# Lab 5

Hugo Morvan, Daniele Bozzoli

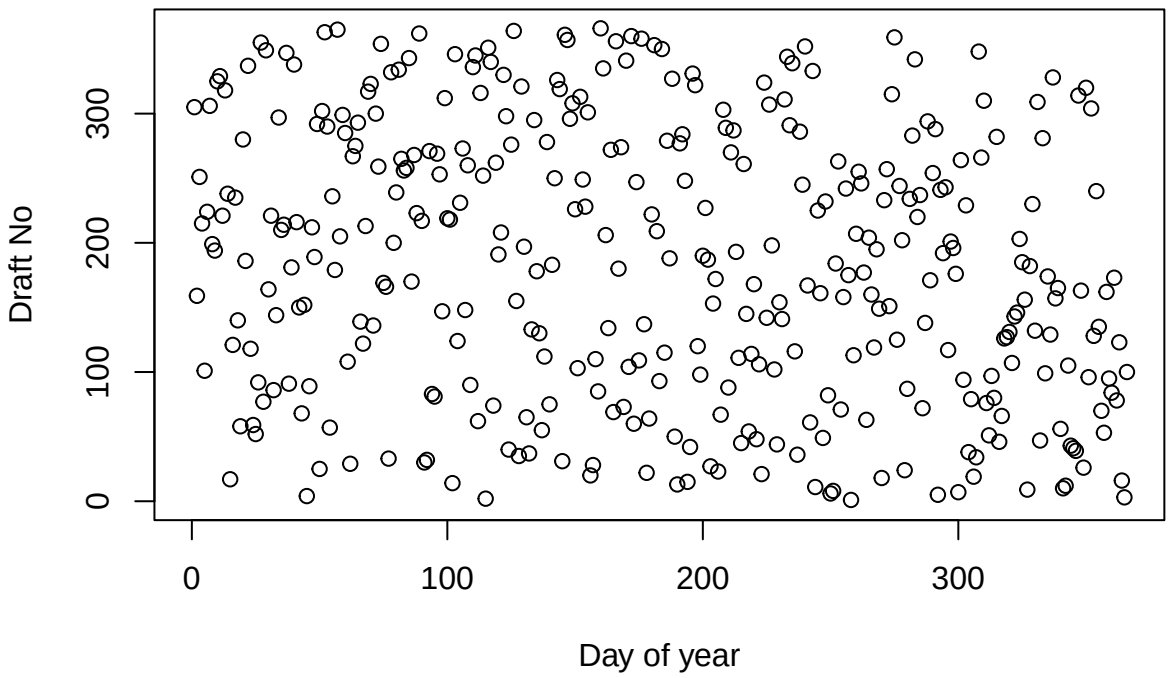
2024-01-18

## 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.csv. 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.

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

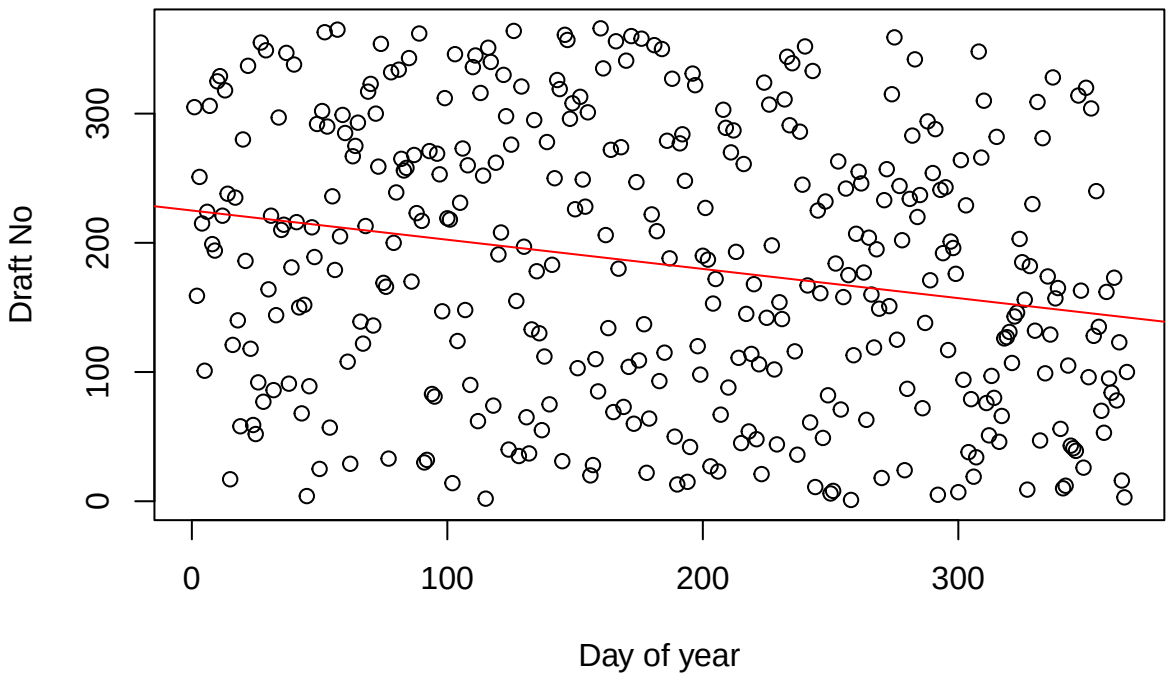



There is no clear pattern visible in this data.

