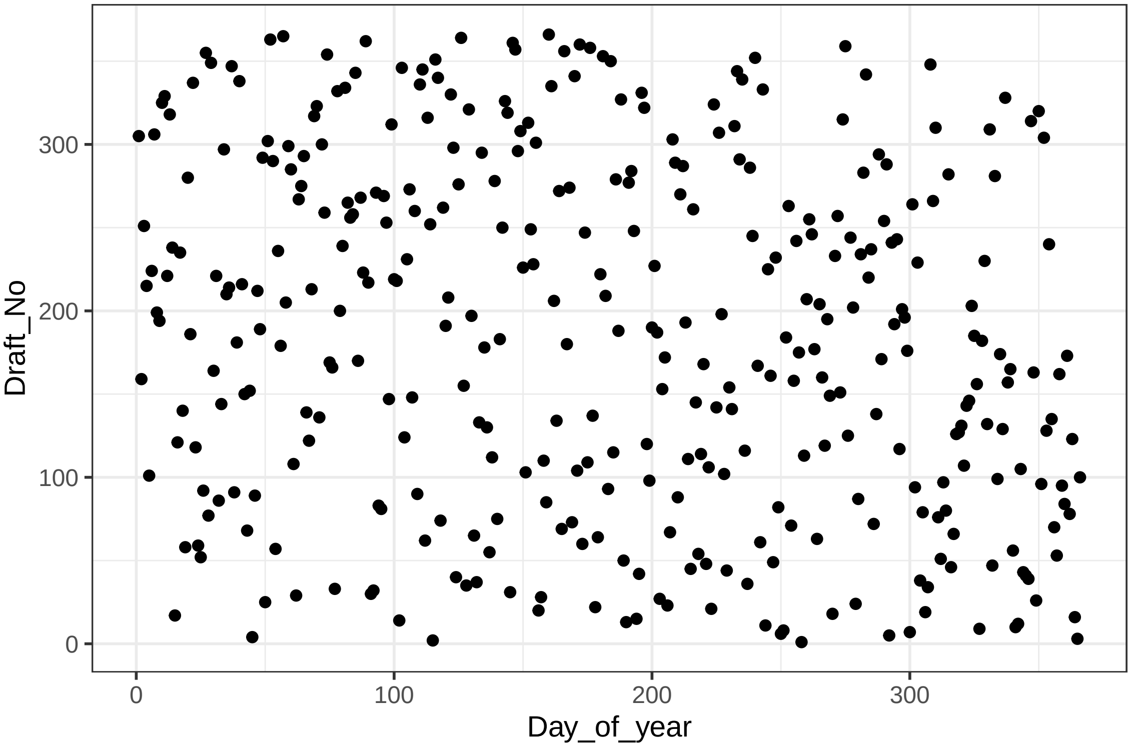


Figure 1: Scatterplot of Draft lottery

There aren't any clear visible linear patterns between Day_of_year and Draft_No, might be a weak negative relationsip one as there are less points in bottom left and top right corners. Otherwise the points are uniformly spread across both axis'.

1.2 2

Fit a curve to the data. First fit an ordinary linear model and then fit and then one using loess(). Do these curves suggest that the lottery is random? Explore how the resulting estimated curves are encoded and whether it is possible to identify which parameters are responsible for non-randomness.

```
# linear model and plotting
linmod <- lm(Draft_No~Day_of_year, lott)

plot(x=lott$Day_of_year,y=lott$Draft_No)
lines(linmod$fitted, col='blue')
```

