CS Computer Lab 5

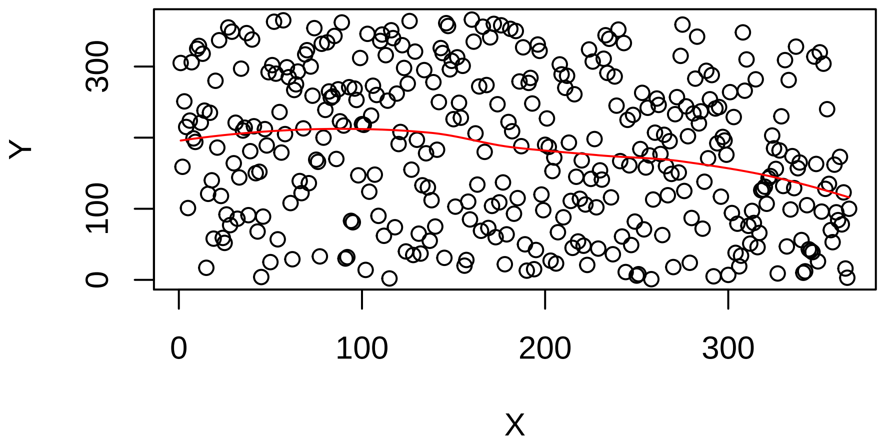
Joshua Hudson, Carles Sans

4 March 2017

Question 1: Hypothesis testing

In this section, we used the data set **lottery.csv** containing US military draft information, notably the Draft Numbers (our Y variable) of the men selected at random and the Day of the Year (our X variable) they were born.

We started by plotting X vs Y, as shown in the figure below.



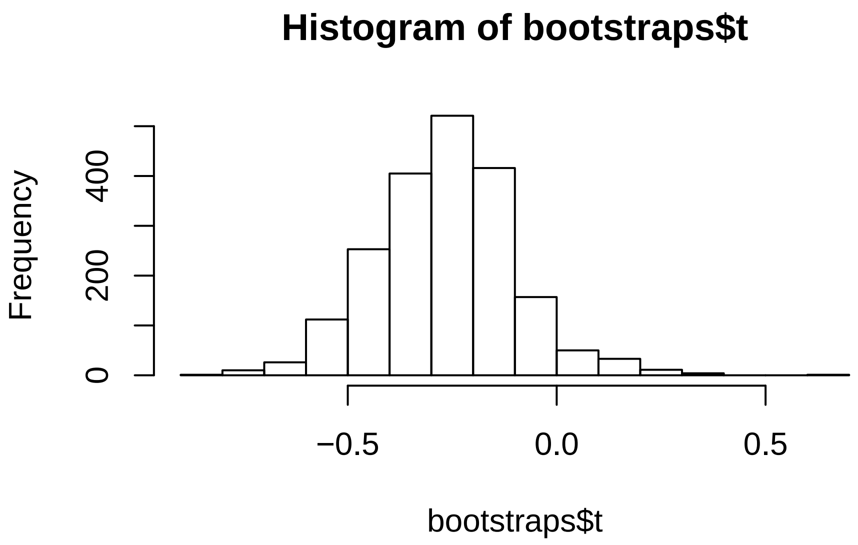
Judging from the scatterplot, it would seem that the lottery is random. We also computed the loess smoother estimate of Y using the `loess()` function and adding the curve of these fitted values to the figure.

To truly check whether the lottery was random, we used the following test statistic:

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

where $X_b = argmax_X Y(X), X_a = argmin_X Y(X)$

If this statistic is significantly larger than 0, it indicates that the lottery is not random. First we used a non-parametric bootstrap with B=2000 to estimate the distribution of T. The results are shown into the histogram of the T statistic for each bootstrap sample.



From the figure we could estimate that the distribution of T is normal, judging by the shape of the histogram. The values of T are close to zero and high values of T get less and less likely as it follows a normal distribution centered around -0.2642089; therefore there is no evidence to suggest that the lottery is not random based on this test statistic. If the hypothesis tested here is that T is larger than 0 then the p-value of the test was 0.0495, so very close to 0.05: testing with different seeds before the bootstrapping function showed that the p-value fluctuated between above and below 0.05, which is the criteria for hypotheses rejection usually taken.

