STAT 206 Homework 7_Lihua_Xu

Due Monday, November 20, 5:00 PM

General instructions for homework: Homework must be completed as an R Markdown file. Be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

Part I - Beverton-Holt model

The dataset at [http://www.faculty.ucr.edu/~jflegal/fish.txt] contains 40 annual counts of the numbers of spawners `S` and recruits `R` in a salmon population. The units are thousands of fish. Spawners are fish that are laying eggs. Spawners die after laying eggs. Recruits are fish that enter the catchable population.

The classic **Beverton-Holt** model for the relationship between spawners and recruits is

$$R = \frac{1}{\beta_1 + \beta_2/S}\,, \quad \beta_1 > 0, \beta_2 > 0$$

where R and S are the number of recruits and spawners respectively.

Consider the problem of maintaining a sustainable fishery. The total population abundance will only stabilize if $R = S$. The total population will decline if fewer recruits are produced than the number of spawners who died producing them. If too many recruits are produced, the population will also decline eventually because there is not enough food for them all. Thus, only a balanced level of recruits can be sustained indefinitely in a stable population. This stable population level is the point where the 45° line intersects the curve relating R and S. In other words, it is the N such that

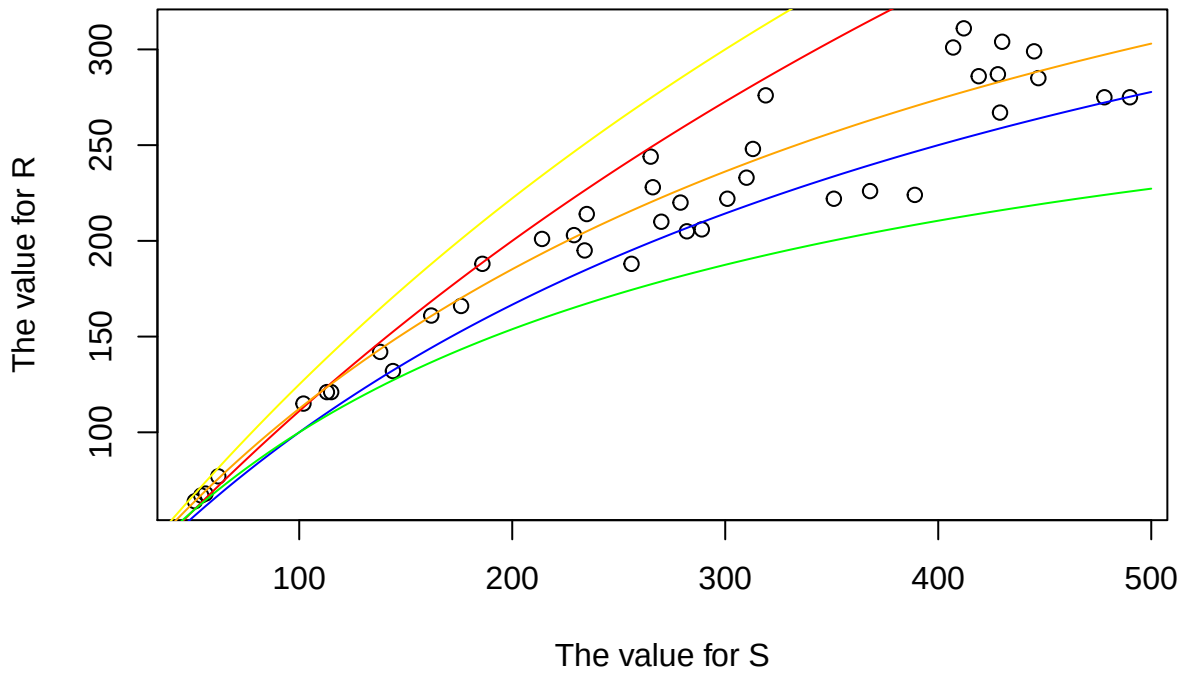
$$N = \frac{1}{\beta_1 + \beta_2/N}.$$

Solving for N we see that the stable population level is $N = (1 - \beta_2)/\beta_1$.

