Statistical Calculation and Software

Assignment 3

Hanbin Liu 11912410

3.1

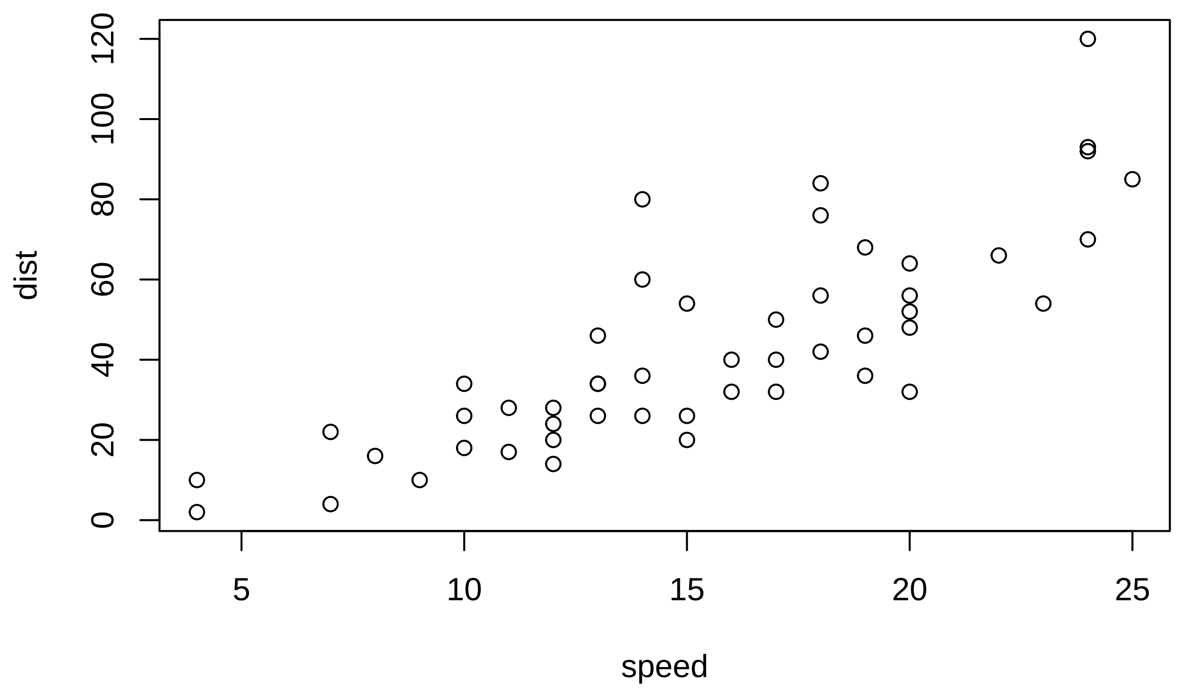
```r
data(cars)
any(is.na(cars))
```

```
## [1] FALSE
```

