Computational Statistics (732A90) Lab5

Anubhav Dikshit(anudi287) and Thijs Quast(thiqu264)

12 Feburary 2019

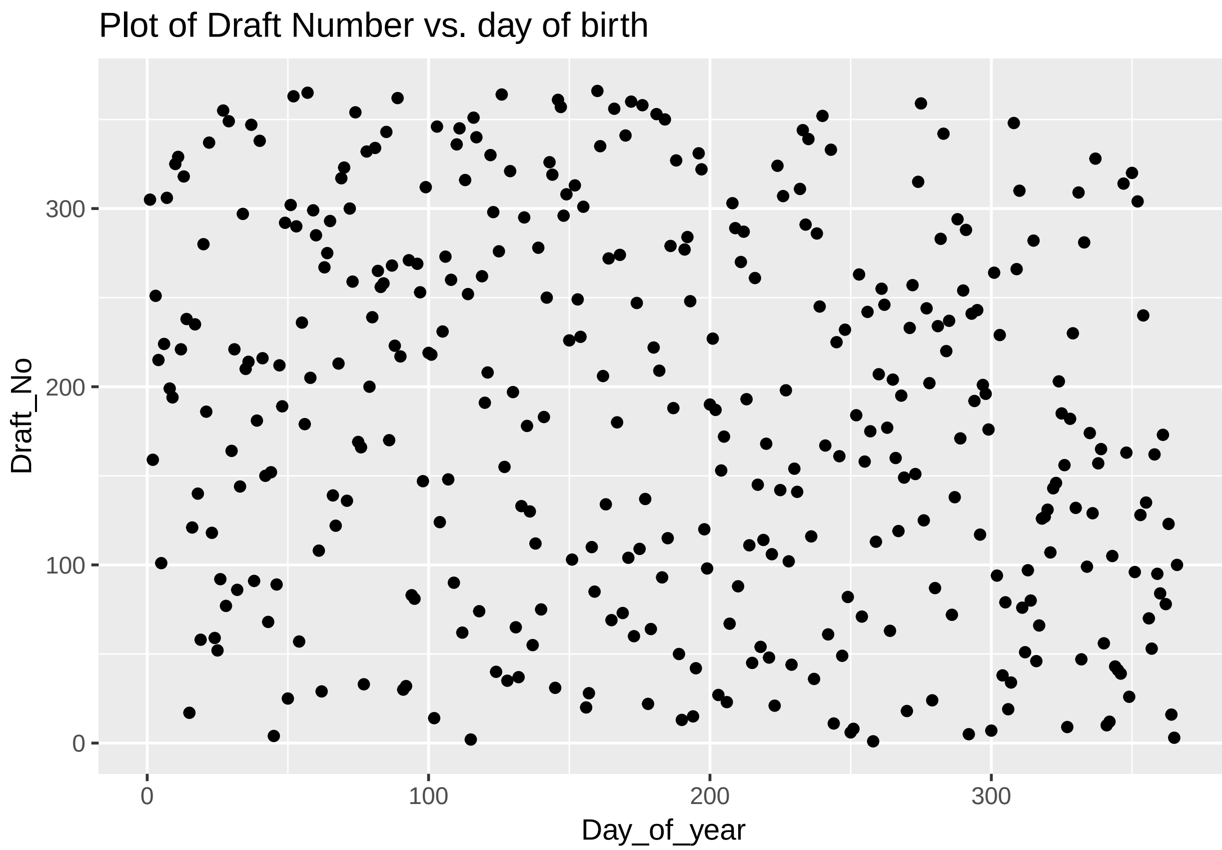
Contents

Question 1: Hypothesis testing

1. Make a scatterplot of Y(draft_no) versus X(day_of_year) and