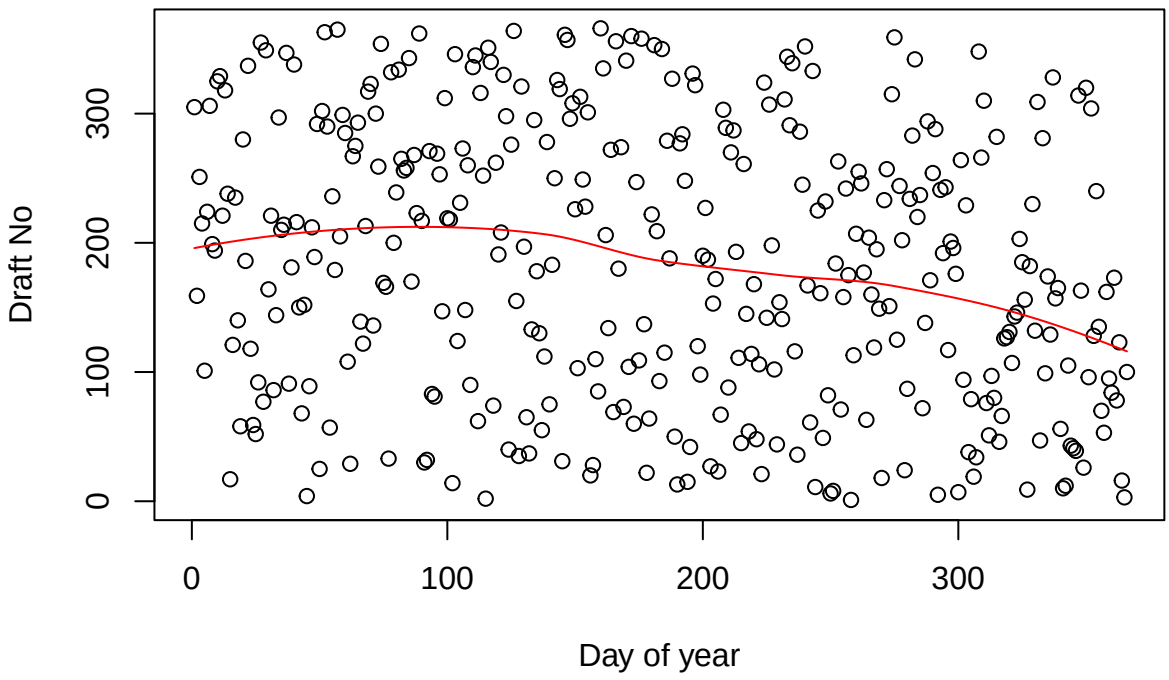
**2.**

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.

```
##
## Call:
## lm(formula = Draft_No ~ Day_of_year, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -210.837  -85.629   -0.519   84.612  196.157
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 225.00922   10.81197  20.811  < 2e-16 ***
## Day_of_year  -0.22606    0.05106  -4.427 1.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 103.2 on 364 degrees of freedom
## Multiple R-squared:  0.05109,    Adjusted R-squared:  0.04849
## F-statistic:  19.6 on 1 and 364 DF,  p-value: 1.264e-05
```




```
## Call:
## loess(formula = Draft_No ~ Day_of_year, data = data)
##
## Number of Observations: 366
## Equivalent Number of Parameters: 4.35
## Residual Standard Error: 103
## Trace of smoother matrix: 4.73  (exact)
##
## Control settings:
##   span     :  0.75
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```




There seems to be a slight downward trend in the data as shown by the red lines. This suggests that the lottery is not completely random and is dependent on the day of the year. ## 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, equaling the empirical mean of the observed responses, Y . The statistic we will consider will be

$$S = \sum_{i=1}^{n} |\hat{Y}_i - \bar{Y}|$$

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.

```
#1.3
S = sum(abs(model2$fitted-mean(data$Draft_No)))
B=2000
S_boot=rep(NA, B)

for (b in 1:B){
  data_boot=data[sample(1:nrow(data), replace = TRUE),]
  model_boot=loess(Draft_No ~ Day_of_year , data=data_boot)
  S_boot[b]=sum(abs(model_boot$fitted-mean(data_boot$Draft_No)))
}
p_value=mean(S_boot>S)
cat("S: ", S, "\n")
```

```
## S:  8238.649
```

```r
cat("p-value: ", p_value, "\n")
```

```
## p-value:  0.559
```

The p_value is 0.559 on 364 degrees of freedom. We can't reject the null hypothesis that the lottery is random.

## 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 H0 : Lottery is random versus H1 : Lottery is non–random. The function should return the value of S and its p–value, based on 2000 bootstrap samples.

```r
#1.4
#Permutation test:

test_hyp=function(data){
  model=loess(Draft_No ~ Day_of_year , data=data)
  S=sum(abs(model$fitted-mean(data$Draft_No))) #1: T(X) value of statistic from observed data
  B=2000
  S_boot=rep(NA, B)
  #generate B bootstrap samples without replacement
  for (b in 1:B){
    data_boot = data
    data_boot$Day_of_year = sample(data$Day_of_year, replace = FALSE) #without replacement
    model_boot=loess(Draft_No ~ Day_of_year , data=data_boot) #fitting the model
    #Evaluate test statistic on each permutation
    S_boot[b] = sum(abs(model_boot$fitted-mean(data_boot$Draft_No)))