Next we implemented a function that tests the hypothesis H0: that the lottery is random vs H1: the lottery is non-random, by using a permutation test with T. The output of our function for B=2000 is displayed below: it shows the value of the test statistic for the original data followed by the p-value.

```
## [1] -0.2671794  0.0885000
```

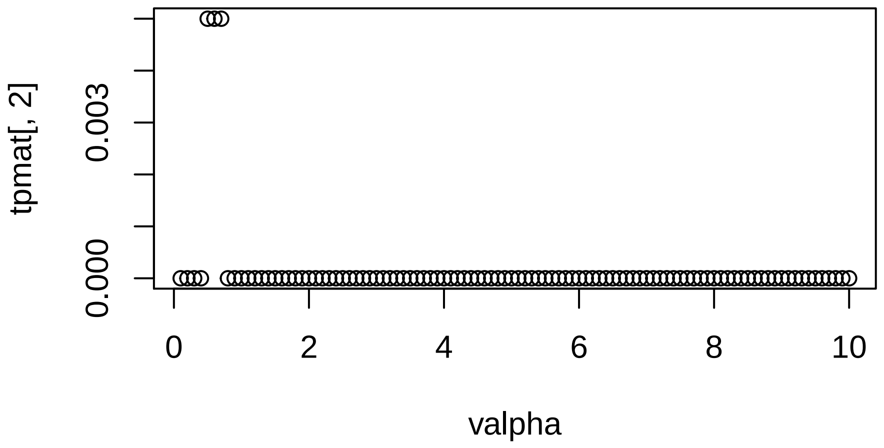
The p-value is 0.0885 so we cannot reject the null hypothesis H0 that the lottery is random.

Finally, we wanted to estimate the power of this test. To do this, we created an alternative, non-random, data set where Y is non-random and depends on X using the following function:

$$Y(x) = max(0, min(\alpha x + \beta, 366))$$

where $\alpha = 0.1$ and $\beta \sim N(183, sd = 10)$.

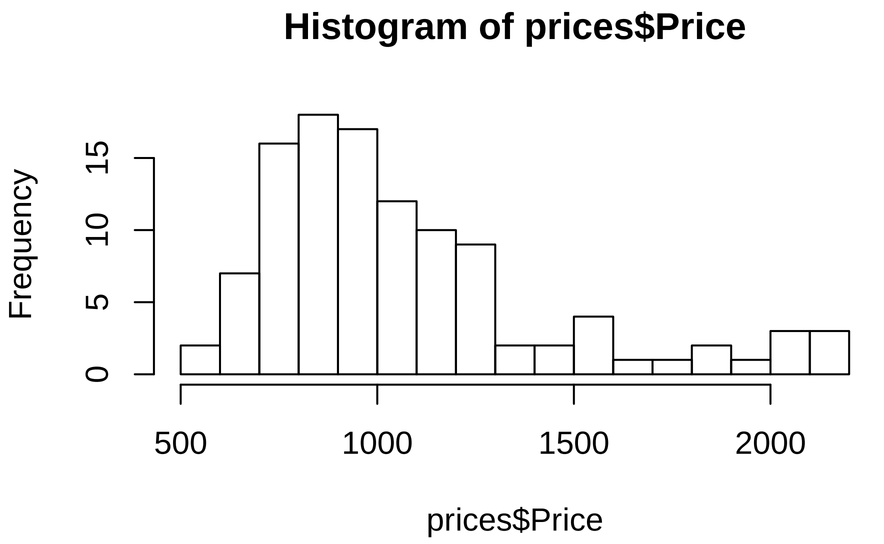
The p-value found was 0 so the null hypothesis (that the lottery was random) was rejected. We then tested out other values of $\alpha = 0.2, 0.3, ..., 10$ and repeated the permutation test for each generated data set. The plot of the p-values for each alpha is shown.