(a)

```r
attach(cars)
```

```r
plot(speed, dist)
```



```r
nw_box <- ksmooth(speed, dist, kernel = 'box', bandwidth = 1)
nw_box
```

```
## $x
##   [1]  4.000000  4.212121  4.424242  4.636364  4.848485  5.060606  5.272727
##   [8]  5.484848  5.696970  5.909091  6.121212  6.333333  6.545455  6.757576
##  [15]  6.969697  7.181818  7.393939  7.606061  7.818182  8.030303  8.242424
##  [22]  8.454545  8.666667  8.878788  9.090909  9.303030  9.515152  9.727273
##  [29]  9.939394 10.151515 10.363636 10.575758 10.787879 11.000000 11.212121
##  [36] 11.424242 11.636364 11.848485 12.060606 12.272727 12.484848 12.696970
##  [43] 12.909091 13.121212 13.333333 13.545455 13.757576 13.969697 14.181818
##  [50] 14.393939 14.606061 14.818182 15.030303 15.242424 15.454545 15.666667
##  [57] 15.878788 16.090909 16.303030 16.515152 16.727273 16.939394 17.151515
##  [64] 17.363636 17.575758 17.787879 18.000000 18.212121 18.424242 18.636364
##  [71] 18.848485 19.060606 19.272727 19.484848 19.696970 19.909091 20.121212
##  [78] 20.333333 20.545455 20.757576 20.969697 21.181818 21.393939 21.606061
##  [85] 21.818182 22.030303 22.242424 22.454545 22.666667 22.878788 23.090909
##  [92] 23.303030 23.515152 23.727273 23.939394 24.151515 24.363636 24.575758
##  [99] 24.787879 25.000000
##
## $y
##   [1]  6.00000  6.00000  6.00000       NA       NA       NA       NA       NA
##   [9]       NA       NA       NA       NA 13.00000 13.00000 13.00000 13.00000
##  [17] 13.00000 16.00000 16.00000 16.00000 16.00000 16.00000 10.00000 10.00000
##  [25] 10.00000 10.00000 26.00000 26.00000 26.00000 26.00000 26.00000 22.50000
##  [33] 22.50000 22.50000 22.50000 22.50000 21.50000 21.50000 21.50000 21.50000
##  [41] 21.50000 35.00000 35.00000 35.00000 35.00000 50.50000 50.50000 50.50000
##  [49] 50.50000 50.50000 33.33333 33.33333 33.33333 33.33333 33.33333 36.00000
##  [57] 36.00000 36.00000 36.00000 40.66667 40.66667 40.66667 40.66667 40.66667
##  [65] 64.50000 64.50000 64.50000 64.50000 64.50000 50.00000 50.00000 50.00000
##  [73] 50.00000 50.00000 50.40000 50.40000 50.40000 50.40000       NA       NA
##  [81]       NA       NA       NA 66.00000 66.00000 66.00000 66.00000 66.00000
##  [89] 54.00000 54.00000 54.00000 54.00000 93.75000 93.75000 93.75000 93.75000
##  [97] 93.75000 85.00000 85.00000 85.00000
```

```r
nw_gaussian <-
  ksmooth(speed, dist, kernel = 'normal', bandwidth = 1)
nw_gaussian
```

```
## $x
##   [1]  4.000000  4.212121  4.424242  4.636364  4.848485  5.060606  5.272727
##   [8]  5.484848  5.696970  5.909091  6.121212  6.333333  6.545455  6.757576
##  [15]  6.969697  7.181818  7.393939  7.606061  7.818182  8.030303  8.242424
##  [22]  8.454545  8.666667  8.878788  9.090909  9.303030  9.515152  9.727273
##  [29]  9.939394 10.151515 10.363636 10.575758 10.787879 11.000000 11.212121
##  [36] 11.424242 11.636364 11.848485 12.060606 12.272727 12.484848 12.696970
##  [43] 12.909091 13.121212 13.333333 13.545455 13.757576 13.969697 14.181818
##  [50] 14.393939 14.606061 14.818182 15.030303 15.242424 15.454545 15.666667
##  [57] 15.878788 16.090909 16.303030 16.515152 16.727273 16.939394 17.151515
##  [64] 17.363636 17.575758 17.787879 18.000000 18.212121 18.424242 18.636364
##  [71] 18.848485 19.060606 19.272727 19.484848 19.696970 19.909091 20.121212
##  [78] 20.333333 20.545455 20.757576 20.969697 21.181818 21.393939 21.606061
##  [85] 21.818182 22.030303 22.242424 22.454545 22.666667 22.878788 23.090909
##  [92] 23.303030 23.515152 23.727273 23.939394 24.151515 24.363636 24.575758
##  [99] 24.787879 25.000000
##
## $y
##   [1]  6.00000  6.00000  6.00000  6.00000  6.00000  6.00000  6.00000       NA
##   [9] 13.00000 13.00000 13.00000 13.00000 13.00144 13.00673 13.03127 13.14104
##  [17] 13.56304 14.55558 15.47362 15.69961 15.18509 13.49127 11.45293 10.82344
##  [25] 12.16487 16.67077 22.32167 25.03552 25.74906 25.78105 25.29895 24.11914
##  [33] 23.03072 22.57818 22.32735 21.96800 21.67726 21.65002 22.03337 23.66535
##  [41] 27.87816 32.44321 34.55172 35.77912 38.52125 44.01605 48.39480 49.75723
##  [49] 49.22462 46.06958 39.87539 35.33184 33.84805 33.67226 34.21093 35.19083
##  [57] 35.84004 36.27550 37.21687 38.92205 40.24251 41.05715 42.90506 48.53854
##  [65] 57.30442 62.43897 63.77348 63.18850 60.11286 54.78641 51.37114 50.32450
##  [73] 50.14902 50.23953 50.34994 50.38814 50.39741 50.39944 50.40417 50.49097
##  [81] 52.17829 61.91842 65.75199 65.98211 65.91667 65.61949 64.40428 60.98363
##  [89] 57.01260 56.24933 60.78352 73.39266 86.47830 91.93045 93.31359 93.49584
##  [97] 92.99142 91.10208 87.88560 85.83190
```