  }
  p_value=mean(S_boot>=S) #4: Estimate p-value: p^ = #{T(X_gb) >= T(X)}/B
  return(list(S=S, p_value=p_value))
}

test_hyp(data)
```

```
## $S
## [1] 8238.649
##
## $p_value
## [1] 0
```

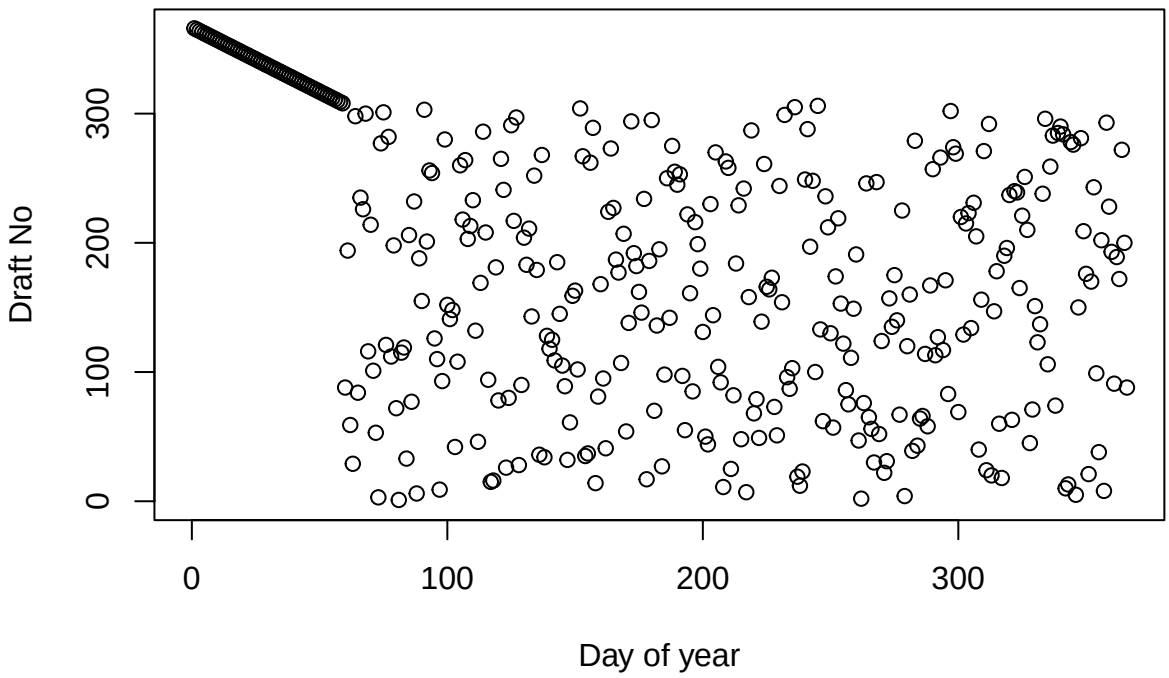
## 5.

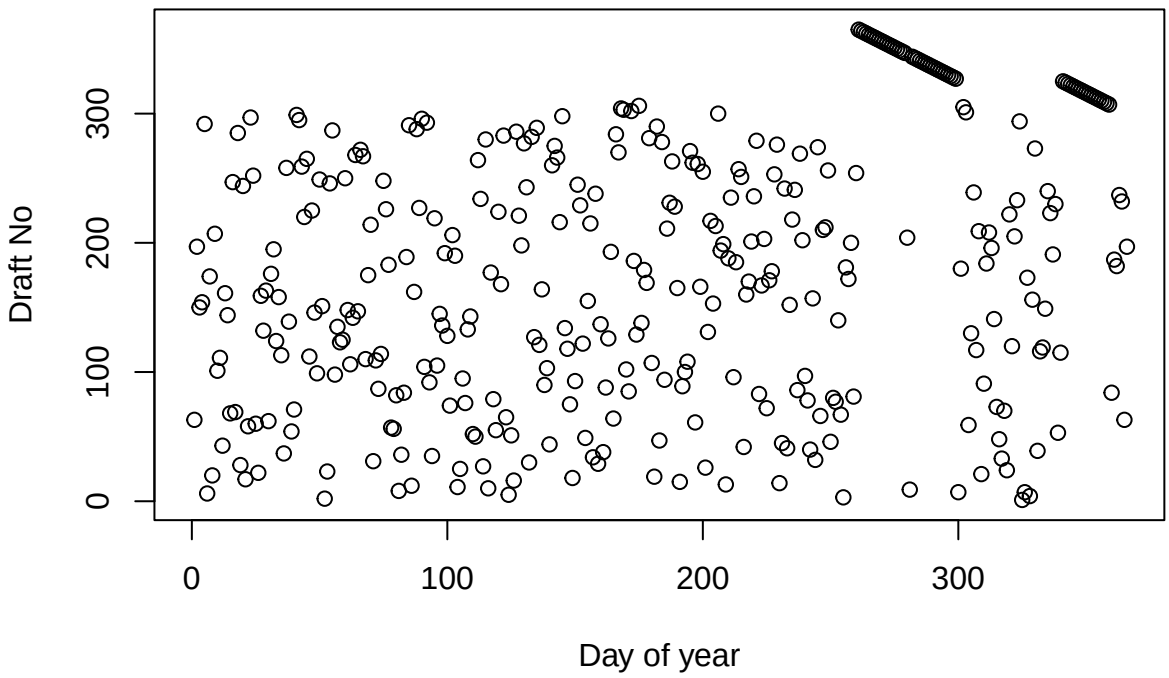
Now we will try to make a rough estimate of the power of the test constructed in Step 4 by generating more and more biased samples:

(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:

i. k consecutive dates,

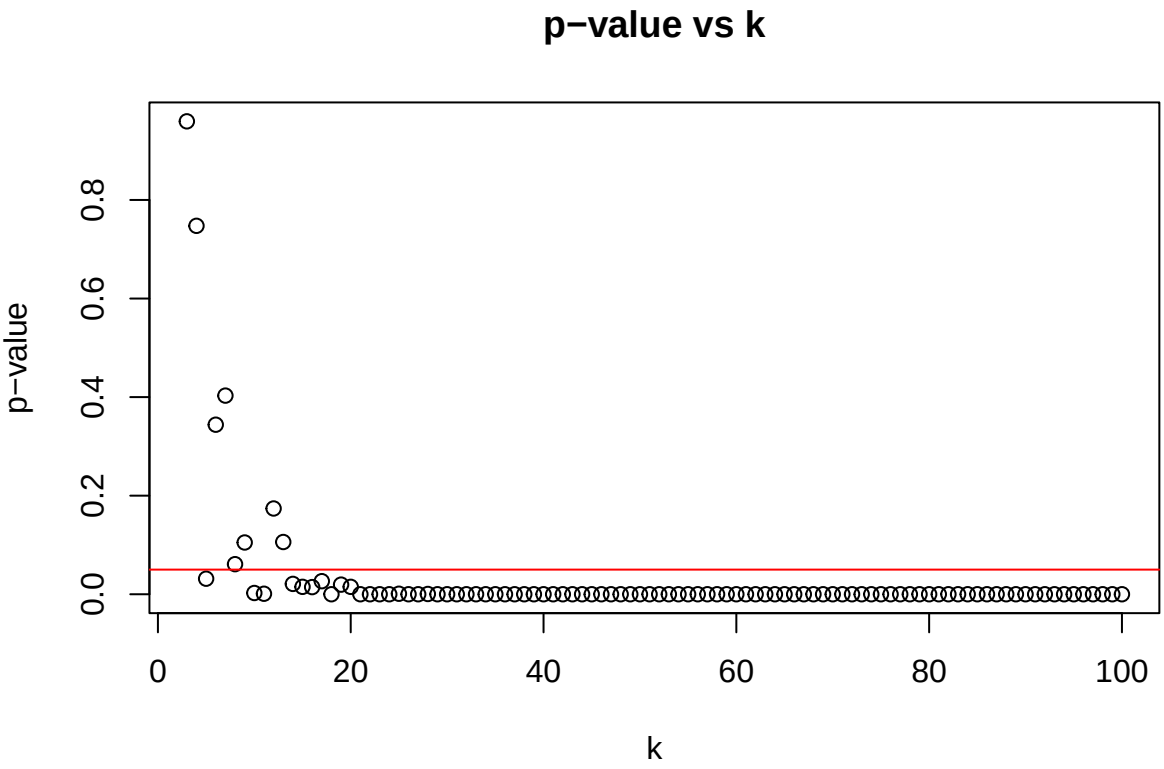
ii. as blocks (randomly scattered) of $\lfloor k/3 \rfloor$ consecutive dates (this is of course for $k >= 3$, and if k is not divisible by 3, then some blocks can be of length $\lfloor k/3 \rfloor + 1$).

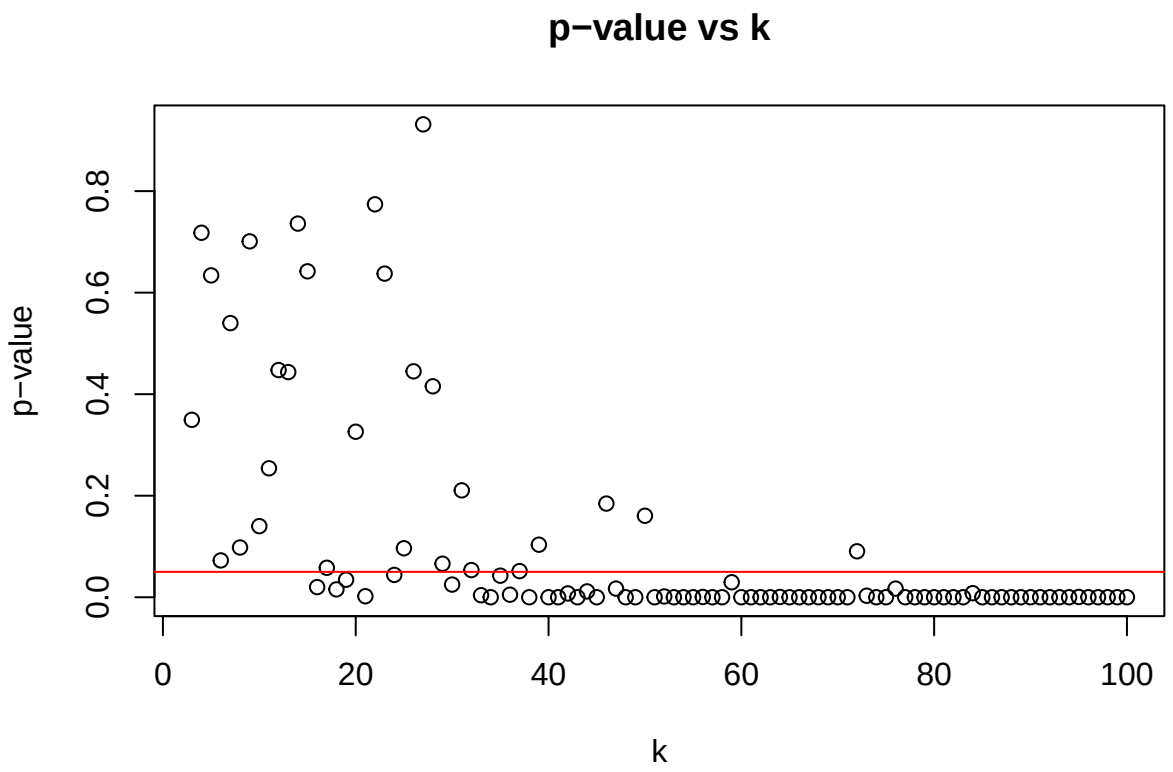

Draft No
0
100