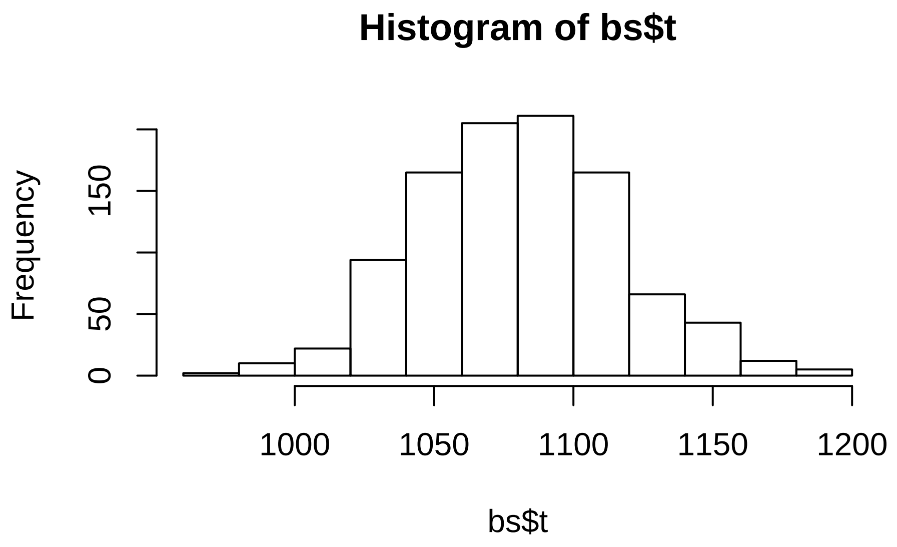
The plot shows that all the p-values were below 0.05 so for all 100 datasets, the null hypothesis was rejected. As we generated these data sets purposely non-randomly, we know H1 to be true. Therefore there were no type II errors and the power of our terst was estimated to be 1.

Question 2: Bootstrap, jackknife, confidence intervals

In this section, we used the data set **prices1.csv**, containing information on home prices in Aulberque, 1993. We just dealt with the `Price` variable here and started by plotting its histogram.



The histogram shape would indicate that price follows a Gamma distribution. The mean price was 1080.4727273. Next we estimated the distribution of the mean price using 2000 bootstrap samples. The histogram of the mean price for the bootstrap samples is shown below.



The histogram shows that mean price seems to follow a normal distribution.

We calculated the bootstrap bias-correction: 1080.2203182 and the variance of the mean price: 1290.8087355. We then used `boot.ci()` function to obtain the 95% confidence intervals for the mean price using first order normal approximation, bootstrap percentile and bootstrap BCa respectively:

```
#Confidence intervals
boot.ci(bs, type=c("norm", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bs, type = c("norm", "perc", "bca"))
##
## Intervals :
## Level      Normal              Percentile            BCa
## 95%   (1010, 1151 )   (1013, 1155 )   (1017, 1158 )
## Calculations and Intervals on Original Scale
```

Next we estimate the variance of the mean price using the jackknife method with B=n(=110)

We obtained an estimate of 1320.9110441 for the variance, so slightly above the estimate we obtained using bootstrap. This seems usual as the jackknife does tend to over-estimate the variance.

Finally we found the 95% confidence interval for the mean price with jackknife.

```
## [1] 1009.239 1151.706
```

Looking at the 4 confidence intervals we obtained, the lengths were very close: 141, 142, 141 and 143 for the normal, percentile, BCa and jackknife normal confidence intervals respectively. As for the position of the mean in each interval, it is as close to the middle as it could be for the normal approximation in both bootstrap and jackknife methods. The mean was slightly closer to the lower bound for the percentile and even closer yet to the lower bound for the BCa. Given that all confidence intervals are almost the same, this means that our mean price is almost normally distributed, and thus the intervals coincide on bounds.

