Laboration report in Computational Statistics

Laboration 5

732A90

Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

05 December 2023

1 Question 1: Hypothesis testing

Description of data:

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.

The aim for the question:

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.

1.1 1.1

Question

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

Answer

```
lottery <- read.csv2("lottery.csv", stringsAsFactors = TRUE)

ggplot(lottery, aes(x = Day_of_year, y = Draft_No)) + geom_point() +
  theme_bw() + labs(x = "Day of year",y = "Draft number")
```

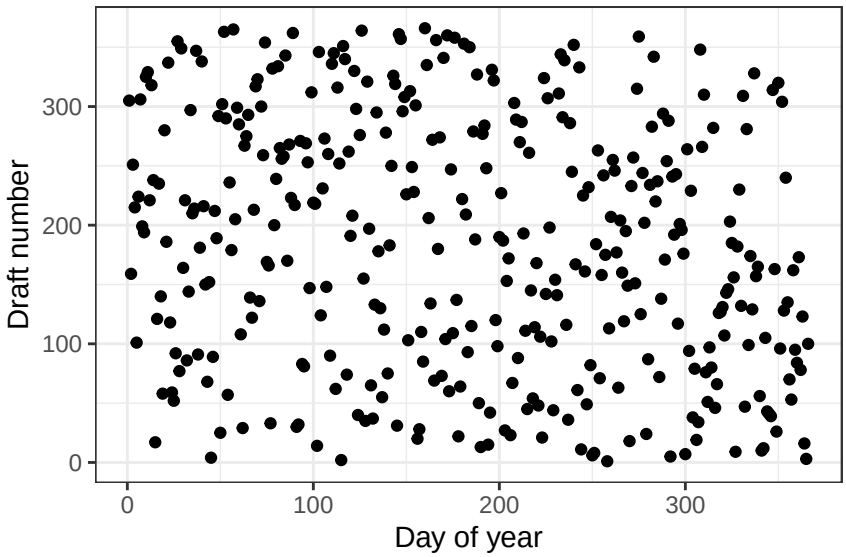

Figure 1: Scatterplot of draft numbers versus Day of year

Figure 1 does not show any clear patterns.

1.2 1.2

Question

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.

Answer

```
ggplot(lottery, aes(x = Day_of_year, y = Draft_No)) +
  geom_point() +
  theme_bw() +
  labs(x = "Day of year", y = "Draft number") +
  stat_smooth(method = lm, se = FALSE, aes(color = "Linear\nRegression")) +
  stat_smooth(method = loess, se = FALSE, aes(color = "Loess")) +
  scale_color_manual(values = c("indianred", "steelblue"),
                     name = "Method",
                     labels = c("Linear\nRegression", "Loess"))
```

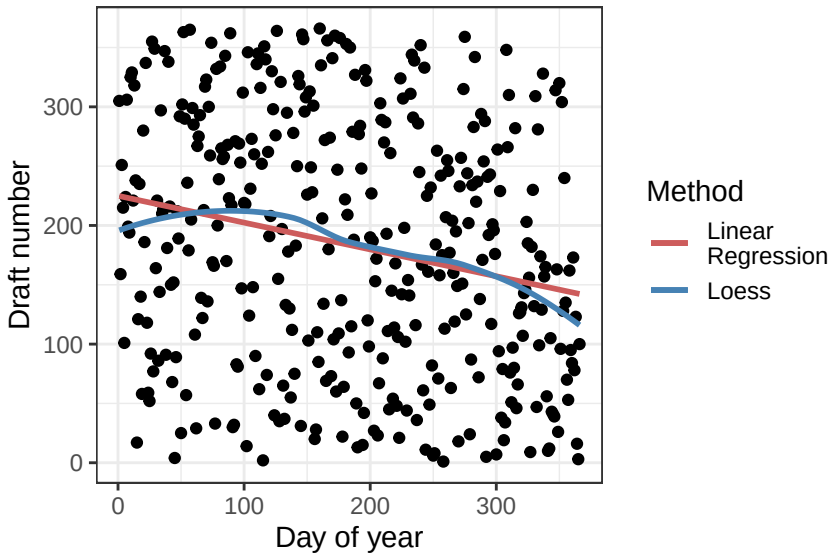

Figure 2: Scatterplot of draft numbers versus Day of year with lm and loess regression lines

```
mod_loess <- loess(Draft_No ~ Day_of_year, data=lottery)
T_X <- sum(abs(mod_loess$fitted - mean(lottery$Draft_No)))
```

In figure 2, the plot suggest that lottery might not be random because it seems to be a correlation between "Day of year" and "Draft year".