(b)

```r
tt1 <-
  loess(
    dist ~ speed,
    data = cars,
    span = 0.15,
    degree = 2,
    family = 'gaussian'
  )
tt2 <-
  loess(
    dist ~ speed,
    data = cars,
    span = 0.5,
    degree = 2,
    family = 'gaussian'
  )
```

(c)

Nadaraya-Watson Kernel Regression model:

```
##
nw_fitted_box <-
  ksmooth(
    speed,
    dist,
    kernel = 'box',
    bandwidth = 1,
    x.points = speed
  )$y
nw_fitted_box
```

```
##  [1]  6.00000  6.00000 13.00000 13.00000 16.00000 10.00000 26.00000 26.00000
##  [9] 26.00000 22.50000 22.50000 21.50000 21.50000 21.50000 21.50000 35.00000
## [17] 35.00000 35.00000 35.00000 50.50000 50.50000 50.50000 50.50000 33.33333
## [25] 33.33333 33.33333 36.00000 36.00000 40.66667 40.66667 40.66667 64.50000
## [33] 64.50000 64.50000 64.50000 50.00000 50.00000 50.00000 50.40000 50.40000
## [41] 50.40000 50.40000 50.40000 66.00000 54.00000 93.75000 93.75000 93.75000
## [49] 93.75000 85.00000
```

```
##
nw_fitted_gaussian <-
  ksmooth(
    speed,
    dist,
    kernel = 'normal',
    bandwidth = 1,
    x.points = speed
  )$y
nw_fitted_gaussian
```

```
##  [1]  6.00000  6.00000 13.03889 13.03889 15.70783 11.28350 25.80378 25.80378
##  [9] 25.80378 22.57818 22.57818 21.85378 21.85378 21.85378 21.85378 35.04991
## [17] 35.04991 35.04991 35.04991 49.78746 49.78746 49.78746 49.78746 33.94889
## [25] 33.94889 33.94889 36.07304 36.07304 41.38204 41.38204 41.38204 63.77348
## [33] 63.77348 63.77348 63.77348 50.48694 50.48694 50.48694 50.39379 50.39379
## [41] 50.39379 50.39379 50.39379 65.69288 57.97006 93.43566 93.43566 93.43566
## [49] 93.43566 85.83190
```