1. Make a scatterplot of the data and overlay the Beverton-Holt curve for a couple different choices of β_1 and β_2.

```
loading_data <- read.table("http://www.faculty.ucr.edu/~jflegal/fish.txt",head=TRUE)
plot(loading_data$S,loading_data$R,xlab="The value for S", ylab="The value for R")

#Beta_1=0.002,Beta_2=0.8
curve(1/(0.002+0.8/x),from=1,to=500,add=TRUE,type="l",col="blue")
#Beta_1=0.001,Beta_2=0.8
curve(1/(0.001+0.8/x),from=1,to=500,add=TRUE,type="l",col="red")
#Beta_1=0.001,Beta_2=0.7
curve(1/(0.001+0.7/x),from=1,to=500,add=TRUE,type="l",col="yellow")
#Beta_1=0.003,Beta_2=0.7
curve(1/(0.003+0.7/x),from=1,to=500,add=TRUE,type="l",col="green")
#Beta_1=0.0019,Beta_2=0.7
curve(1/(0.0019+0.7/x),from=1,to=500,add=TRUE,type="l",col="orange")
```

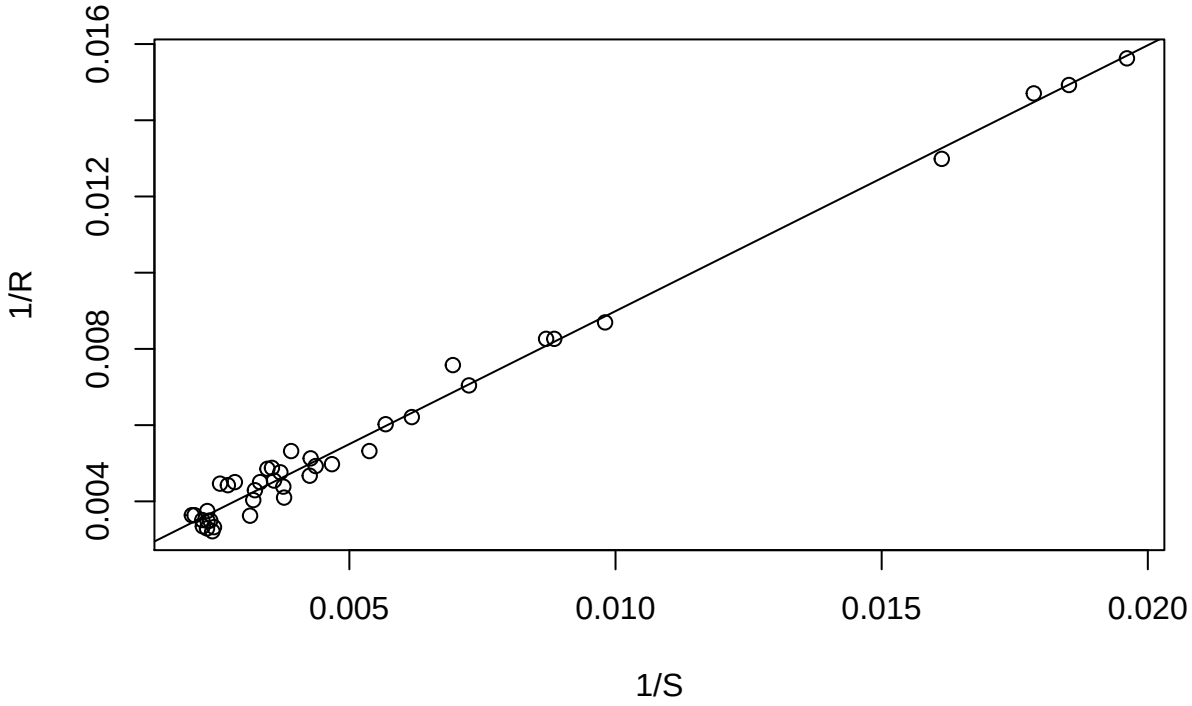


2. The Beverton-Holt model can be found by transforming $R \mapsto (1/R)$ and and $S \mapsto (1/S)$. That is,

$$(1/R) = \beta_1 + \beta_2(1/S).$$

This is a linear model with response variable $(1/R)$ and covariate $(1/S)$. Use least squares regression to fit this model to the fish dataset.

```
plot(1/loading_data$S,1/loading_data$R,xlab="1/S", ylab="1/R")
Y <- 1/loading_data$R
X <- 1/loading_data$S
fit <- lm(Y~X)
abline(fit)
```