1.3 1.3

Question

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

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

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.

Answer

```
# Y-bar
mean_y <- mean(lottery$Draft_No)

# Function to used for bootstrap
stat1 <- function(data, vn){
  data <- as.data.frame(data[vn, ])
  fit_mod <- loess(Draft_No ~ Day_of_year, data=data)
  s <- sum(abs(fit_mod$fitted - mean_y))
  return(s)
}

set.seed(13)
# Number of bootstrap samples
B <- 2000

res <- boot(lottery, stat1, R=B)
res$t0
```

```
## [1] 8238.649
```

The value of S from 2000 bootstrap samples is around 8238, the distribution of S is presented in figure 3.

```r
plot_data <- data.frame(x=res$t)

ggplot(plot_data) +
  geom_histogram(aes(x=x),fill="steelblue", color="black") +
  theme_bw() +
  labs(x="S")
```

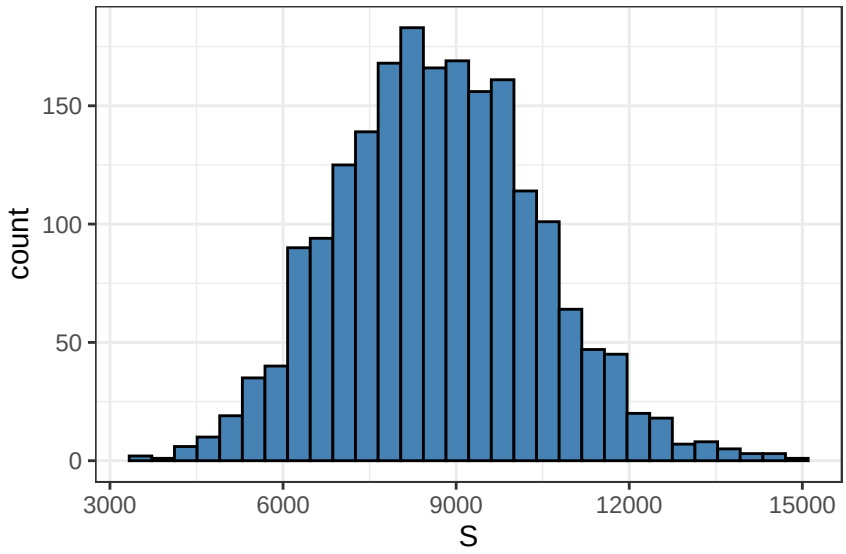

Figure 3: Histogram over S-values for 2000 bootstrap samples

In figure 3, the distribution of S appears to be normal distributed. The distribution of S under the assumption of the lottery is random(null hypothesis) was generated with permutation test with 2000 samples. The histogram of S under null hypothesis is presented in figure 4.

```r
# Simulating the null distribution with permutation test.
set.seed(13)
index <- list()
# Creates 2000 permutations.
for(iter in 1:2000){
  new_data_index <- list(sample(1:366, 366))
  index[[iter]] <- new_data_index
}

# Ensures that the all the permutations are unique
index <- unique(index)

mean_y <- mean(lottery$Draft_No)
get_s <- function(index){
  temp_data <- lottery
  temp_data$Draft_No <- index
```

```
  fit_mod <- loess(Draft_No ~ Day_of_year, data=temp_data)
  s <- sum(abs(fit_mod$fitted - mean_y))
  return(s)
}

null_s <- c()
for(iter in 1:2000){
  null_s[iter] <- get_s(unlist(index[[iter]]))
}

plot_data <- data.frame(x=null_s)
ggplot(plot_data) +
  geom_histogram(aes(x=x),fill="indianred", color="black") +
  theme_bw() +
  labs(x="S")
```