conclude whether the lottery looks random.

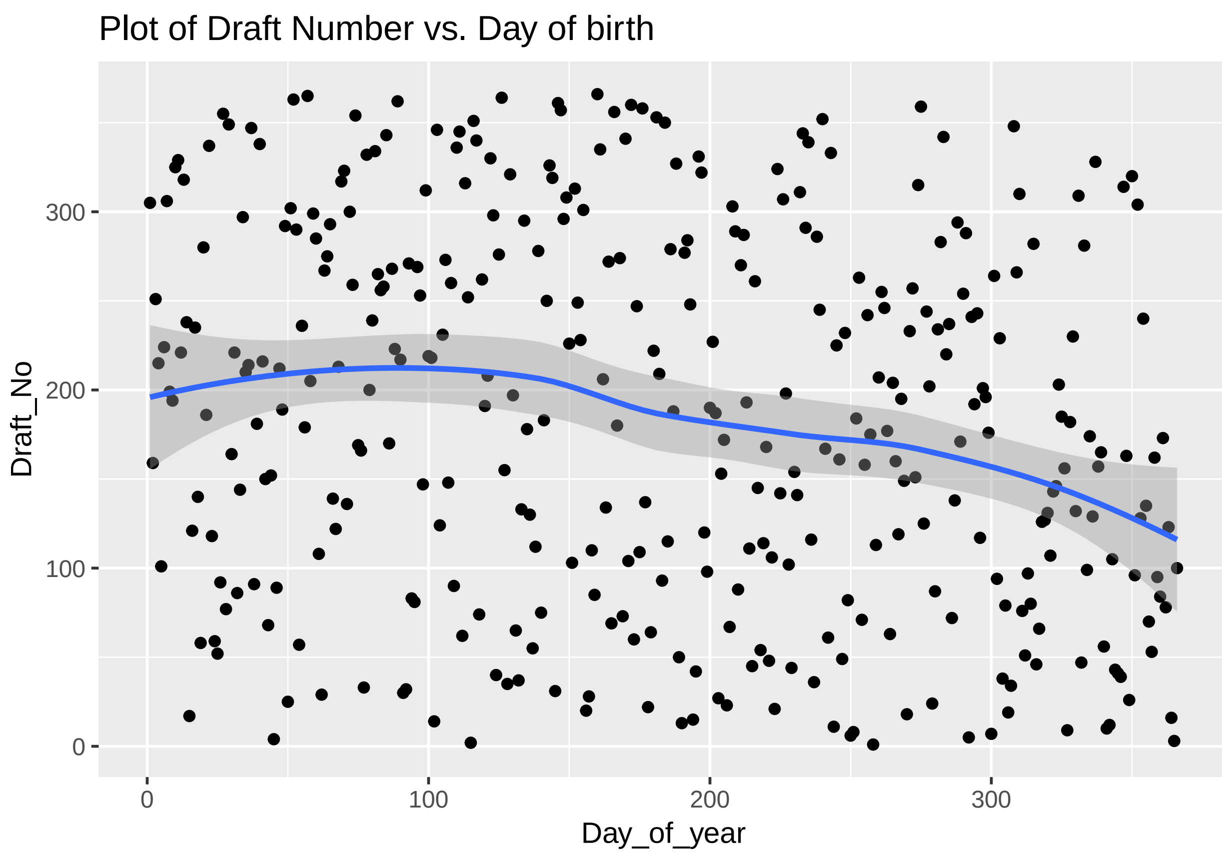
```
lottery <- read.csv("lottery.csv", sep=";")

ggplot(lottery, aes(x=Day_of_year, y = Draft_No)) + geom_point() +
  ggtitle("Plot of Draft Number vs. day of birth")
```