Local polynomial model:

```
## model with span=0.15: fitted y
fitted(tt1) # == predict(tt1)
```

```
##  [1]  5.97671  5.97671 13.00000 13.00000 16.00000 10.00000 26.00000 26.00000
##  [9] 26.00000 22.50000 22.50000 21.50000 21.50000 21.50000 21.50000 35.00000
## [17] 35.00000 35.00000 35.00000 50.50000 50.50000 50.50000 50.50000 33.33333
## [25] 33.33333 33.33333 36.00000 36.00000 40.66667 40.66667 40.66667 64.50000
## [33] 64.50000 64.50000 64.50000 50.00000 50.00000 50.00000 50.40000 50.40000
## [41] 50.40000 50.40000 50.40000 66.00000 74.50000 93.75000 93.75000 93.75000
## [49] 93.75000 85.00000
```

```
## model with span=0.5: fitted y
fitted(tt2) # == predict(tt2)
```

```
##  [1]  6.124161  6.124161 12.329738 12.329738 14.801415 16.733767 18.990452
##  [8] 18.990452 18.990452 22.667608 22.667608 29.208622 29.208622 29.208622
## [15] 29.208622 35.383851 35.383851 35.383851 35.383851 41.588031 41.588031
## [22] 41.588031 41.588031 40.553445 40.553445 40.553445 43.519416 43.519416
## [29] 47.479712 47.479712 47.479712 55.016200 55.016200 55.016200 55.016200
## [36] 54.434758 54.434758 54.434758 54.012981 54.012981 54.012981 54.012981
## [43] 54.012981 64.322532 73.542693 85.271671 85.271671 85.271671 85.271671
## [50] 99.770407
```

Compare:

```
nw_box_error <- dist - nw_fitted_box
nw_box_error
```

```
##  [1]  -4.0000000   4.0000000  -9.0000000   9.0000000   0.0000000   0.0000000
##  [7]  -8.0000000   0.0000000   8.0000000  -5.5000000   5.5000000  -7.5000000
## [13]  -1.5000000   2.5000000   6.5000000  -9.0000000  -1.0000000  -1.0000000
## [19]  11.0000000 -24.5000000 -14.5000000   9.5000000  29.5000000 -13.3333333
## [25]  -7.3333333  20.6666667  -4.0000000   4.0000000  -8.6666667  -0.6666667
## [31]   9.3333333 -22.5000000  -8.5000000  11.5000000  19.5000000 -14.0000000
## [37]  -4.0000000  18.0000000 -18.4000000  -2.4000000   1.6000000   5.6000000
## [43]  13.6000000   0.0000000   0.0000000 -23.7500000  -1.7500000  -0.7500000
## [49]  26.2500000   0.0000000
```

```
nw_gaussian_error <- dist - nw_fitted_gaussian
nw_gaussian_error
```

```
##  [1]  -4.0000000   4.0000000  -9.0388879   8.9611121   0.2921668  -1.2835007
##  [7]  -7.8037832   0.1962168   8.1962168  -5.5781768   5.4218232  -7.8537816
## [13]  -1.8537816   2.1462184   6.1462184  -9.0499096  -1.0499096  -1.0499096
## [19]  10.9500904 -23.7874619 -13.7874619  10.2125381  30.2125381 -13.9488853
## [25]  -7.9488853  20.0511147  -4.0730417   3.9269583  -9.3820379  -1.3820379
## [31]   8.6179621 -21.7734842  -7.7734842  12.2265158  20.2265158 -14.4869446
## [37]  -4.4869446  17.5130554 -18.3937940  -2.3937940   1.6062060   5.6062060
## [43]  13.6062060   0.3071219  -3.9700566 -23.4356563  -1.4356563  -0.4356563
## [49]  26.5643437  -0.8318986
```

```
local_error1 <- dist - fitted(tt1)
local_error1
```

```
##  [1] -3.976710e+00  4.023290e+00 -9.000000e+00  9.000000e+00  3.552714e-15
##  [6]  5.329071e-15 -8.000000e+00  7.105427e-15  8.000000e+00 -5.500000e+00
## [11]  5.500000e+00 -7.500000e+00 -1.500000e+00  2.500000e+00  6.500000e+00
## [16] -9.000000e+00 -1.000000e+00 -1.000000e+00  1.100000e+01 -2.450000e+01
## [21] -1.450000e+01  9.500000e+00  2.950000e+01 -1.333333e+01 -7.333333e+00
## [26]  2.066667e+01 -4.000000e+00  4.000000e+00 -8.666667e+00 -6.666667e-01
## [31]  9.333333e+00 -2.250000e+01 -8.500000e+00  1.150000e+01  1.950000e+01
## [36] -1.400000e+01 -4.000000e+00  1.800000e+01 -1.840000e+01 -2.400000e+00
## [41]  1.600000e+00  5.600000e+00  1.360000e+01  0.000000e+00 -2.050000e+01
## [46] -2.375000e+01 -1.750000e+00 -7.500000e-01  2.625000e+01  0.000000e+00
```

```
local_error2 <- dist - fitted(tt2)
local_error2
```

```
##  [1]  -4.1241610   3.8758390  -8.3297383   9.6702617   1.1985851  -6.7337672
##  [7]  -0.9904521   7.0095479  15.0095479  -5.6676077   5.3323923 -15.2086223
## [13]  -9.2086223  -5.2086223  -1.2086223  -9.3838512  -1.3838512  -1.3838512
## [19]  10.6161488 -15.5880314  -5.5880314  18.4119686  38.4119686 -20.5534451
## [25] -14.5534451  13.4465549 -11.5194163  -3.5194163 -15.4797119  -7.4797119
## [31]   2.5202881 -13.0162002   0.9837998  20.9837998  28.9837998 -18.4347581
## [37]  -8.4347581  13.5652419 -22.0129809  -6.0129809  -2.0129809   1.9870191
## [43]   9.9870191   1.6774684 -19.5426926 -15.2716715   6.7283285   7.7283285
## [49]  34.7283285 -14.7704067
```

```
## SSE & MSE & std.error
nw_box_sse <-
  sum(nw_box_error ^ 2)
nw_gaussian_sse <- sum(nw_gaussian_error ^ 2)
local_sse1 <- sum(local_error1 ^ 2)
local_sse2 <- sum(local_error2 ^ 2)
nw_box_mse <- nw_box_sse / 50
nw_gaussian_mse <- nw_gaussian_sse / 50
local_mse1 <- local_sse1 / 50
local_mse2 <- local_sse2 / 50
Model <-
  c(
    "Nadaraya-Watson with box kernel",
    "Nadaraya-Watson with gaussian kernel",
    "local polynomial with span=0.15",
    "local polynomial with span=0.5"
  )
SSE <- c(nw_box_sse, nw_gaussian_sse, local_sse1, local_sse2)
MSE <- c(nw_box_mse, nw_gaussian_mse, local_mse1, local_mse2)
std.error <-
```