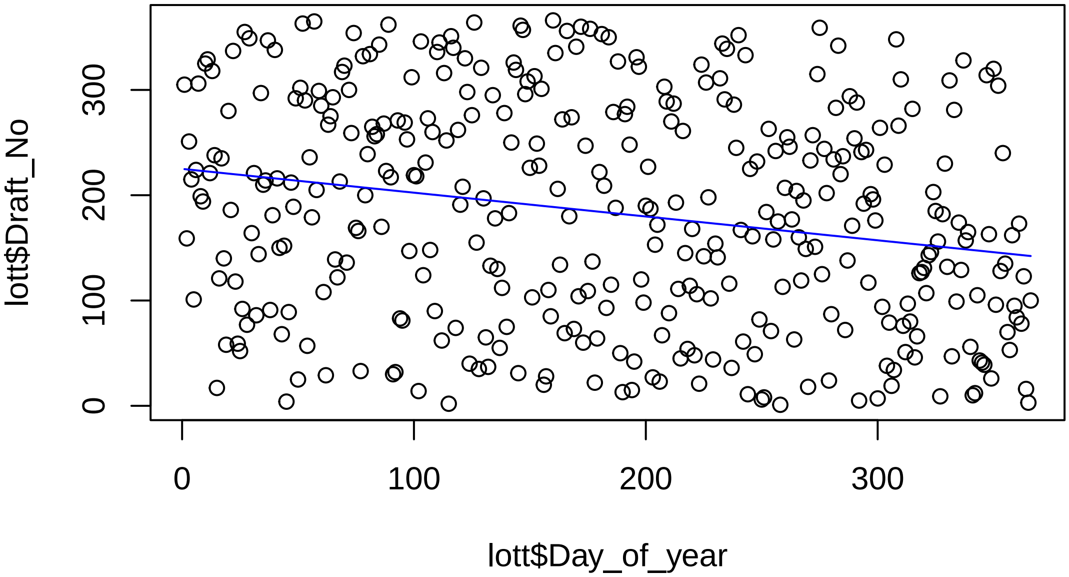


Figure 2: Linear model fit plotted

```
# fitting loess curve and plotting
loesmod <- loess(Draft_No~Day_of_year, lott)

plot(x=lott$Day_of_year,y=lott$Draft_No)
lines(loesmod$fitted, col='blue')
```

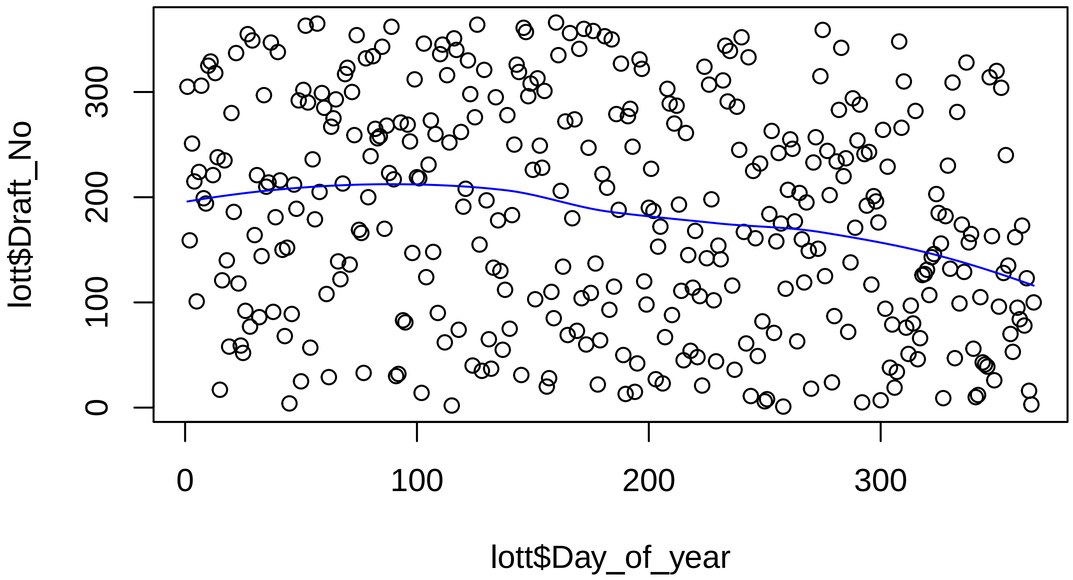


Figure 3: LOESS model fit plotted

As both curves indicate a negative relationship between the variables a case can be made for it not being random, i.e the second part of the year were more often drawn earlier than the first part. However, it could be due to sampling error and can't draw any conclusions regarding it from just 1 sample.