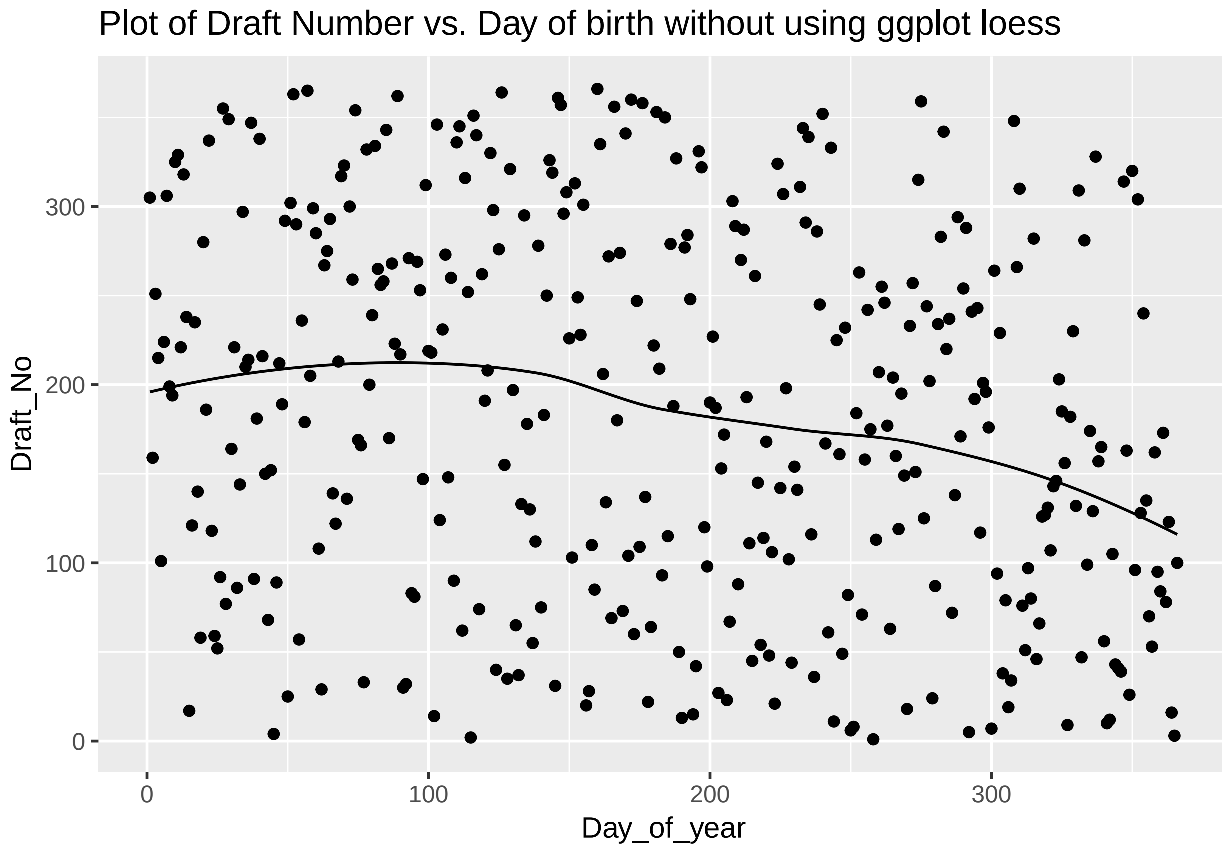
2. Compute an estimate Y(hat) of the expected response as a function of X by using a loess smoother (use loess()), put the curve Y(hat) versus X in the previous graph and state again whether the lottery looks random.

```
ggplot(lottery, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_smooth(method = loess) +
  ggtitle("Plot of Draft Number vs. Day of birth")
```




```
model <- loess(Draft_No ~ Day_of_year, lottery)
lottery$Y_hat <- predict(model, lottery)

ggplot(lottery, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_line(aes(y = Y_hat)) +
  ggtitle("Plot of Draft Number vs. Day of birth without using ggplot loess")
```



3. To check whether the lottery is random, it is reasonable to use test statistics

$$T = \frac{\hat{Y}(X_b) - \hat{Y}(X_a)}{X_b - X_a}$$

Where $X_b = argmax_x Y(X)$ and $X_a = argmin_x Y(X)$.

If this value is significantly greater than zero, then there should be a trend in the data and the lottery is not random. Estimate the distribution of T by using a non-parametric bootstrap with B = 2000 and comment whether the lottery is random or not. What is the p-value of the test?

```
library("boot")

stat1 <- function(data, index){
    data <- data[index,]
    model <- loess(Draft_No ~ Day_of_year, data)
    res <- predict(model, data)
    X_a <- data$Day_of_year[which.max(data$Draft_No)]
    X_b <- data$Day_of_year[which.min(data$Draft_No)]
    Y_a <- res[X_a]
    Y_b <-res[X_b]
    answer <- ((Y_b - Y_a) / (X_b - X_a))
    return(answer)
}
```