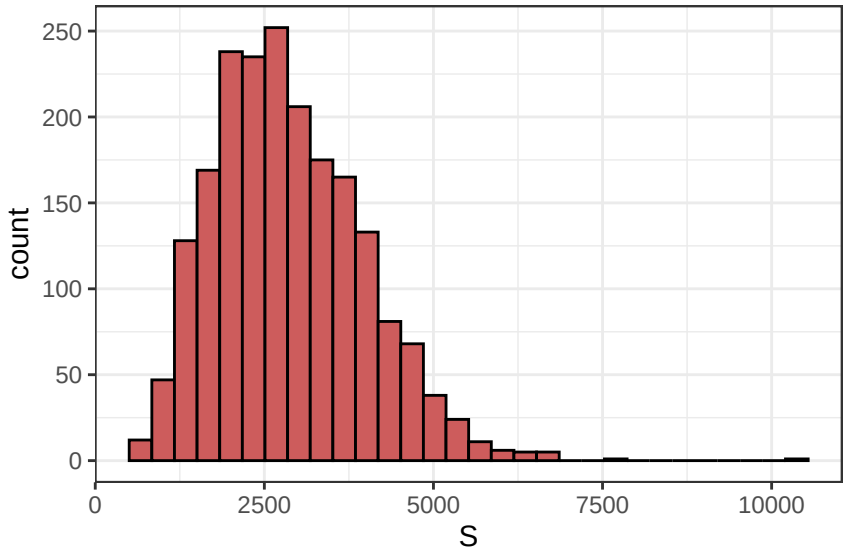

Figure 4: Null distribution of S under assumption of random lottery.

```
B <- 2000
p <- sum(null_s > T_X)/B
p
```

```
## [1] 5e-04
```

The p-value from the observed value of S from bootstrap sampling under the null hypothesis is 0.0005. For this whole assignment the significant level will be $\alpha = 0.05$, this will lead to the conclusion that the lottery does not look random.

1.4 1.4

Question

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

$$H_0: \text{ Lottery is random}$$

versus

$$H_1: \text{ Lottery is non–random}$$

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

Answer The implemented function is as follows:

```
# Function that runs 2000 bootstrap samples and returns S and p-value under null hypothesis
mean_y <- mean(lottery$Draft_No)
get_boot <- function(data){
  # Calculates value of S for bootstrap samples
  stat2 <- function(data, vn){
    data <- as.data.frame(data[vn, ])
    fit_mod <- loess(Draft_No ~ Day_of_year, data=data)
    s <- sum(abs(fit_mod$fitted - mean_y))
    return(s)
  }
  # Bootstrap sampling
  B <- 2000
  res <- boot(data, stat2, R=B)

  # Test statistics
  s <- res$t0
  p <- sum(null_s > s)/B
  result <- list(s=s, p=p)
  return(result)
}
```

1.5 1.5

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:

Question

- (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 $\{1, \ldots, 366 - k\}$). 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 $k \geq 3$, and if k is not divisible by 3, then some blocks can be of length $\lfloor k/3 \rfloor + 1$).

- (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 $k = 1, \ldots,$ until you have observed a couple of rejections.