The loess curve is encoded with an α parameter which control the proportion of the points that should be included for the weighted distance calculation for each observation, the default is 0.75. so for observations with low value wont take the highest values in to the distance calculation as they are to far away and vice verca. So the loess curve shows that there is a difference between how people were drawn in the first and the second half of the year.

1.3 3

In order to check if the lottery is random, one can use various statistics. One such possibility is based on the expected responses. The fitted loess smoother provides an estimate $\hat{Y}$ as a function of X. It the lottery was random, we would expect $\hat{Y}$ to be a flat line, equalling the empirical mean of the observed responses, $\bar{Y}$. The statistic we will consider will be

$$S = \sum_{i}^{n} |\hat{Y}_i - \bar{Y}|$$

If S is not close to zero, then this indicates some trend in the data, and throws suspicion on the randomness of the lottery. Estimate S's distribution through a non–parametric bootstrap, taking $B = 2000$ bootstrap samples. Decide if the lottery looks random, what is the p–value of the observed value of S.

```r
# function to return parameter b1
stat1<-function(data,vn){
    data<-as.data.frame(data[vn,])
    res <-loess(Draft_No~Day_of_year, data)

    sum(abs(res$fitted - mean(data$Draft_No)))

}
# bootstrap function
res <- boot(lott,stat1,R=2000)

print(boot.ci(res, conf=0.99))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, conf = 0.99)
##
## Intervals :
## Level      Normal              Basic
## 99%   ( 3386, 12364 )   ( 3388, 12391 )
##
## Level     Percentile            BCa
## 99%   ( 4086, 13090 )   ( 3565, 12405 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```r
S_T <- sum(abs(loesmod$fitted - mean(lott$Draft_No)))
# p-values
```

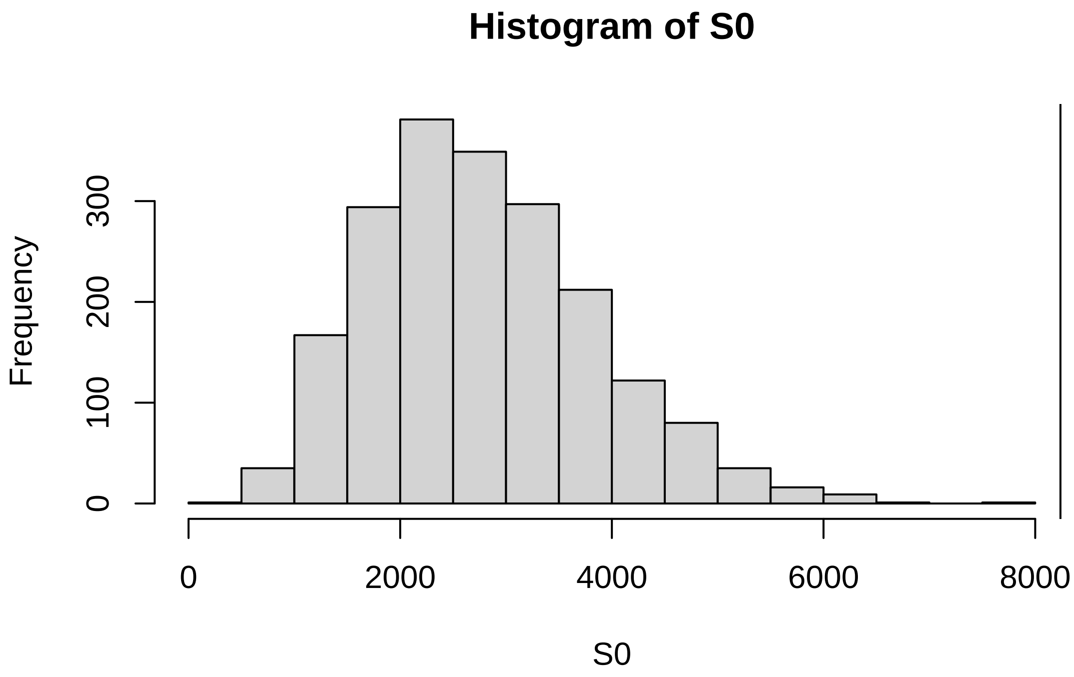
Our point estimate and confidence intervals at 1% significance level for S is $3539 \leq 8239 \leq 12370$.

```r
S0 <- c()
permutation_h0 <- function(B, data){
  n <- nrow(data)

  for(b in 1:2000){
    data_boot <- data
    data_boot$Day_of_year <- sample(data$Day_of_year,n,replace = FALSE)
    res <-loess(Draft_No~Day_of_year, data_boot)
    S0[b] <<- sum(abs(res$fitted - mean(data_boot$Draft_No)))
    #cat(b, "\r")
  }



}
permutation_h0(2000, lott)
p_permutation <- sum(S0>S_T) / 2000
hist(S0)
abline(v = S_T, col =)
```



```
knitr::kable(p_permutation, caption = "P-value for S")
```