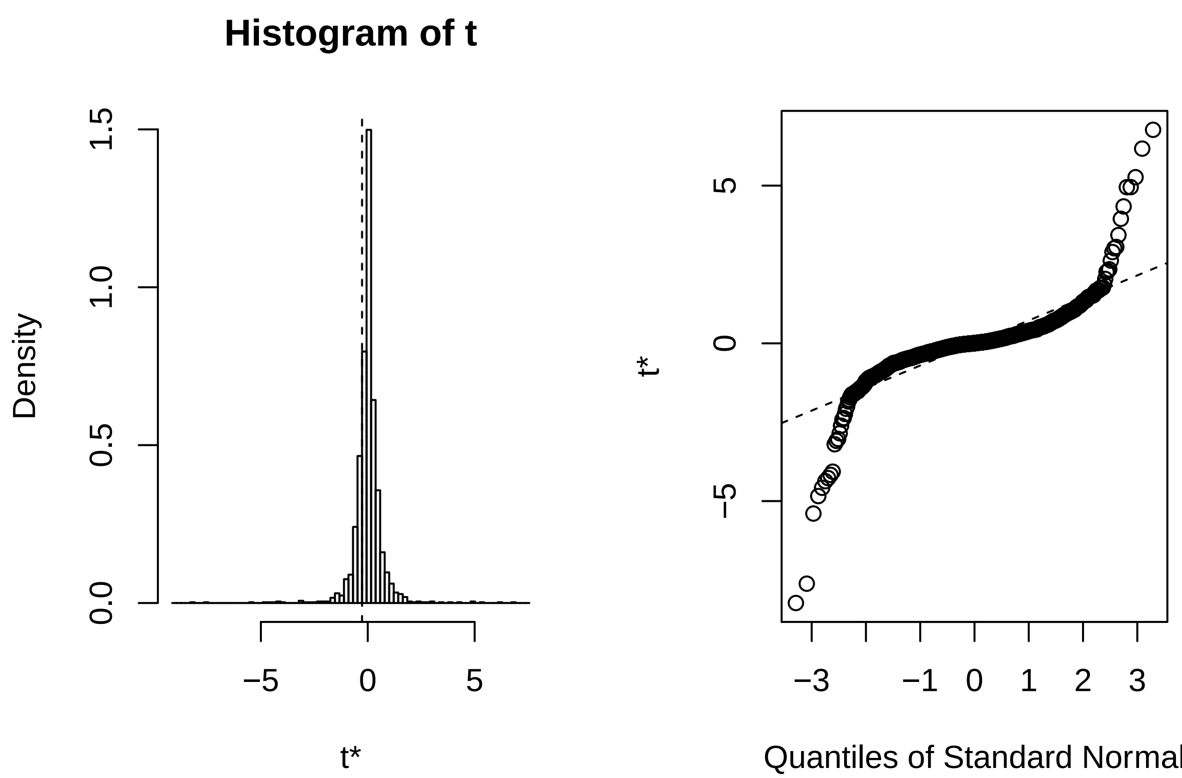
```r
res <- boot(data=lottery, statistic = stat1, R=2000)
print(boot.ci(res))
```

```
## Warning in boot.ci(res): bootstrap variances needed for studentized
## intervals

## Warning in norm.inter(t, adj.alpha): extreme order statistics used as
## endpoints

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-1.9456,  0.8549 )   (-1.7641,  0.5903 )
##
## Level     Percentile            BCa
## 95%   (-1.1247,  1.2298 )   (-8.2307,  0.0948 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```r
plot(res)
```



4.Implement a function depending on data and B that tests the hypothesis H0: Lottery is random versus H1: Lottery is non-random by using a permutation test with statistics T. The function is to return the p-value of this test. Test this function on our data with B = 2000.

```
my_permu <- function(data, index){
    data <- data[index,]
    model <- loess(Draft_No ~., data)
    res <- predict(model, data)
    X_a <- data$Day_of_year[which.max(data$Draft_No)]
    X_b <- data$Day_of_year[which.min(data$Draft_No)]
    Y_a <- res[X_a]
    Y_b <-res[X_b]
    answer <- ((Y_b - Y_a) / (X_b - X_a))
  return(answer)
}

data <- lottery
data$Month <- NULL
res <- boot(data=lottery, statistic = stat1, R=2000)
```