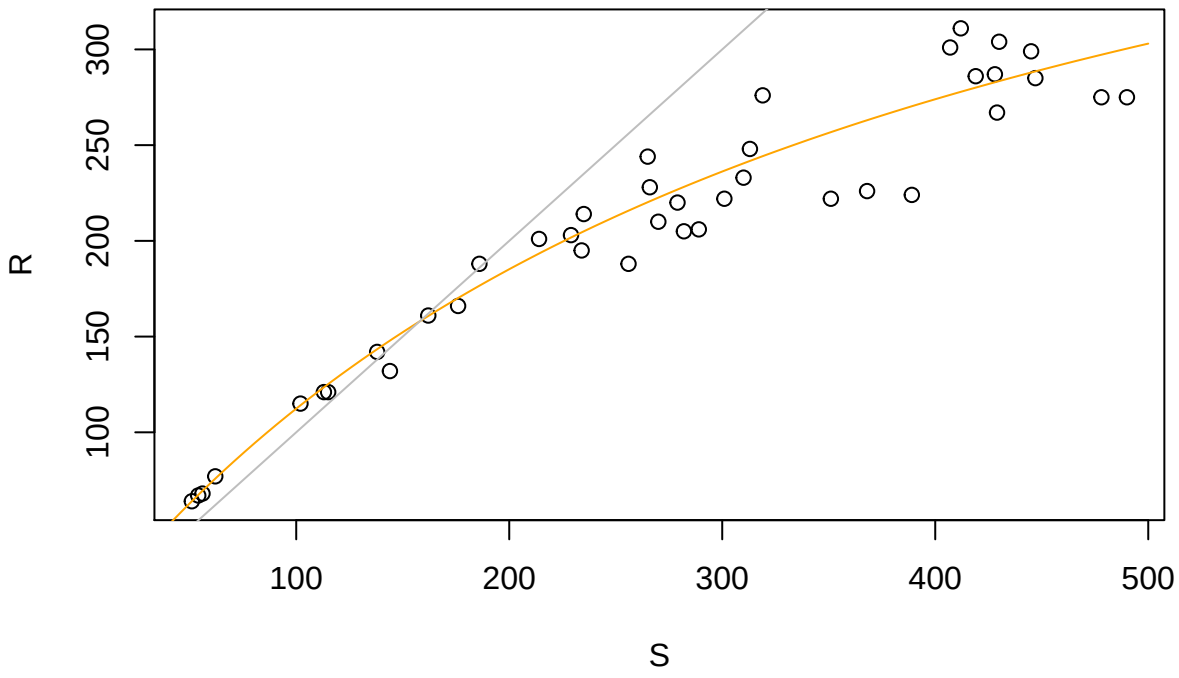



```
#The intercept and the slop should be as following
fit
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Coefficients:
## (Intercept)            X
##    0.002013     0.697819
```

3. Find an estimate for the stable population level, where $R = S$ in the Beverton-Holt model.

```
#As an estimate, from the Prob 1, we can see when Beta_1=0.0019,Beta_2=0.8,
#it gives a "good" estimate so under this situation:
#We need to the point where the 45 line intersects the fitting curve.
plot(loading_data$S,loading_data$R,xlab="S", ylab="R")
curve(1/(0.0019+0.7/x),from=1,to=500,add=TRUE,type="l",col="orange")
abline(a=0,b=1,col="grey")
```




```
N <- round((1-0.7)/0.0019)
N
```

```
## [1] 158
```

```
#From the plot, we could see the intercept is around the value 158.
#158 is my estimate for the stable population level.
```