Table 1: P-value for S

x
0

It looks to be non-random as all of the permutated values for S is lower than our true value. The point estimate for S is situated at the very end of the right tail of the histogram and there are no points that excel it which results in a very low p-value.

1.4 4

We will now want to investigate the power of our considered test. First based on the test statistic S, implement a function that tests the hypothesis

- H_0: Lottery is random

versus

- H_1: Lottery is non–random

The function should return the value of S and its p–value, based on 2000 bootstrap samples.

```
# creating MCMC hypo function
hypo_t <- function(data){
  res <-loess(Draft_No~Day_of_year, data )
  S <- sum(abs(res$fitted - mean(data$Draft_No)))
  #list("p-val" = sum(S0 > S)/2000, "S-val"=  S)
  sum(S0 > S)/2000
}
```

This function models the relationship between Y and X, calculates test statistica and calculates p-value.

1.5 5

Now we will try to make a rough estimate of the power of the test constructed in Step 4 by generating more and more biased samples:

1.5.1 a.

Create a dataset of the same dimensions as the original data. Choose k, out of the 366, dates and assign them the end numbers of the lottery (i.e., they are not legible for the draw). The remaining 366 − k dates should have random numbers assigned (from the set {1, . . . , 366 − k}). The k dates should be chosen in two ways:

1.5.1.1 i. *k consecutive dates,*

```
## a


df_1a <- matrix(0,ncol=2, nrow=366)
df_1a[,1] <- 1:366


df_1a[92:121,2] <- 366:(366-29) # filling April with 366 to 336

idx <- sample(c(1:91,122:366),336) # sampling the rest
 nr <- 1
for (i in idx) { # filling the data set from 1 to 336 from the sample

  df_1a[i,2] <- nr
  nr <- nr + 1 # draw nr
}


df_1a <- as.data.frame(df_1a)
colnames(df_1a)  <- c('Day_of_year','Draft_No')
knitr::kable(head(df_1a, n = 10), caption = "Resample with k consecutive dates")
```

Table 2: Resample with k consecutive dates

Day_of_year	Draft_No
1	134
2	195
3	241
4	286
5	275
6	135
7	154
8	26
9	177
10	7

K is here set to 30. It makes for a good number due to being divisible by 3 for task ii. but also almost 10% of the dataset which should be sufficient to show effect of increasing bias.