Question 2: Bootstrap, jackknife and confidence intervals

1. Plot the histogram of Price. Does it remind any conventional distribution? Compute the mean price.

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

ggplot(data=price_data,aes(Price)) +
  geom_histogram(bins=20) +
  ggtitle("Histogram of Price")
```

2. Estimate the distribution of the mean price of the house using bootstrap. Determine the bootstrap bias-correction and the variance of the mean price. Compute a 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and first-order normal approximation

Bias correction

$$T1 = 2.T(D) - \frac{1}{D}\sum_{i=1}^{B} T_i^*$$

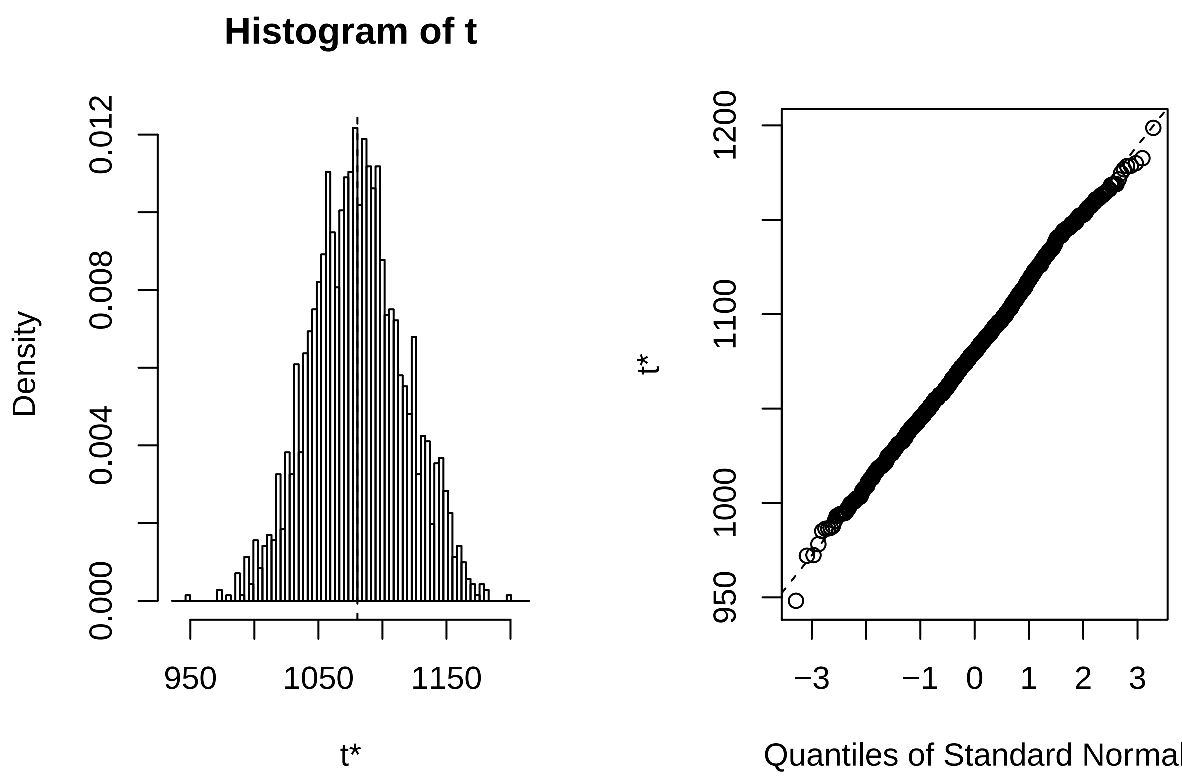
```
# Estimation of mean of Price
stat_mean <- function(data, index){
    data <- data[index,]
    answer <- mean(data$Price)
    return(answer)
}

res <- boot::boot(data=price_data, statistic = stat_mean, R=2000)
res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = price_data, statistic = stat_mean, R = 2000)
```

```
##
##
## Bootstrap Statistics :
##      original     bias    std. error
## t1* 1080.473 0.3080682    36.16977
```

```r
plot(res,index = 1)
```




```r
#95% CI for mean using percentile
boot.ci(res, index=1, type=c('perc'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = c("perc"), index = 1)
##
## Intervals :
## Level     Percentile
## 95%   (1011, 1152 )
## Calculations and Intervals on Original Scale
```

```r
#95% CI for mean using bca
boot.ci(res, index=1, type=c('bca'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = res, type = c("bca"), index = 1)
##
## Intervals :
## Level       BCa
## 95%   (1015, 1156 )
## Calculations and Intervals on Original Scale
```

```
#95% CI for mean using first order normal
boot.ci(res, index=1, type=c('norm'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = c("norm"), index = 1)
##
## Intervals :
## Level      Normal
## 95%   (1009, 1151 )
## Calculations and Intervals on Original Scale
```

```
# Bias-correction and Varience of Price

stat_bias_correction <- function(data, index){
    t_d <- 2*mean(data$Price)
    data2 <- data[index,]
    answer <- t_d - mean(data2$Price)
    return(answer)
}

res <- boot(data=price_data, statistic = stat_bias_correction, R=2000)
res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = price_data, statistic = stat_bias_correction, R = 2000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 1080.473 0.7735909    35.29654
```

```
print(boot.ci(res))
```

```
## Warning in boot.ci(res): bootstrap variances needed for studentized
## intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res)
##
```

```
## Intervals :
## Level      Normal              Basic
## 95%   (1011, 1149 )   (1009, 1150 )
##
## Level     Percentile            BCa
## 95%   (1011, 1152 )   (1007, 1148 )
## Calculations and Intervals on Original Scale
```