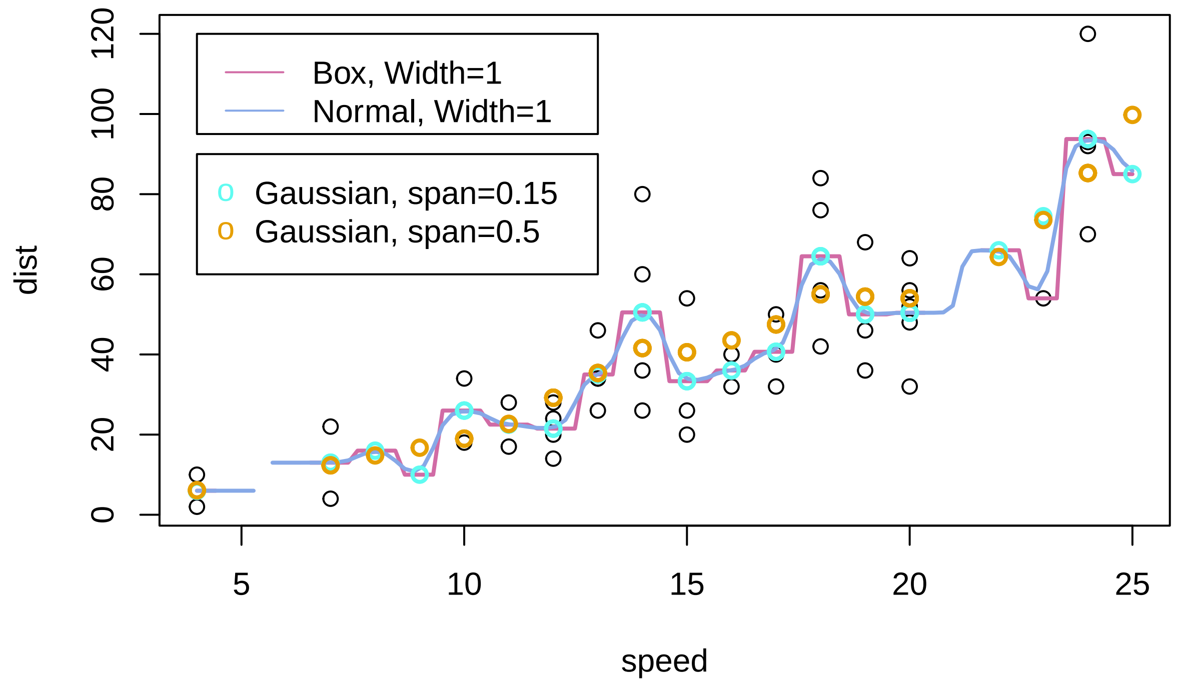
```
  c(sd(nw_box_error),
    sd(nw_gaussian_error),
    sd(local_error1),
    sd(local_error2))
data.frame(Model, SSE, MSE, std.error)
```

```
##                                  Model      SSE      MSE std.error
## 1      Nadaraya-Watson with box kernel 6764.783 135.2957  11.74976
## 2 Nadaraya-Watson with gaussian kernel 6791.637 135.8327  11.77257
## 3      local polynomial with span=0.15 7185.034 143.7007  12.10217
## 4       local polynomial with span=0.5 9299.113 185.9823  13.76285
```