1.5.1.2 ii *as blocks (randomly scattered) of bk/3c consecutive dates (this is of course for k 3, and if k is not divisible by 3, then some blocks can be of length bk/3c+ 1).*

```
## a
k<- 30
k <- k/3 -1

df_1b <- matrix(0,ncol=2, nrow=366)
df_1b[,1] <- 1:366



nr <- 366
draw <- 1
  # drawing 3 points
date <- sample(1:366, 3,replace=FALSE)

while(draw <= 3){

  if(any(date[draw]:(date[draw]+k)) %in% date[-draw] || date[draw] > 357){ # checking if the dates overl
  date[draw] <- sample(1:366, 1,replace=FALSE)
  }else{
    idx <- date[draw]:(date[draw]+k)
    df_1b[idx,2] <-nr:(nr-k) # filling the random draw with new values
    draw <- draw + 1
    nr <- nr-k
  }

}
```

```
idx <- which(df_1b[,2] > 0)
vec <- c(1:366)
index <- sample(c(vec[-idx]), 336,replace = FALSE) # removing my blocks
nr <- 1 # starting from 1
for (i in index) { # filling the data set from 1 to 336 from the sample

  df_1b[i,2] <- nr
  nr <- nr + 1 # draw nr
}


  # as df...
df_1b <- as.data.frame(df_1b)
colnames(df_1b)  <- c('Day_of_year','Draft_No')
knitr::kable(head(df_1b, n = 10), caption = "Resample with randomly scattered k/3 blocks")
```

Table 3: Resample with randomly scattered k/3 blocks

Day_of_year	Draft_No
1	132
2	140
3	255
4	330
5	308
6	98
7	81
8	4
9	227
10	63

1.5.2 b

For each of the Plug the two new not–completely–random data sets from item 5a into the bootstrap test with B = 2000 and note whether it was rejected.

```
# Bootstrap for the df_1a

hypo_t(data=df_1a)
```

```
## [1] 0
```

When we have chosen 30 days in a row with falinng drawn numbers we reject the null, the draw is non-random which seem logical.

```
# Bootstrap for the df_1b
hypo_t(df_1b)
```

```
## [1] 0.64
```

When splitting the 30 days in to 3 groups and randomizing when these 3 groups over the year, the null hypothesis is not rejected.