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

Answer We implemented the code and ran the code until we got at least 3 rejections. The code and output are presented as follows:

```
# Function that does bootstrap sampling for both i) and ii)
bias_data <- function(k){
  # Data for i.
  data_i <- lottery
  data_i$Day_of_year[1:k] <- (366-k+1):366
  data_i$Day_of_year[(k+1):366] <- sample(1:(366-k), (366-k))

  # Data for ii.
  data_ii <- lottery
  # Split the data into chunks
  data_chunk <- split(x, sort(rep_len(1:k, length(x))))
  # Randomise order of chunks
  blocks <- sample(1:k, k)

  # Chose randomize block 1 to change values for
  index <- unname(unlist(data_chunk[blocks[[1]]]))
  data_ii$Day_of_year[index] <- (366-(length(index))+1):366
  data_ii$Day_of_year[-index] <- sample(1:(366-(length(index))), (366-(length(index))))

  result_i <- get_boot(data_i)
  result_ii <- get_boot(data_ii)
  result <- list(result_i, result_ii)
  return(result)
}


x <- 1:366
k <- 3
rejection <- 0
info_list_i <- list()
info_list_ii <- list()
while(k <= 27){
  result <- bias_data(k)
  cat("i)  For k=", k, ", s value is: ", result[[1]]$s, " with the p-value: ", result[[1]]$p, ".\n", sep
  cat("ii) For k=", k, ", s value is: ", result[[2]]$s, " with the p-value: ", result[[2]]$p, ".\n", sep
  if(result[[1]]$p < 0.05){
    rejection <- rejection + 1
    info_list_i[rejection] <- k
  }
  if(result[[2]]$p < 0.05){
```

```
    rejection <- rejection + 1
    info_list_ii[rejection] <- k
  }
  if(k==366){
    break
  }
  k <- k+1
}
```

```
## i)  For k=3, s value is: 4209.492 with the p-value: 0.117.
## ii) For k=3, s value is: 2499.619 with the p-value: 0.5885.
## i)  For k=4, s value is: 6158.586 with the p-value: 0.006.
## ii) For k=4, s value is: 3684.315 with the p-value: 0.222.
## i)  For k=5, s value is: 4226.881 with the p-value: 0.1155.
## ii) For k=5, s value is: 2874.82 with the p-value: 0.451.
## i)  For k=6, s value is: 3522.174 with the p-value: 0.267.
## ii) For k=6, s value is: 3639.421 with the p-value: 0.235.
## i)  For k=7, s value is: 2915.195 with the p-value: 0.4415.
## ii) For k=7, s value is: 1581.314 with the p-value: 0.8885.
## i)  For k=8, s value is: 1902.53 with the p-value: 0.803.
## ii) For k=8, s value is: 3600.017 with the p-value: 0.246.
## i)  For k=9, s value is: 2267.297 with the p-value: 0.673.
## ii) For k=9, s value is: 1846.164 with the p-value: 0.8195.
## i)  For k=10, s value is: 2682.184 with the p-value: 0.5165.
## ii) For k=10, s value is: 2023.097 with the p-value: 0.7565.
## i)  For k=11, s value is: 3157.592 with the p-value: 0.364.
## ii) For k=11, s value is: 2193.406 with the p-value: 0.6965.
## i)  For k=12, s value is: 4991.514 with the p-value: 0.0335.
## ii) For k=12, s value is: 1681.56 with the p-value: 0.863.
## i)  For k=13, s value is: 2911.283 with the p-value: 0.4425.
## ii) For k=13, s value is: 2152.947 with the p-value: 0.7115.
## i)  For k=14, s value is: 5478.882 with the p-value: 0.016.
## ii) For k=14, s value is: 2100.299 with the p-value: 0.7285.
## i)  For k=15, s value is: 1276.533 with the p-value: 0.9545.
## ii) For k=15, s value is: 2317.805 with the p-value: 0.6575.
## i)  For k=16, s value is: 3107.51 with the p-value: 0.3835.
## ii) For k=16, s value is: 2557.445 with the p-value: 0.5685.
## i)  For k=17, s value is: 2680.055 with the p-value: 0.517.
## ii) For k=17, s value is: 2436.111 with the p-value: 0.6155.
## i)  For k=18, s value is: 4403.065 with the p-value: 0.0885.
## ii) For k=18, s value is: 1670.696 with the p-value: 0.8665.
## i)  For k=19, s value is: 2308.084 with the p-value: 0.6605.
## ii) For k=19, s value is: 1513.308 with the p-value: 0.9045.
## i)  For k=20, s value is: 2992.825 with the p-value: 0.4185.
## ii) For k=20, s value is: 5110.624 with the p-value: 0.0285.
## i)  For k=21, s value is: 3743.872 with the p-value: 0.2075.
## ii) For k=21, s value is: 3518.916 with the p-value: 0.2685.
## i)  For k=22, s value is: 2032.779 with the p-value: 0.7515.
## ii) For k=22, s value is: 1899.675 with the p-value: 0.8045.
```

```
## i)  For k=23, s value is: 4641.737 with the p-value: 0.0605.
## ii) For k=23, s value is: 2110.417 with the p-value: 0.7265.
## i)  For k=24, s value is: 3337.915 with the p-value: 0.307.
## ii) For k=24, s value is: 3793.934 with the p-value: 0.195.
## i)  For k=25, s value is: 3528.28 with the p-value: 0.264.
## ii) For k=25, s value is: 2454.465 with the p-value: 0.6105.
## i)  For k=26, s value is: 2631.994 with the p-value: 0.5335.
## ii) For k=26, s value is: 3847.278 with the p-value: 0.187.
## i)  For k=27, s value is: 3373.841 with the p-value: 0.298.
## ii) For k=27, s value is: 1885.912 with the p-value: 0.8075.
```