4. Use the bootstrap to obtain the sampling distribution and standard error for the stable population level. Use the bootstrap to construct a 95% confidence interval for the stable population level.

```
library(boot)
Prob4 <- function(formula,data,indices){
  d <- data[indices,]
  fit <- lm(formula, data=d)
  return(summary(fit)$r.square)
}
data <- 1/loading_data
results <- boot(data, statistic=Prob4, R=1000, formula=(R~S))
results
```

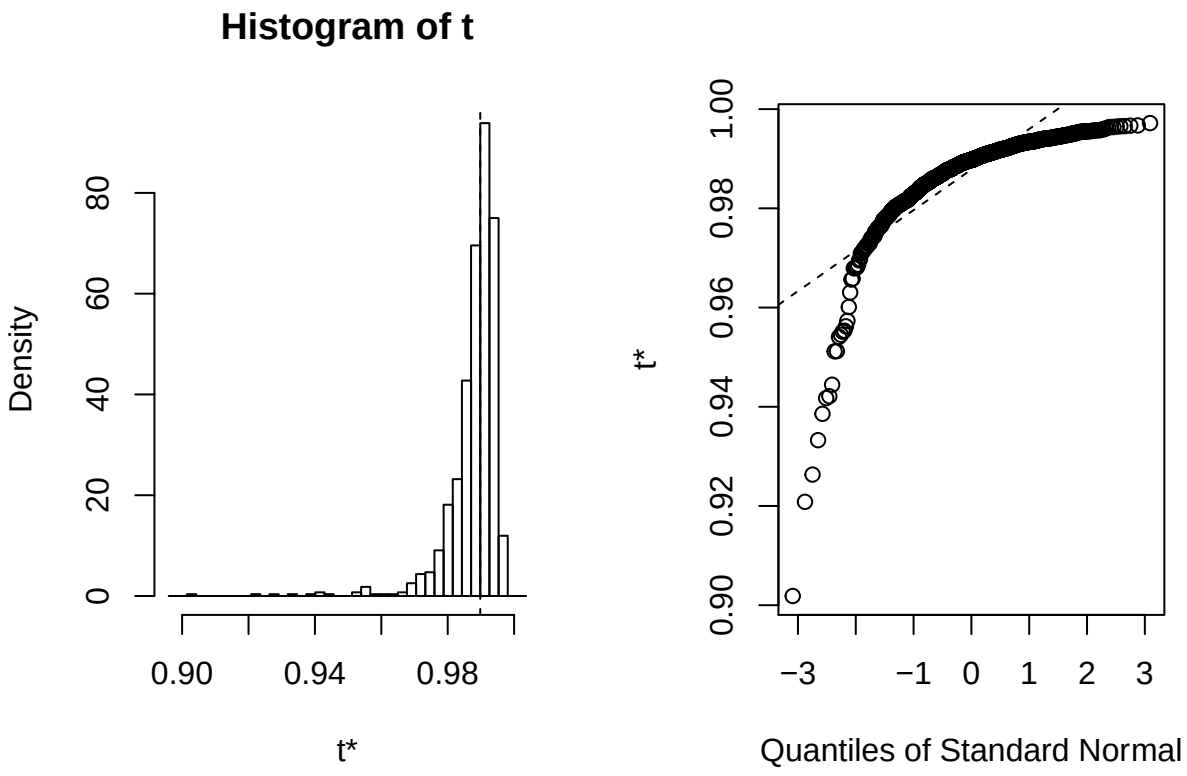
```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data, statistic = Prob4, R = 1000, formula = (R ~
##     S))
```

```
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.9897986 -0.001980879 0.008169258
```

```
boot.ci(results, type="bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.9729,  0.9957 )
## Calculations and Intervals on Original Scale
```

```
plot(results)
```




```
#The standard error 0.008592133.
#The 95% confidence interval for the stable population level is ( 0.9713,  0.9959 )
#The sampling distribution is as following plot.
```

Part II - Snowfall accumulations

The data set `buffalo` at [http://www.faculty.ucr.edu/~jflegal/buffalo.txt] contains annual snowfall accumulations in Buffalo, NY from 1910 to 1973.