1.5.3 c

```
set.seed(12345)
# creating variables for my loop
reject <- 0
df_reject <- data.frame('K'=0, 'method'='empty', 'P-value'=0)
df_reject <- df_reject[-1,]

k <- 1
while(reject <= 20) {



df_1a <- matrix(0,ncol=2, nrow=366)
df_1a[,1] <- 1:366


df_1a[92:(92+k -1),2] <- 366:(366-k+1) # filling April 1:k  with 366 to 366-k

idx <- sample(c(1:92,(92+k-1):366),366-k) # sampling the rest
 nr <- 1
for (i in idx) { # filling the data set from 1 to 336 from the sample

  df_1a[i,2] <- nr
  nr <- nr + 1 # draw nr
}


df_1a <- as.data.frame(df_1a)
colnames(df_1a)  <- c('Day_of_year','Draft_No')

test_1 <- hypo_t(data=df_1a)

if(test_1 < 0.10){ # checking if we reject the null
  reject <- reject +1
  df_reject <- rbind(df_reject,c(k,'i',test_1))
}
```

```
if(k>= 3){
k1 <- ceiling(k/3)


df_1b <- matrix(0,ncol=2, nrow=366)
df_1b[,1] <- 1:366
nr <- 366
draw <- 1
  # drawing 3 points
date <- sample(1:366, 3,replace=FALSE)
k1 <- k1-1
while(draw <= 3){

  if(any(date[draw]:(date[draw]+k1)) %in% date[-draw] || date[draw] > 357){ # checking if the dates over
  date[draw] <- sample(1:366, 1,replace=FALSE)
  }else{
    idx <- date[draw]:(date[draw]+k1)
    df_1b[idx,2] <-nr:(nr-k1) # filling the random draw with new values
    draw <- draw + 1
    nr <- nr-k1-1
  }

}

idx <- which(df_1b[,2] > 0)
vec <- c(1:366)
index <- sample(c(vec[-idx]), 366-length(idx),replace = FALSE) # removing my blocks
nr <- 1 # starting from 1
for (i in index) { # filling the data set from 1 to 336 from the sample

  df_1b[i,2] <- nr
  nr <- nr + 1 # draw nr
}

  # as df...
df_1b <- as.data.frame(df_1b)
colnames(df_1b)  <- c('Day_of_year','Draft_No')

test_2 <-hypo_t(df_1b)
if(test_2< 0.10){ # checking if we reject the null
  reject <- reject +1
  df_reject <- rbind(df_reject,c(k,'ii', test_2))
}


}
k <- k+1
if(k>366){break()}
#cat(k, '\r')
```

```
}
colnames(df_reject) <- c('k', 'Method from 5a', 'P-value')
df_reject
```

```
##     k Method from 5a P-value
## 1   5              i  0.0125
## 2   7              i   0.007
## 3   7             ii   0.061
## 4   8             ii   0.084
## 5  11             ii  0.0305
## 6  13              i  0.0345
## 7  14              i   0.046
## 8  15              i  0.0425
## 9  15             ii  0.0795
## 10 16              i  0.0795
## 11 16             ii  0.0745
## 12 17              i  0.0645
## 13 19              i  0.0195
## 14 21              i  0.0355
## 15 21             ii   0.071
## 16 22              i  0.0225
## 17 23              i   0.005
## 18 24              i   5e-04
## 19 25              i   0.067
## 20 26              i  0.0635
## 21 26             ii   0.007
```

How good is your test statistic at rejecting the null hypothesis of a random lottery?

The first test statistic i. which is permutated using consecutive dates has been rejected more times than the ii. blocks permutation. This is expected due to the blocks being randomly sampled out over the year and the consecutive dates more bias introducing.

2 Question 2: Bootstrap, jackknife and confidence intervals

The data you are going to continue analyzing is the database of home prices in Albuquerque, 1993. The variables present are Price; SqFt: the area of a house; FEATS: number of features such as dishwasher, refrigerator and so on; Taxes: annual taxes paid for the house. Explore the file prices1.xls. The source of the original is the Data and Story Library (https:// dasl.datadescription.com/) and it can be recovered from (https://web.archive.org/web/20151022095618/http://lib.stat.cmu.edu/DASL/Datafiles/homedat.html).

2.1 1

```
# loading the data
price <- read.csv2('prices1.csv')
```

```r
ggplot(price,aes(x=SqFt,y=Price)) + geom_point()+geom_smooth(method=lm, se=FALSE)+
  theme_bw() + ggtitle('Square feet versus Price') +
        theme(plot.title = element_text(hjust=0.5))
```