200
300
0
100
200
300
Day of year




(b) For each of the Plug the two new not-completely-random datasets from item 5a into the bootstrap test with B = 2000 and note whether it was rejected.

(c) Repeat Steps 5a–5b for k = 1, . . . , until you have observed a couple of rejections.


p−value vs k
p−value
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100
k




In both cases, the red line represents the threshold below which we can reject the null hypothesis, i.e. the lottery is NOT random. For the consecutive dates, we can that the p-value starts to go below the threshold at around k=9, and is consistently below the threshold after k = 15. For the dates by chunks, the p-value starts to go below the threshold at around k=15, and is consistently below the threshold after k = 40.

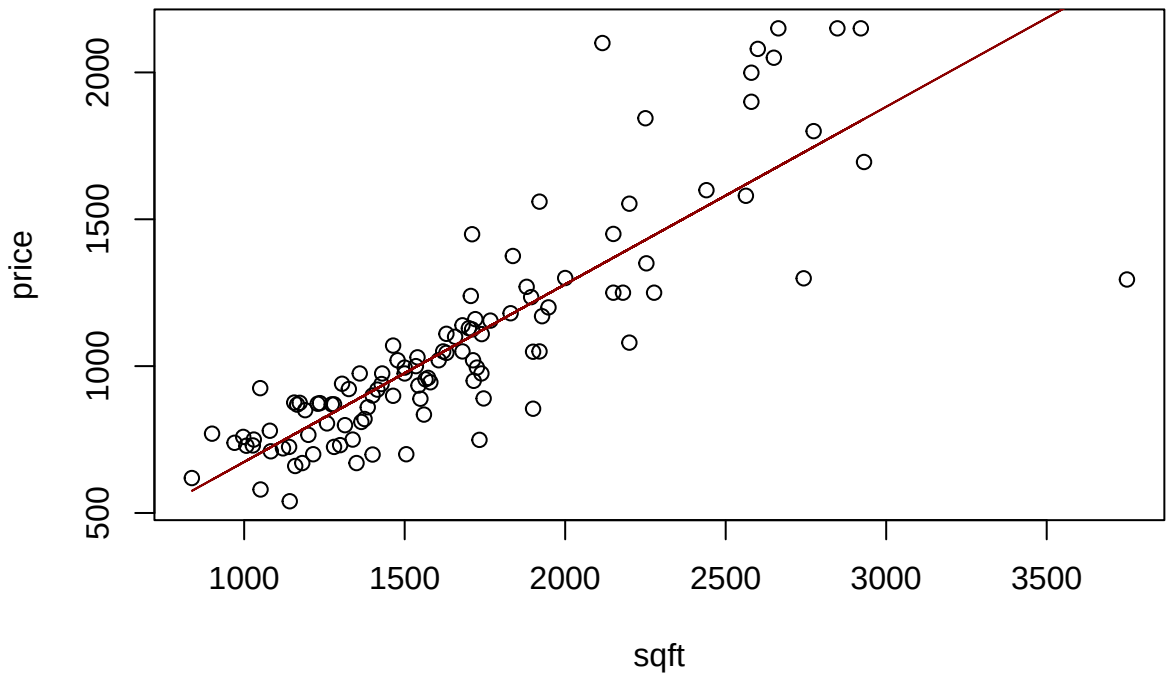
Therefore, we can say that our test statistics is not super good at rejecting the null hypothesis in the case were the draft number are fixed in several different chunks. However, it is better at rejecting the null hypothesis in the case were the draft number are fixed in consecutive dates.

## Question 2: Bootstrap, jackknife and coincidence 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).

### 1.

Create a scatter plot of SqFt versus Price. Fit a linear model to it. Does a straight line seem like a good fit?




A straight line seems to summarize the data pretty well.

**2.**

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

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

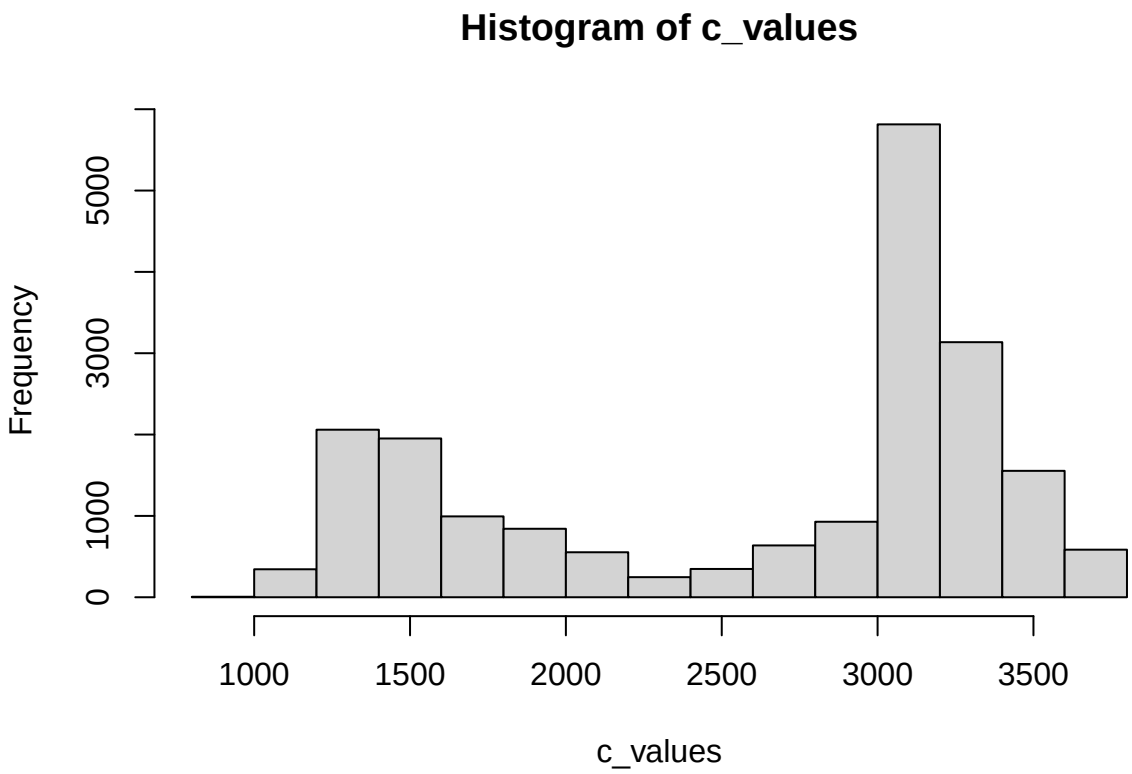
```
## [1] 3309413
```