Nadaraya-Watson kernel regression model with box kernel fits better.

(d)

```
plot(speed, dist)
lines(nw_box, col = '#D16BA5', lwd = 2)
lines(nw_gaussian, col = '#86A8E7', lwd = 2)
points(tt1$x, fitted(tt1), col = '#5FFBF1', lwd = 2)
points(tt2$x, fitted(tt2), col = '#E69F00', lwd = 2)
legend(
  x = c(4, 13),
  y = c(95, 120),
  c("Box, Width=1", "Normal, Width=1"),
  col = c('#D16BA5', '#86A8E7'),
  lwd = 1
)
legend(
  x = c(4, 13),
  y = c(60, 90),
  c("Gaussian, span=0.15", "Gaussian, span=0.5"),
  col = c('#5FFBF1', '#E69F00'),
  pch = 'o'
)
```



```
detach(cars)
```

3.2

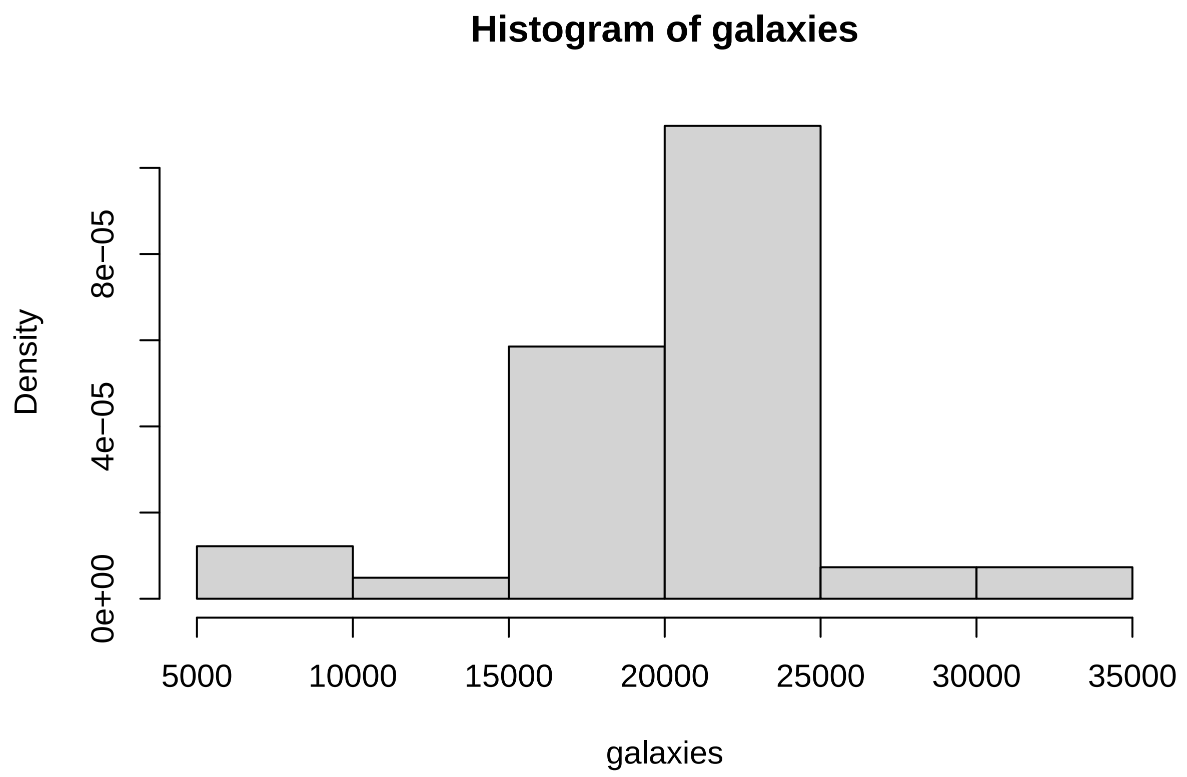
```
library(MASS)
data(galaxies)
```

Histogram Smoothing:

s is the sample standard deviation and n is the sample zie, then

$$h^* = 3.491sn^{-1/3}$$

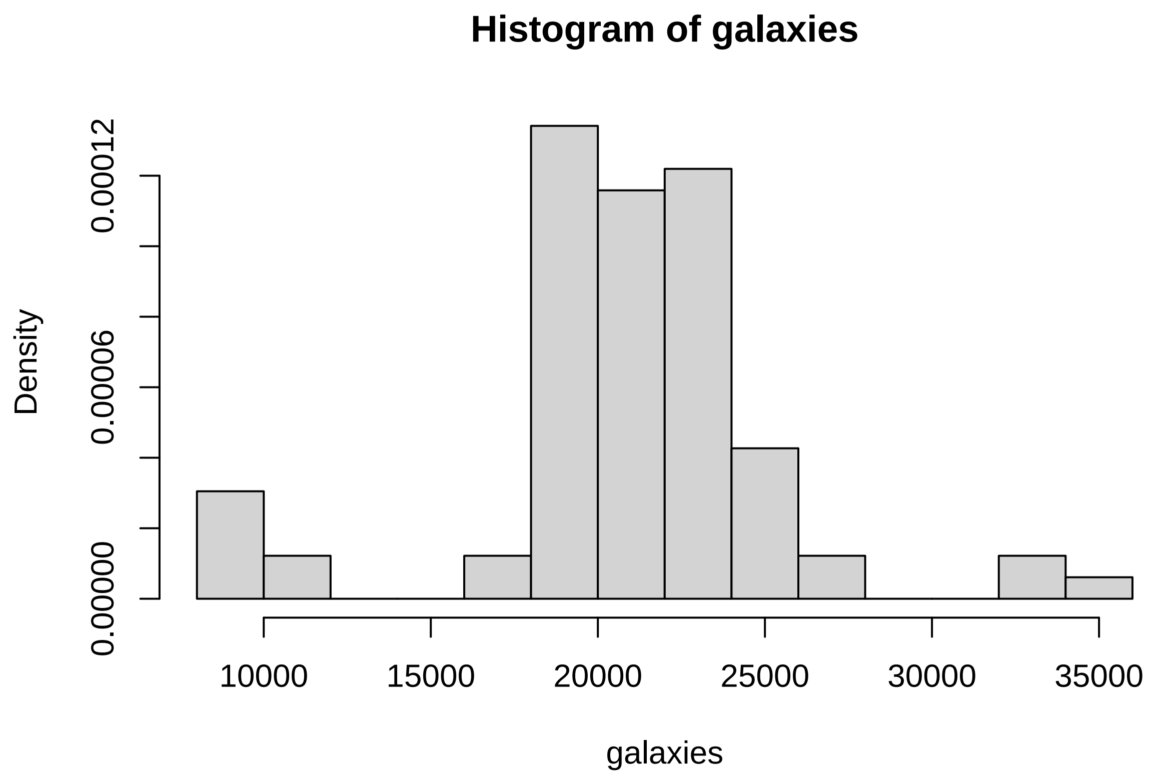
```
n <- length(galaxies)
s <- sd(galaxies)
iqr <- IQR(galaxies)
h1 <- 3.491 * s * n ^ {
  -1 / 3
}
nobreaks <- (max(galaxies) - min(galaxies)) / h1
hist(galaxies,
     breaks = round(nobreaks),
     probability = TRUE)
```