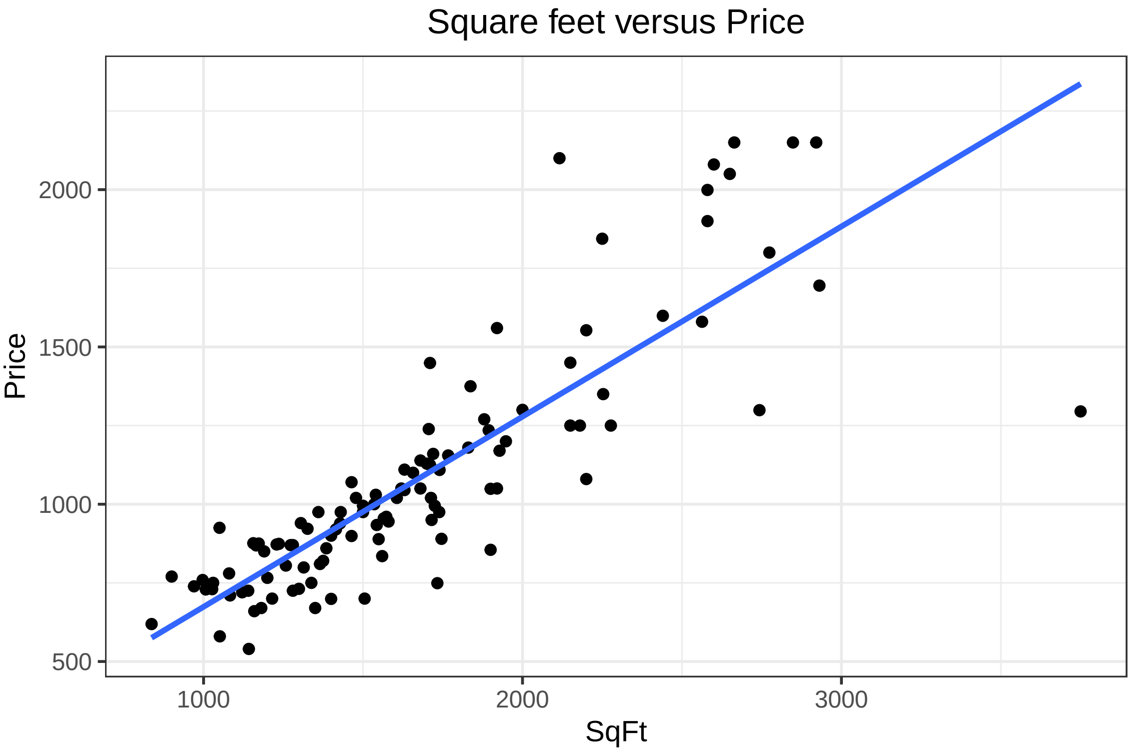


Figure 4: Scatterplot of SqFt vs Price

The relationship looks to be linear to a certain value of x and y, around 2500 for square feet and 1750 for price, then the relationship seems to differ and become nonlinear.

2.2 2

Fit a curve to the data. First fit an ordinary linear model and then fit and then one using loess(). Do these curves suggest that the lottery is random? Explore how the resulting estimated curves are encoded and whether it is possible to identify which parameters are responsible for non–randomness

While the data do seem to follow a linear trend, a new sort of pattern seems to appear around 2000ft2. Consider a new linear mode

$$Price = b + a_1 \cdot SqFt + a_2(SqFt - c)1_{SqFt>c}$$

where c is the area value where the model changes. You can determine c using an optimizer, e.g., optim(), with the residual sum of squares (RSS) as the value to be minimized. For each value of c, the objective function should estimate b, a1, and a2; then calculate (and return) the resulting RSS.

```r
fn <- function(c){
  # creating (SqFt-c) when SqFt > c
  C <- ifelse(price$SqFt>c,price$SqFt-c,0)

  new_dat <- cbind(price,C) # cbinding the new data

  mod <- lm(Price~ SqFt + C, new_dat) # creating the model

  sum((mod$fitted.values - new_dat$Price)^2) # RSS

}


optim(2300, fn,gr=NULL, method="Nelder-Mead")
```

```
## $par
## [1] 3450
##
## $value
## [1] 3309413
##
## $counts
## function gradient
##        6       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

The returned value of c is close to 3000 which mean that only the max value of square feet is the observation that will be captured by the model change.

2.3 3

Using the bootstrap estimate the distribution of c. Determine the bootstrap bias–correction and the variance of c. Compute a 95% confidence interval for c using bootstrap percentile, bootstrap BCa, and first–order normal approximation (Hint: use boot(),boot.ci(),plot.boot(),print.bootci())

Uncertainty estimation of variance estimator:

$$V\widehat{ar[T(\cdot)]} = \frac{1}{B-1}\sum_{i=1}^{B}\left(T(D_i^*) - \overline{T(D^*)}\right)^2$$