```
## [1] 3309413
```

**3.**

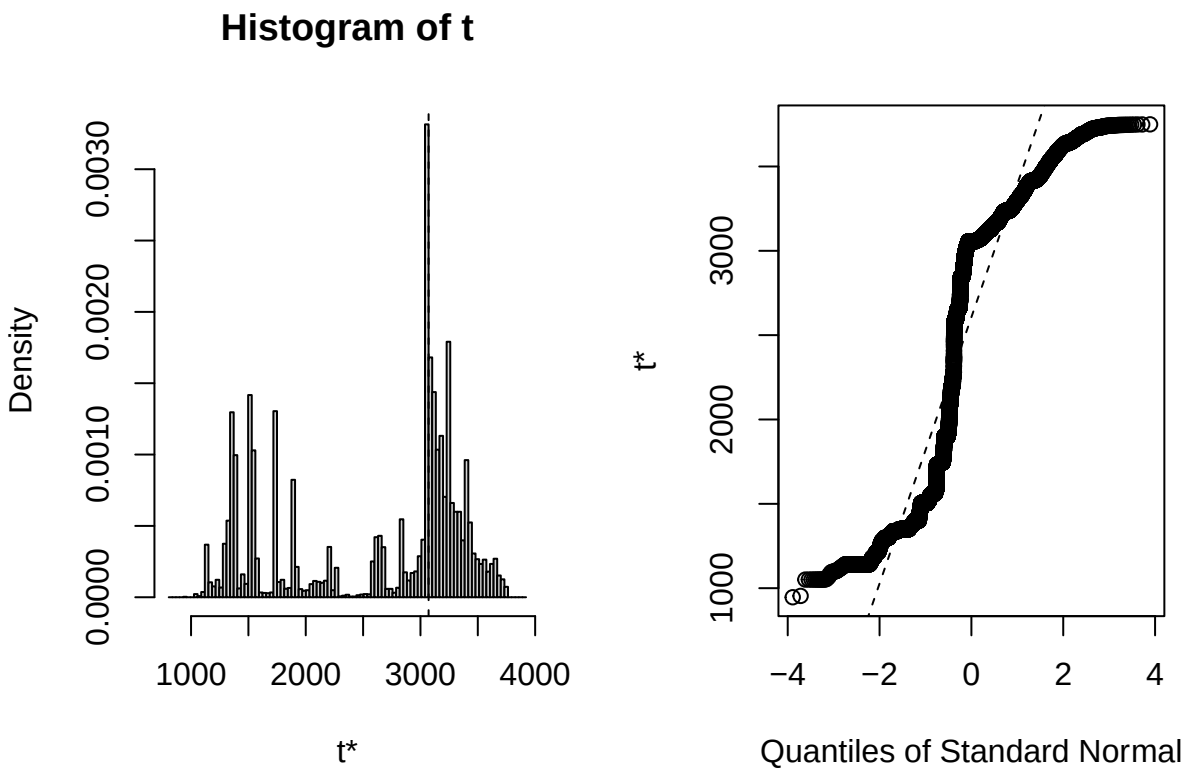
Using the bootstrap estimate the distribution of c. Determine the bootstrap bias{correction and the variance of c. Compute a 95% confidence interval for c using bootstrap percentile, bootstrap BCa, and first{order normal approximation

(Hint: use boot(),boot.ci(),plot.boot(),print.bootci())