Appendix

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, fig.width = 5, fig.asp = 0.66, fig.show = "asis", 
#1.1
setwd("~/Semester2/CS/Lab5")
lott <- read.csv2(paste0(getwd(),"/lottery.csv"))

Y <- lott$Draft_No
X <- lott$Day_of_year

#1.2
fit <- loess(Draft_No ~ Day_of_year, data = lott)
Yhat <- predict(fit, newdata = lott)
plot(X, Y)
lines(X, Yhat, col="red")
#1.3
library(boot)
data <- data.frame(X=lott$Day_of_year, Y=lott$Draft_No)

stat1 <- function(data, vn) {
  data <- as.data.frame(data[vn, ])
  fit <- loess(Y ~ X, data = data)
  Xb <- data$X[which.max(data$Y)]
  Xa <- data$X[which.min(data$Y)]
  Yhatb <- predict(fit, newdata=Xb)
  Yhata <- predict(fit, newdata=Xa)
  teststat <- (Yhatb-Yhata)/(Xb-Xa)
  return(teststat)
}

set.seed(12345)
bootstraps <- boot(data = data, statistic = stat1, R = 2000)
hist(bootstraps$t)
tmean <- mean(bootstraps$t)
tsd <- sd(bootstraps$t)
pvalue <- mean(bootstraps$t > 0)
#1.4
Htest <- function(data, B) {
  n <- dim(data)[1]
  stat0 <- stat1(data = data, vn=1:n)
  stat <- numeric(B)

  set.seed(12345)
  for (b in 1:B) {
    Gb <- sample(1:n, n)
    data$Y <- data$Y[Gb]
    stat[b] <- stat1(data = data, vn=1:n)
  }

  pvalue <- mean(abs(stat) >= abs(stat0))
  return(c(stat0, pvalue))
}
```

```
Htest(data, 2000)
#1.5
#a
alpha <- 0.1
set.seed(12345)
beta <- rnorm(1, mean = 183, sd = 10)

Y2 <- sapply(X, FUN= function(x) {

  max(0, min(alpha*x+beta, 366))
})
data2 <- data.frame(X=X, Y=Y2)
#b
res5a <- Htest(data2, 200)


#c (includes 0.1)
valpha <- seq(0.1, 10, 0.1)
tpmat <- matrix(0, ncol=2, nrow = length(valpha))
for (i in 1:length(valpha)) {
  alpha <- valpha[i]
  set.seed(12345)
  beta <- rnorm(1, mean = 183, sd = 10)
  Y3 <- sapply(X, FUN= function(x) {
    max(0, min(alpha*x+beta, 366))
  })
  data3 <- data.frame(X=X, Y=Y3)
  tpmat[i, ] <- Htest(data3, 200)
}

plot(valpha, tpmat[, 2])

#2.1
setwd("~/Semester2/CS/Lab5")
prices <- read.csv2(paste0(getwd(),"/prices1.csv"))

hist(prices$Price, breaks=seq(500, 2200, 100))
#gamma?
meanprice <- mean(prices$Price)
#2.2
library(boot)
stat2 <- function(data, vn) {
  data <- as.data.frame(data[vn, ])
  stat <- mean(data$Price)
}
set.seed(12345)
bs <- boot(data=prices, statistic = stat2, 1000)
#distribution of mean price
hist(bs$t) #normal?
#bootstrap bias corrected estimator
biasco <- 2*bs$t0 - mean(bs$t)
#variance of T = mean price
bootvar <- 1/999*sum((bs$t-mean(bs$t))^2)
```

```r
#Confidence intervals
boot.ci(bs, type=c("norm", "perc", "bca"))

#2.3
jackknife <- function(data, B) {
  stat <- vector(length = B)
  for (i in 1:B) {
    data2 <- data[-i, ]
    n <- dim(data2)[1]
    stat[i] <- stat2(data=data2, vn=1:n)
  }
  return(stat)
}
n <- dim(prices)[1]
jackstat <- jackknife(data = prices, B = n)

#variance calc
Tstar <- n*mean(prices$Price) -(n-1)*jackstat
JT <- mean(Tstar)
jackvar <- 1/(n*(n-1))*sum((Tstar-JT)^2)

z <- qnorm(0.975, 0, 1, (n-1))
jackci <- c(meanprice-z*sqrt(jackvar), meanprice+z*sqrt(jackvar))
print(jackci)
```

Contributions

Our group did this report in conjunction with Group 6 and so will be submitting the same report.