From the output we got 4 rejections ($\alpha = 0.05\%$) for:

- i) at $k = 4, 12, 14$
- ii) at $k = 20$

Out of total 50 tests, we rejected the null hypothesis for 4 samples, which is around 8% of all the tests.

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.

2.1 2.1

Question

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

Answer

```
prices <- read.csv2("prices1.csv")

ggplot(prices, aes(x = SqFt, y = Price)) + geom_point() +
  theme_bw() + geom_smooth(method='lm', se = FALSE) +
  labs(x = "Square feet", y ="Price")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

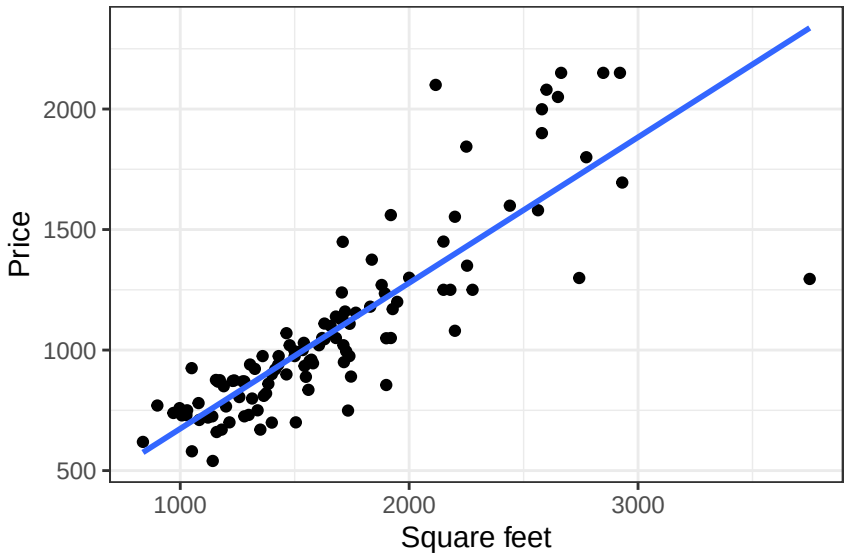

Figure 5: Scatterplot of SqFt versus Price

2.2 2.2

Question

While the data do seem to follow a linear trend, a new sort of pattern seems to appear around 2000 ft^2 Consider a new linear model

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

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, a_1, and a_2; then calculate (and return) the resulting RSS.

Answer

```
# Function that calculates RSS
RSS <- function(data, par) {
  with(data, sum((Price - (par[1] + par[2]*SqFt + par[3]*(SqFt-par[4])*ifelse(SqFt-par[4]>0, 1, 0)) )^2)
}

# Find the optimal parameters that minimized RSS
result <- optim(par = c(0, 1, 0, 2000), fn = RSS, data = prices)


T_D <- result$par[4]
# Equation of the optimal line
new_line <- function(x){
  result$par[1] + result$par[2]*x + result$par[3]*(x-result$par[4])*ifelse(x-result$par[4]>0, 1, 0)
}
```