5. Construct kernel density estimates of the data using the Gaussian and Epanechnikov kernels.

```
loading <- read.table("http://www.faculty.ucr.edu/~jflegal/buffalo.txt")
loading <- as.list(loading)
#Gaussian kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("epanechnikov"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("epanechnikov"),     weights = N
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004571
##  1st Qu.: 45.69   1st Qu.:0.0039309
##  Median : 75.70   Median :0.0080912
##  Mean   : 75.70   Mean   :0.0081404
##  3rd Qu.:105.71   3rd Qu.:0.0122746
##  Max.   :135.72   Max.   :0.0156971
```

```
#Epanechnikov kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("gaussian"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("gaussian"),     weights = NULL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004882
##  1st Qu.: 45.69   1st Qu.:0.0038465
##  Median : 75.70   Median :0.0084991
##  Mean   : 75.70   Mean   :0.0081412
##  3rd Qu.:105.71   3rd Qu.:0.0115233
##  Max.   :135.72   Max.   :0.0165421
```

6. Compare the estimates for different choices of bandwidth.

```
#Epanechnikov kernel
#width=10
density(loading$V1, bw = "nrd0", adjust = 10,
        kernel = c("epanechnikov"),
```

```
        weights = NULL, window = kernel, width=2,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 10, kernel = c("epanechnikov"),     weights =
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 93.21
##
##        x                 y
##  Min.   :-68.215   Min.   :0.001729
##  1st Qu.:  3.743   1st Qu.:0.002568
##  Median : 75.700   Median :0.003097
##  Mean   : 75.700   Mean   :0.002962
##  3rd Qu.:147.657   3rd Qu.:0.003438
##  Max.   :219.615   Max.   :0.003552
```

```
#width=20
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("epanechnikov"),
        weights = NULL, window = kernel, width=20,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("epanechnikov"),     weights = l
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004571
##  1st Qu.: 45.69   1st Qu.:0.0039309
##  Median : 75.70   Median :0.0080912
##  Mean   : 75.70   Mean   :0.0081404
##  3rd Qu.:105.71   3rd Qu.:0.0122746
##  Max.   :135.72   Max.   :0.0156971
```

```
#width=50
density(loading$V1, bw = "nrd0", adjust = 50,
        kernel = c("epanechnikov"),
        weights = NULL, window = kernel, width=10,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 50, kernel = c("epanechnikov"),     weights =
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 466.1
##
##        x                y
##  Min.   :-441.1   Min.   :0.0005395
##  1st Qu.:-182.7   1st Qu.:0.0006178
##  Median :  75.7   Median :0.0006738
```

```
##  Mean   :  75.7   Mean   :0.0006585
##  3rd Qu.: 334.1   3rd Qu.:0.0007075
##  Max.   : 592.5   Max.   :0.0007192
```

```
#Gaussian
#width=10
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("gaussian"),
        weights = NULL, window = kernel, width=10,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("gaussian"),     weights = NULL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004882
##  1st Qu.: 45.69   1st Qu.:0.0038465
##  Median : 75.70   Median :0.0084991
##  Mean   : 75.70   Mean   :0.0081412
##  3rd Qu.:105.71   3rd Qu.:0.0115233
##  Max.   :135.72   Max.   :0.0165421
```

```
#width=20
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("gaussian"),
        weights = NULL, window = kernel, width=20,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("gaussian"),     weights = NULL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004882
##  1st Qu.: 45.69   1st Qu.:0.0038465
##  Median : 75.70   Median :0.0084991
##  Mean   : 75.70   Mean   :0.0081412
##  3rd Qu.:105.71   3rd Qu.:0.0115233
##  Max.   :135.72   Max.   :0.0165421
```

```
#width=50
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("gaussian"),
        weights = NULL, window = kernel, width=50,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("gaussian"),     weights = NULL
```

```
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004882
##  1st Qu.: 45.69   1st Qu.:0.0038465
##  Median : 75.70   Median :0.0084991
##  Mean   : 75.70   Mean   :0.0081412
##  3rd Qu.:105.71   3rd Qu.:0.0115233
##  Max.   :135.72   Max.   :0.0165421
```

```
#Different choices of bandwidth will influence "Epanechnikov" method a lot.
#Different choices of bandwidth will not influence "Gaussian" method a lot.
```