```
# Varience using bootstrap
stat_varience <- function(data, index){
    data2 <- data[index,]
    n <- length(data2)
    answer <- (1/(n-1)) * sum(data2$Price - mean(data2$Price))^2
    return(answer)
}

res <- boot(data=price_data, statistic = stat_varience, R=2000)
res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = price_data, statistic = stat_varience, R = 2000)
##
##
## Bootstrap Statistics :
##                               original                         bias
## t1* 0.000000000000000000000002481542 0.00000000000000000000000141906
##                            std. error
## t1* 0.00000000000000000000001538808
```

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

```
stat_jackknife_varience <- function(data, index){
    data2 <- data[-index,]
        n <- length(data2)
    answer <- (1/(n-1)) * sum(data2$Price - mean(data2$Price))^2
    return(answer)
}

res <- boot(data=price_data, statistic = stat_jackknife_varience, R=2000)
res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = price_data, statistic = stat_jackknife_varience,
##     R = 2000)
##
```

```
##
## Bootstrap Statistics :
##     original                           bias
## t1*        0 0.00000000000000000000002138245
##                          std. error
## t1* 0.00000000000000000000002103544
```

4. Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals.

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)
library(dplyr)
library(ggplot2)
lottery <- read.csv("lottery.csv", sep=";")

ggplot(lottery, aes(x=Day_of_year, y = Draft_No)) + geom_point() +
  ggtitle("Plot of Draft Number vs. day of birth")
ggplot(lottery, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_smooth(method = loess) +
  ggtitle("Plot of Draft Number vs. Day of birth")


model <- loess(Draft_No ~ Day_of_year, lottery)
lottery$Y_hat <- predict(model, lottery)

ggplot(lottery, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_line(aes(y = Y_hat)) +
  ggtitle("Plot of Draft Number vs. Day of birth without using ggplot loess")


library("boot")

stat1 <- function(data, index){
    data <- data[index,]
    model <- loess(Draft_No ~ Day_of_year, data)
    res <- predict(model, data)
    X_a <- data$Day_of_year[which.max(data$Draft_No)]
    X_b <- data$Day_of_year[which.min(data$Draft_No)]
    Y_a <- res[X_a]
    Y_b <-res[X_b]
    answer <- ((Y_b - Y_a) / (X_b - X_a))
    return(answer)
}

res <- boot(data=lottery, statistic = stat1, R=2000)
print(boot.ci(res))
```

```r
plot(res)

my_permu <- function(data, index){
    data <- data[index,]
    model <- loess(Draft_No ~., data)
    res <- predict(model, data)
    X_a <- data$Day_of_year[which.max(data$Draft_No)]
    X_b <- data$Day_of_year[which.min(data$Draft_No)]
    Y_a <- res[X_a]
    Y_b <-res[X_b]
    answer <- ((Y_b - Y_a) / (X_b - X_a))
  return(answer)
}

data <- lottery
data$Month <- NULL
res <- boot(data=lottery, statistic = stat1, R=2000)


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

ggplot(data=price_data,aes(Price)) +
  geom_histogram(bins=20) +
  ggtitle("Histogram of Price")

# Estimation of mean of Price
stat_mean <- function(data, index){
    data <- data[index,]
    answer <- mean(data$Price)
    return(answer)
}

res <- boot::boot(data=price_data, statistic = stat_mean, R=2000)
res
plot(res,index = 1)

#95% CI for mean using percentile
boot.ci(res, index=1, type=c('perc'))

#95% CI for mean using bca
boot.ci(res, index=1, type=c('bca'))

#95% CI for mean using first order normal
boot.ci(res, index=1, type=c('norm'))

# Bias-correction and Varience of Price

stat_bias_correction <- function(data, index){
    t_d <- 2*mean(data$Price)
    data2 <- data[index,]
    answer <- t_d - mean(data2$Price)
    return(answer)
}
```

```r
res <- boot(data=price_data, statistic = stat_bias_correction, R=2000)
res
print(boot.ci(res))

# Varience using bootstrap
stat_varience <- function(data, index){
    data2 <- data[index,]
    n <- length(data2)
    answer <- (1/(n-1)) * sum(data2$Price - mean(data2$Price))^2
    return(answer)
}

res <- boot(data=price_data, statistic = stat_varience, R=2000)
res



stat_jackknife_varience <- function(data, index){
    data2 <- data[-index,]
        n <- length(data2)
    answer <- (1/(n-1)) * sum(data2$Price - mean(data2$Price))^2
    return(answer)
}

res <- boot(data=price_data, statistic = stat_jackknife_varience, R=2000)
res
```