```r
# Plots the optimal line
ggplot(prices, aes(x=SqFt, y=Price)) +
  geom_point() +
  geom_function(fun=new_line, col="steelblue", size=1) +
  theme_bw() +
  labs(x ="Square feet",
       y ="Price")
```

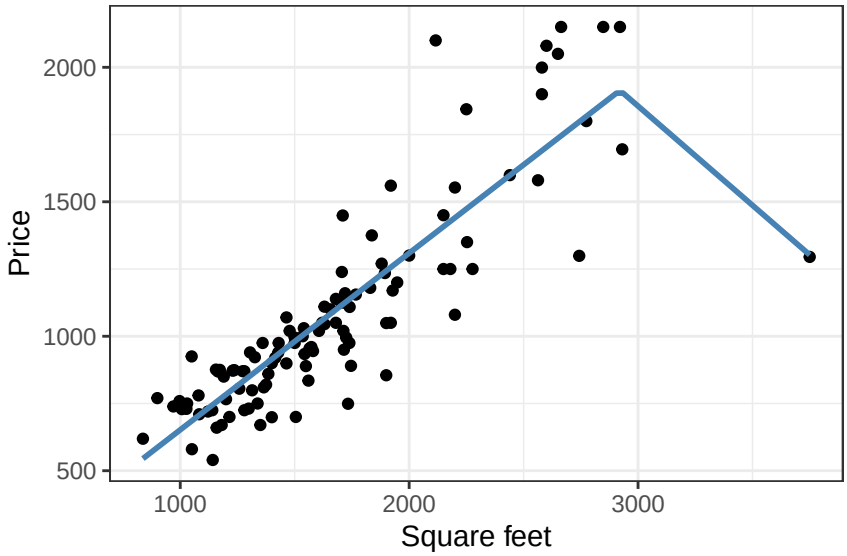

Figure 6: Scatterplot of SqFt versus Price with the new line

2.3 2.3

Question

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

Answer

```r
get_c <- function(data, vn){
  data <- as.data.frame(data[vn, ])
  result <- optim(par = c(0, 1, 0, 2000), fn = RSS, data = data)
  # c
  return(result$par[4])
}

B <- 2000
res <- boot(prices, get_c, R=B)
```

The bootstrap-bias correction of c is 3370.64.

```
# Bias corrected estimator
bias_boot <- 2*T_D  - mean(res$t)
bias_boot
```

```
## [1] 3370.64
```

The variance of c is 232963.4.

```
# Variance of c, bootstrap
var_bootstrap <- 1/(B-1)*sum((res$t - mean(res$t))^2)
var(res$t)
```

```
##          [,1]
## [1,] 232963.4
```

```
# 95% confidence interval with BCA and percentile
boot.ci(res, conf=0.95, type=c("norm", "perc", "bca"))
```

```
## 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, conf = 0.95, type = c("norm", "perc",
##     "bca"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   (2425, 4317 )   (1505, 3241 )   (2664, 3568 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```
Normal <- c(2403, 4316)
Percentile <- c(1505, 3220)
BCa <- c(2664, 3642)

table <- data.frame(rbind(Normal, Percentile, BCa))
colnames(table) <- c("Lower", "Upper")
kable(table)
```

	Lower	Upper
Normal	2403	4316
Percentile	1505	3220
BCa	2664	3642

2.4 2.4

Question

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

Answer

```
get_c <- function(data, vn){
  data <- as.data.frame(data[vn, ])
  result <- optim(par = c(0, 1, 0, 2000), fn = RSS, data = data)
  # c
  return(result$par[4])
}

results <- c()
# Jackknife data by removing one observation at a time => 110 jackknife samples
for(i in 1:nrow(prices)){
  results[i] <- get_c(prices, -i)
}

Ti_star <- 110*T_D-109*results
J_T <- mean(Ti_star)

# Variance of c, jackknife
var_jackknife <- 1/(length(results)*(length(results)-1))*sum( (Ti_star - J_T)^2)


var_jackknife
```

```
## [1] 3962738
```

```
var_bootstrap
```

```
## [1] 232963.4
```

From the output the variance for c from jackknife is 3 962 738 and for bootstrap the estimate is around 232 963. From literature the variance from jackknife is expected to be larger.

2.5 2.5

Question

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.

Answer

The confidence interval for Normal is widest and BCa is most narrow. The value of c is around 2921 and all the confidence intervals covers c. Normal and BCa covers the value c near the lower end of the confidence interval. Percentile covers c near the upper end of the confidence interval.

3 Statement of Contribution

We solved the tasks and wrote the report as a group.

4 Appendix

The code used in this laboration report are summarised in the code as follows:

```
library(knitr)
library(dplyr)
library(ggplot2)
library(boot)


knitr::opts_chunk$set(
  echo = TRUE,
  fig.width = 4.5,
  fig.height = 3)

lottery <- read.csv2("lottery.csv", stringsAsFactors = TRUE)

ggplot(lottery, aes(x = Day_of_year, y = Draft_No)) + geom_point() +
  theme_bw() + labs(x = "Day of year",y = "Draft number")



ggplot(lottery, aes(x = Day_of_year, y = Draft_No)) +
  geom_point() +
  theme_bw() +
  labs(x = "Day of year", y = "Draft number") +
  stat_smooth(method = lm, se = FALSE, aes(color = "Linear\nRegression")) +
  stat_smooth(method = loess, se = FALSE, aes(color = "Loess")) +
  scale_color_manual(values = c("indianred", "steelblue"),
                     name = "Method",
                     labels = c("Linear\nRegression", "Loess"))
```

```r
mod_loess <- loess(Draft_No ~ Day_of_year, data=lottery)
T_X <- sum(abs(mod_loess$fitted - mean(lottery$Draft_No)))


# Y-bar
mean_y <- mean(lottery$Draft_No)

# Function to used for bootstrap
stat1 <- function(data, vn){
  data <- as.data.frame(data[vn, ])
  fit_mod <- loess(Draft_No ~ Day_of_year, data=data)
  s <- sum(abs(fit_mod$fitted - mean_y))
  return(s)
}

set.seed(13)
# Number of bootstrap samples
B <- 2000

res <- boot(lottery, stat1, R=B)
res$t0
plot_data <- data.frame(x=res$t)

ggplot(plot_data) +
  geom_histogram(aes(x=x),fill="steelblue", color="black") +
  theme_bw() +
  labs(x="S")
# Simulating the null distribution with permutation test.
set.seed(13)
index <- list()
# Creates 2000 permutations.
for(iter in 1:2000){
  new_data_index <- list(sample(1:366, 366))
  index[[iter]] <- new_data_index
}

# Ensures that the all the permutations are unique
index <- unique(index)

mean_y <- mean(lottery$Draft_No)
get_s <- function(index){
  temp_data <- lottery
  temp_data$Draft_No <- index
  fit_mod <- loess(Draft_No ~ Day_of_year, data=temp_data)
  s <- sum(abs(fit_mod$fitted - mean_y))
  return(s)
}
```

```r
null_s <- c()
for(iter in 1:2000){
  null_s[iter] <- get_s(unlist(index[[iter]]))
}

plot_data <- data.frame(x=null_s)
ggplot(plot_data) +
  geom_histogram(aes(x=x),fill="indianred", color="black") +
  theme_bw() +
  labs(x="S")
B <- 2000
p <- sum(null_s > T_X)/B
p
# Function that runs 2000 bootstrap samples and returns S and p-value under null hypothesis
mean_y <- mean(lottery$Draft_No)
get_boot <- function(data){
  # Calculates value of S for bootstrap samples
  stat2 <- function(data, vn){
    data <- as.data.frame(data[vn, ])
    fit_mod <- loess(Draft_No ~ Day_of_year, data=data)
    s <- sum(abs(fit_mod$fitted - mean_y))
    return(s)
  }
  # Bootstrap sampling
  B <- 2000
  res <- boot(data, stat2, R=B)

  # Test statistics
  s <- res$t0
  p <- sum(null_s > s)/B
  result <- list(s=s, p=p)
  return(result)
}
# Function that does bootstrap sampling for both i) and ii)
bias_data <- function(k){
  # Data for i.
  data_i <- lottery
  data_i$Day_of_year[1:k] <- (366-k+1):366
  data_i$Day_of_year[(k+1):366] <- sample(1:(366-k), (366-k))