```r
bot_c<-function(data,vn){
    data<-as.data.frame(data[vn,])

   fn <- function(c){
  # creating (SqFt-c) when SqFt > c
  C <- ifelse(data$SqFt>c,data$SqFt-c,0)

  new_dat <- cbind(data,C) # cbinding the new data

  mod <- lm(Price~ SqFt + C, new_dat) # creating the model

  sum((mod$fitted.values - new_dat$Price)^2) # RSS

    }

p <- optim(2000, fn,gr=NULL, method="Nelder-Mead")

p$par

}



# bootstrap function
res4 <- boot(price,bot_c,R=2000)

var_boot <- var(res4$t)

bias <- 2*3000 -(sum(res4$t)/2000) # corrected

df_2_4 <- data.frame('Variance'= var_boot, 'Corrected.Bias' = bias)
knitr::kable(df_2_4, digits=0)
```

Variance	Corrected.Bias
474244	3438

```r
print(boot.ci(res4, conf=0.95))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res4, conf = 0.95)
##
## Intervals :
## Level      Normal              Basic
## 95%   (2088, 4788 )   (2600, 4773 )
##
## Level     Percentile            BCa
## 95%   (1227, 3400 )   (1350, 3400 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

The 95% CI intervals are as follow,

Bootstrap percentile: $(1242, 3400)$

Bootstrap BCa: $(1350, 3400)$

Normal: $(2092, 4783)$

2.4 4

Estimate the variance of using the jackknife and compare it with the bootstrap estimate.

Jackknife uncertainty estimation of variance estimator(n=B):

$$\widehat{Var[T(\cdot)]} = \frac{1}{n(n-1)}\sum_{i=1}^{n}((T_i^*) - J(T))^2$$

where

$$T_i^* = nT(D) - (n-1)T(D_i^*)$$

$$J(T) = \frac{1}{n}\sum_{i=1}^{n} T_i^*$$

```
# choosing k
k <- 15

jackky<-function(k){
  jack_c <- numeric(0)
  for (i in 1:110) {

    data<-as.data.frame(price[-i,])

   fn <- function(c){
  # creating (SqFt-c) when SqFt > c
  C <- ifelse(data$SqFt>c,data$SqFt-c,0)

  new_dat <- cbind(data,C) # cbinding the new data

  mod <- lm(Price~ SqFt + C, new_dat) # creating the model
  sum((mod$fitted.values - new_dat$Price)^2) # RSS

    }

p <- optim(2000, fn,gr=NULL, method="Nelder-Mead")

jack_c[i] <- p$par
  }
  jack_c
}



T_i <- 110*3000 - (109*jackky(1)) # calculating t_I

J_T <- mean(T_i) # mean of T_i

jack_var <- (1/(110*109)) *sum((T_i-J_T)^2) # variance of jackknife

jack_var
```

```
## [1] 6936201
```

The jackknife variance of 6 936 201 is much higher than the previous one of 471 472.

2.5 5

Summarize the results of your investigation by comparing all of the confidence intervals with respect to their length and the location of c inside them.

```
BCa <- c(1345, 3400, 3400-1345)
Normal <- c(2074, 4797, 4797-2074)
Percentile <- c(1280,3400,3400 -1280)

df_int <- data.frame(rbind(BCa, Normal,Percentile))

colnames(df_int) <- c('Lower', 'Upper', 'Interval length')

knitr::kable(df_int)
```

	Lower	Upper	Interval length
BCa	1345	3400	2055
Normal	2074	4797	2723
Percentile	1280	3400	2120

The BCa and percentile intervals are close to each other and c's location is on the right(close to upper) side of the middle of the interval for both of them. The one with the lowest lower boundary is the percentile interval and the highest value for any interval is the normal one which is also the widest interval. C's location in the normal interval is on the left(closer to lower) side of the middle of the interval.