7. Is the estimate more influenced by the type of kernel or the bandwidth?

```
#Gaussian kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("epanechnikov"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("epanechnikov"),     weights = N
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004571
##  1st Qu.: 45.69   1st Qu.:0.0039309
##  Median : 75.70   Median :0.0080912
##  Mean   : 75.70   Mean   :0.0081404
##  3rd Qu.:105.71   3rd Qu.:0.0122746
##  Max.   :135.72   Max.   :0.0156971
```

```
#Epanechnikov kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("gaussian"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("gaussian"),     weights = NULL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004882
##  1st Qu.: 45.69   1st Qu.:0.0038465
##  Median : 75.70   Median :0.0084991
##  Mean   : 75.70   Mean   :0.0081412
##  3rd Qu.:105.71   3rd Qu.:0.0115233
```

```
##  Max.   :135.72   Max.   :0.0165421
```

```
#rectangular kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("rectangular"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("rectangular"),     weights = NU
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004916
##  1st Qu.: 45.69   1st Qu.:0.0041203
##  Median : 75.70   Median :0.0078651
##  Mean   : 75.70   Mean   :0.0081585
##  3rd Qu.:105.71   3rd Qu.:0.0129893
##  Max.   :135.72   Max.   :0.0157302
```

```
#triangular kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("triangular"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("triangular"),     weights = NUL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y
##  Min.   : 15.68   Min.   :0.0004117
##  1st Qu.: 45.69   1st Qu.:0.0039334
##  Median : 75.70   Median :0.0083618
##  Mean   : 75.70   Mean   :0.0081408
##  3rd Qu.:105.71   3rd Qu.:0.0118010
##  Max.   :135.72   Max.   :0.0160771
```

```
#biweight kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("biweight"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("biweight"),     weights = NULL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
```

```
##        x                y          
##  Min.   : 15.68   Min.   :0.0004463  
##  1st Qu.: 45.69   1st Qu.:0.0039257  
##  Median : 75.70   Median :0.0082776  
##  Mean   : 75.70   Mean   :0.0081412  
##  3rd Qu.:105.71   3rd Qu.:0.0120005  
##  Max.   :135.72   Max.   :0.0159336
```

```
#cosine kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("cosine"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("cosine"),     weights = NULL, w
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y          
##  Min.   : 15.68   Min.   :0.0004546  
##  1st Qu.: 45.69   1st Qu.:0.0039179  
##  Median : 75.70   Median :0.0083123  
##  Mean   : 75.70   Mean   :0.0081414  
##  3rd Qu.:105.71   3rd Qu.:0.0119191  
##  Max.   :135.72   Max.   :0.0160290
```

```
#optcosine kernel
density(loading$V1, bw = "nrd0", adjust = 1,
        kernel = c("optcosine"),
        weights = NULL, window = kernel, width=1,
        give.Rkern = FALSE,
        n = 63, cut = 1, na.rm = FALSE)
```

```
##
## Call:
##  density.default(x = loading$V1, bw = "nrd0", adjust = 1, kernel = c("optcosine"),     weights = NULL
##
## Data: loading$V1 (63 obs.);  Bandwidth 'bw' = 9.321
##
##        x                y          
##  Min.   : 15.68   Min.   :0.0004515  
##  1st Qu.: 45.69   1st Qu.:0.0039317  
##  Median : 75.70   Median :0.0081590  
##  Mean   : 75.70   Mean   :0.0081403  
##  3rd Qu.:105.71   3rd Qu.:0.0121965  
##  Max.   :135.72   Max.   :0.0157509
```

```
#I have listed all the method for calculating the estimate.
#The estimate is influenced a lot by the type of kernel.
#Based on the different width I used in previous caiculations on the estimates.
#The estimate is more infoluenced by the bandwidth.
```