  # Data for ii.
  data_ii <- lottery
  # Split the data into chunks
  data_chunk <- split(x, sort(rep_len(1:k, length(x))))
  # Randomise order of chunks
  blocks <- sample(1:k, k)

  # Chose randomize block 1 to change values for
```

```
  index <- unname(unlist(data_chunk[blocks[[1]]]))
  data_ii$Day_of_year[index] <- (366-(length(index))+1):366
  data_ii$Day_of_year[-index] <- sample(1:(366-(length(index))), (366-(length(index))))

  result_i <- get_boot(data_i)
  result_ii <- get_boot(data_ii)
  result <- list(result_i, result_ii)
  return(result)
}


x <- 1:366
k <- 3
rejection <- 0
info_list_i <- list()
info_list_ii <- list()
while(k <= 27){
  result <- bias_data(k)
  cat("i)  For k=", k, ", s value is: ", result[[1]]$s, " with the p-value: ", result[[1]]$p, ".\n", sep
  cat("ii) For k=", k, ", s value is: ", result[[2]]$s, " with the p-value: ", result[[2]]$p, ".\n", sep
  if(result[[1]]$p < 0.05){
    rejection <- rejection + 1
    info_list_i[rejection] <- k
  }
  if(result[[2]]$p < 0.05){
    rejection <- rejection + 1
    info_list_ii[rejection] <- k
  }
  if(k==366){
    break
  }
  k <- k+1
}

prices <- read.csv2("prices1.csv")

ggplot(prices, aes(x = SqFt, y = Price)) + geom_point() +
  theme_bw() + geom_smooth(method='lm', se = FALSE) +
  labs(x = "Square feet", y ="Price")



# Function that calculates RSS
RSS <- function(data, par) {
  with(data, sum((Price - (par[1] + par[2]*SqFt + par[3]*(SqFt-par[4])*ifelse(SqFt-par[4]>0, 1, 0)) )^2)
}

# Find the optimal parameters that minimized RSS
result <- optim(par = c(0, 1, 0, 2000), fn = RSS, data = prices)
```

```r
T_D <- result$par[4]
# Equation of the optimal line
new_line <- function(x){
  result$par[1] + result$par[2]*x + result$par[3]*(x-result$par[4])*ifelse(x-result$par[4]>0, 1, 0)
}

# Plots the optimal line
ggplot(prices, aes(x=SqFt, y=Price)) +
  geom_point() +
  geom_function(fun=new_line, col="steelblue", size=1) +
  theme_bw() +
  labs(x ="Square feet",
       y ="Price")
get_c <- function(data, vn){
  data <- as.data.frame(data[vn, ])
  result <- optim(par = c(0, 1, 0, 2000), fn = RSS, data = data)
  # c
  return(result$par[4])
}

B <- 2000
res <- boot(prices, get_c, R=B)
# Bias corrected estimator
bias_boot <- 2*T_D  - mean(res$t)
bias_boot
# Variance of c, bootstrap
var_bootstrap <- 1/(B-1)*sum((res$t - mean(res$t))^2)
var(res$t)
# 95% confidence interval with BCA and percentile
boot.ci(res, conf=0.95, type=c("norm", "perc", "bca"))

Normal <- c(2403, 4316)
Percentile <- c(1505, 3220)
BCa <- c(2664, 3642)

table <- data.frame(rbind(Normal, Percentile, BCa))
colnames(table) <- c("Lower", "Upper")
kable(table)


get_c <- function(data, vn){
  data <- as.data.frame(data[vn, ])
  result <- optim(par = c(0, 1, 0, 2000), fn = RSS, data = data)
  # c
  return(result$par[4])
}
```

```r
results <- c()
# Jackknife data by removing one observation at a time => 110 jackknife samples
for(i in 1:nrow(prices)){
  results[i] <- get_c(prices, -i)
}

Ti_star <- 110*T_D-109*results
J_T <- mean(Ti_star)

# Variance of c, jackknife
var_jackknife <- 1/(length(results)*(length(results)-1))*sum( (Ti_star - J_T)^2)


var_jackknife
var_bootstrap
```