```
## [1] 3740.168
```

```
## [1] 619114.5
```




```
## Warning in boot.ci(boot_fun): les variances de bootstrap sont nécessaires pour
## les intervalles studentisés
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_fun)
##
## Intervals :
## Level      Normal              Basic
## 95%   (1988, 5083 )   (2527, 4868 )
##
## Level     Percentile            BCa
## 95%   (1274, 3615 )   (1400, 3747 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

We seem to obtain two Gaussian-ish shaped distributions as the distribution of the optimization values.

## 4.

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

```
## [1] 10031775
```

```
## [1] 619114.5
```

### 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.

```
## Warning in boot.ci(boot_fun): les variances de bootstrap sont nécessaires pour
## les intervalles studentisés
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 20000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_fun)
##
## Intervals :
## Level      Normal              Basic
## 95%   (1988, 5083 )   (2527, 4868 )
##
## Level     Percentile            BCa
## 95%   (1274, 3615 )   (1400, 3747 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

Looking at the outcome of the analysis, we notice how the BCa and Percentile 95% confidence interval are the most similar, and furthermore the littlests, meanwhile the Normal CI gives us the largest interval. Basic CI results similar to Normal CI, with higher left interval value e smaller right value.

# Appendix

```
knitr::opts_chunk$set(echo = FALSE, cache = TRUE)
set.seed(1234)

#1.1
data=read.csv("lottery.csv", header = TRUE, sep = ";")
#summary(data)
plot(data$Day_of_year, data$Draft_No, xlim = c(0, 366), ylim = c(0, 366), xlab = "Day of year", ylab =
#1.2
model=lm(Draft_No ~ Day_of_year , data=data)
summary(model)
plot(data$Day_of_year, data$Draft_No, xlim = c(0, 366), ylim = c(0, 366), xlab = "Day of year", ylab =
abline(model, col="red")

model2=loess(Draft_No ~ Day_of_year , data=data)
summary(model2)
plot(data$Day_of_year, data$Draft_No, xlim = c(0, 366), ylim = c(0, 366), xlab = "Day of year", ylab =
lines(data$Day_of_year, model2$fitted, col="red")

#1.3
S = sum(abs(model2$fitted-mean(data$Draft_No)))
B=2000
S_boot=rep(NA, B)

for (b in 1:B){
  data_boot=data[sample(1:nrow(data), replace = TRUE),]
  model_boot=loess(Draft_No ~ Day_of_year , data=data_boot)
```

```
  S_boot[b]=sum(abs(model_boot$fitted-mean(data_boot$Draft_No)))
}
p_value=mean(S_boot>S)
cat("S: ", S, "\n")
cat("p-value: ", p_value, "\n")
#1.4
#Permutation test:

test_hyp=function(data){
  model=loess(Draft_No ~ Day_of_year , data=data)
  S=sum(abs(model$fitted-mean(data$Draft_No))) #1: T(X) value of statistic from observed data
  B=2000
  S_boot=rep(NA, B)
  #generate B bootstrap samples without replacement
  for (b in 1:B){
    data_boot = data
    data_boot$Day_of_year = sample(data$Day_of_year, replace = FALSE) #without replacement
    model_boot=loess(Draft_No ~ Day_of_year , data=data_boot) #fitting the model
    #Evaluate test statistic on each permutation
    S_boot[b] = sum(abs(model_boot$fitted-mean(data_boot$Draft_No)))

  }
  p_value=mean(S_boot>=S) #4: Estimate p-value: p^ = #{T(X_gb) >= T(X)}/B
  return(list(S=S, p_value=p_value))
}

test_hyp(data)
#1.5
#(a)
#i k consecutive dates,


data2 <- data

#Choose k, out of the 366, dates and assign them the end numbers of the lottery (i.e., they are not leg
#====================
k <- 60
#====================
k_consecutive_dates <- function(k){
  data2$Draft_No[1:k]=seq(from = 366, to = 366-k+1) #Set the first K values to be the last to draft num
  data2$Draft_No[k:366] = sample(1:(366-k), size = 366-k, replace = FALSE) #Randomize the first 366-k v
  return(data2)
}
data2 <- k_consecutive_dates(k)

model2 <- loess(Draft_No ~ Day_of_year , data=data2)
plot(data2$Day_of_year, data2$Draft_No, xlim = c(0, 366), ylim = c(0, 366), xlab = "Day of year", ylab =


#=============================================================================================================

#ii. as blocks (randomly scattered) of $\lfloor k/3 \rfloor$  consecutive dates (this is of course for
```

```r
data3 <- data

blocks_of_k3 <- function(k, data_to_copy){
  data_copy = data_to_copy
  #Problem: When sampling 3 different starting point for the chunks, we risk to have overlapping chunks
  #Solution: scale down the 0-(366-k/3) range by a factor of k/3, such that a chunk is one unit, then s
  if (k %% 3 == 0){
    # 3 "short" blocks
    scale_factor <- ceiling(k/3)
    chunk_starts <- sample(0:((366-k/3)/scale_factor), size = 3, replace = FALSE)
    chunk_starts <- chunk_starts*scale_factor
    chunk_starts <- chunk_starts[order(chunk_starts)]
    #Chunk 1 - short
    data_copy$Draft_No[chunk_starts[1]:(chunk_starts[1]+k/3-1)] <- seq(from = 366, to = 366-k/3+1)
    #Chunk 2 - short
    data_copy$Draft_No[chunk_starts[2]:(chunk_starts[2]+k/3-1)] <- seq(from = 366-k/3, to = 366-2*k/3+1
    #Chunk 3 - short
    data_copy$Draft_No[chunk_starts[3]:(chunk_starts[3]+k/3-1)] <- seq(from = 366-2*k/3, to = 366-3*k/3

  }else if (k %% 3 == 1){
    # 1 "long" block and 2 "short" blocks
    scale_factor <- ceiling(k/3)
    chunk_starts <- sample(0:((366-k/3)/scale_factor), size = 3, replace = FALSE)
    chunk_starts <- chunk_starts*scale_factor
    chunk_starts <- chunk_starts[order(chunk_starts)]
    #Chunk 1 - long
    data_copy$Draft_No[chunk_starts[1]:(chunk_starts[1]+k/3)] <- seq(from = 366, to = 366-k/3)
    #Chunk 2 - short
    data_copy$Draft_No[chunk_starts[2]:(chunk_starts[2]+k/3-1)] <- seq(from = 366-k/3, to = 366-2*k/3+1
    #Chunk 3 - short
    data_copy$Draft_No[chunk_starts[3]:(chunk_starts[3]+k/3-1)] <- seq(from = 366-2*k/3, to = 366-3*k/3

  }else{
    # 2 "long" blocks and 1 "short" block
    scale_factor <- ceiling(k/3)
    chunk_starts <- sample(0:((366-k/3)/scale_factor), size = 3, replace = FALSE)
    chunk_starts <- chunk_starts*scale_factor
    chunk_starts <- chunk_starts[order(chunk_starts)]
    #Chunk 1 - long
    data_copy$Draft_No[chunk_starts[1]:(chunk_starts[1]+k/3)] <- seq(from = 366, to = 366-k/3)
    #Chunk 2 - long
    data_copy$Draft_No[chunk_starts[2]:(chunk_starts[2]+k/3)] <- seq(from = 366-k/3, to = 366-2*k/3)
    #Chunk 3 - short
    data_copy$Draft_No[chunk_starts[3]:(chunk_starts[3]+k/3-1)] <- seq(from = 366-2*k/3, to = 366-3*k/3
  }
  #randomize the points outside of the chunks
  left_over_points <- c(1:chunk_starts[1],                        #Before first chunk
                        (chunk_starts[1]+k/3+1):chunk_starts[2], #Between first and second chunk
                        (chunk_starts[2]+k/3+1):chunk_starts[3], #Between second and third chunk
                        (chunk_starts[3]+k/3+1):366)              #After third chunk

  data_copy$Draft_No[left_over_points] <- sample(1:(366-k), size = 366-k, replace = FALSE)
  return(data_copy)
```

```r
}

data3=blocks_of_k3(k, data)


model3=loess(Draft_No ~ Day_of_year , data=data3)
plot(data3$Day_of_year, data3$Draft_No, xlim = c(0, 366), ylim = c(0, 366), xlab = "Day of year", ylab =
#(b)

#test_hyp(data2)

#test_hyp(data3)

#==== Consecutive dates ====
p_vals = c()
Ss = c()
k_max = 100
for (k in 3:k_max){
  #cat(k, "\r")
  data2=k_consecutive_dates(k)
  res = test_hyp(data2)
  p_vals[k] = res$p_value
  Ss[k] = res$S
}

plot(3:k_max, p_vals[3:k_max], xlab = "k", ylab = "p-value", main = "p-value vs k")
abline(h = 0.05, col = "red")


#==== dates by chunks ====
p_vals_chunk = c()
Ss_chunk = c()

for (k in 3:k_max){
  #cat(k, "\r")
  data3=blocks_of_k3(k, data)
  res = test_hyp(data3)
  p_vals_chunk[k] = res$p_value
  Ss_chunk[k] = res$S
}

plot(3:k_max, p_vals_chunk[3:k_max], xlab = "k", ylab = "p-value", main = "p-value vs k")
abline(h = 0.05, col = "red")

data <- read.csv("prices1.csv", sep=";")

## 1)

price <- data$Price
sqft <- data$SqFt

plot(sqft, price)
fit1 <- lm(Price ~ SqFt, data)
```

```r
lines(sqft, fit1$fitted.values, col="darkred", type="l", lwd=1)
## 2)

fit2 <- function(c, data){
  data$new <- rep(0,110)
  for (i in 1:nrow(data)){
    if (data[i,2] > c) data[i,5] <- data[i,2] - c
  }
  lm2 <- lm(Price ~ SqFt + new, data=data)
  sum(lm2$residuals^2)
}

RSS_opt <- optim(2000, fit2, data=data, method="Brent", lower=0, upper=4500)
c2 <- RSS_opt$par # new c parameter found with optimizer
RSS_opt$value

data$new <- rep(0,110)
for (i in 1:nrow(data)){
  if (data[i,2] > c2) data[i,5] <- data[i,2] - c2
}
lm3 <- lm(Price ~ SqFt + new, data=data)
sum(lm3$residuals^2) # better RSS
## 3)
library(boot)

c_values <- c()

for (i in 1:2E4){
  #if (i%%500==0) cat(i, "\r")
  id <- sample(1:110, 110, replace = TRUE)
  data_boot <- data[id,]
  c_optimizer <- optim(2000, fit2, data=data_boot, method="Brent", lower=0, upper=4000)
  c_values <<- c(c_values, c_optimizer$par)
}

hist(c_values)
2 * RSS_opt$par - sum(c_values) / 2E4            # bootstrap bias-correction
sum((c_values - mean(c_values))^2) / (2E4 - 1)     # bootstrap variance of c

opt_boot <- function(data, id){
  data_boot2 <- data[id,]
  c_optimizer2 <- optim(2000, fit2, data= data_boot2, method = "Brent", lower = 0, upper = 4000)
  c_optimizer2$par
}

boot_fun <- boot(data=data, opt_boot, R=2E4)
plot(boot_fun)

c_1 <- boot_fun$t         # c value considering bootstrap sample
c_2 <- boot_fun$t0        # c value considering whole dataset

boot.ci(boot_fun)         # confidence intervals
## 4)
```

```r
jack_c <- c()  # jackknife c values

# new function for the jackknife with 109 rows instead of 110
fit3 <- function(c, data){
  data$new <- rep(0,109)
  for (i in 1:nrow(data)){
    if (data[i,2] > c) data[i,5] <- data[i,2] - c
  }
  lm2 <- lm(Price ~ SqFt + new, data=data)
  sum(lm2$residuals^2)
}

for (i in 1:110){
  # id <- sample(1:110, 110, replace = TRUE)
  # data_boot3 <- data[id,]
  data_jack <- data[-i,]
  c_optimizer <- optim(2000, fit3, data=data_jack, method = "Brent", lower = 0, upper = 4000)
  jack_c <<- c(jack_c, c_optimizer$par)
}

n <- nrow(data)
sum((n*c_2 - (n-1)*(jack_c) - sum(jack_c)/n) ^2) / (n*(n-1)) #jackknife var
sum((c_values - mean(c_values))^2) / (2E4-1)     # bootstrap variance of c
## 5)
boot.ci(boot_fun)
```