Another sensible estimate is obtained by replacing s by the inter-quantile range, IQR, that is

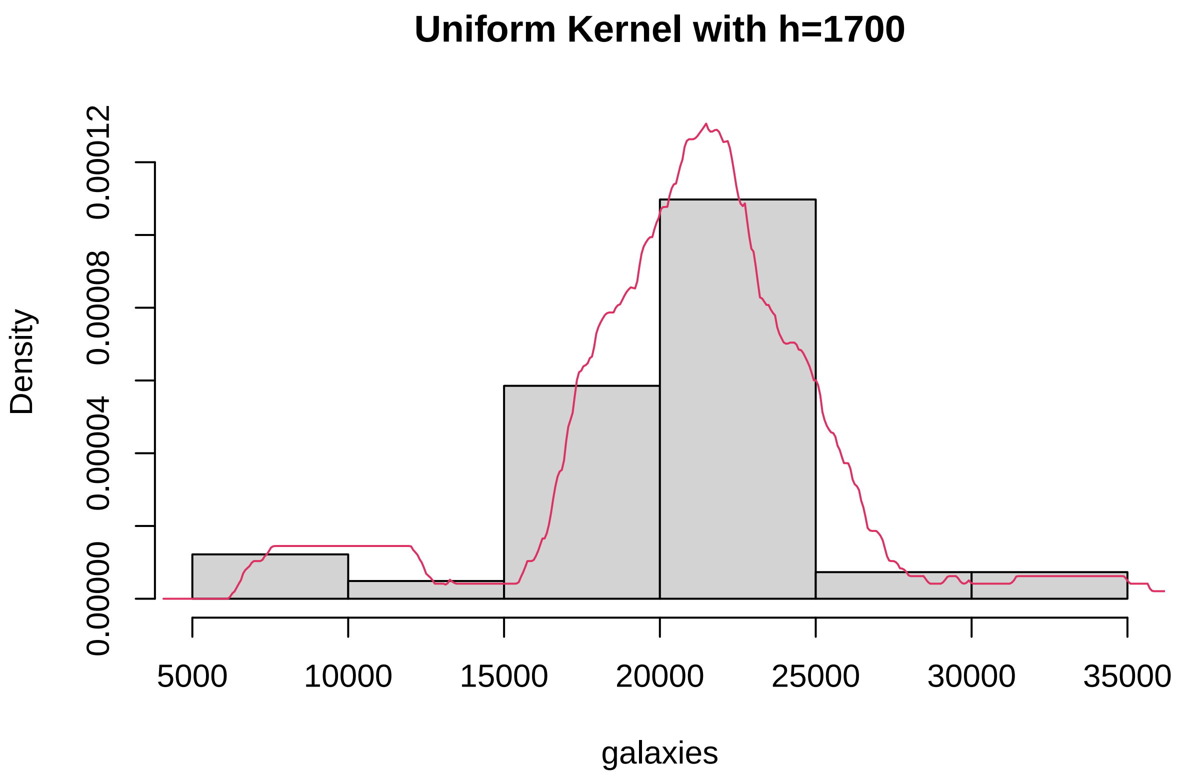
$$h^* = 2.6 IQR \times n^{-1/3}$$

```
h2 <- 2.6 * iqr * n ^ {
  -1 / 3
}
nobreaks2 <- (max(galaxies) - min(galaxies)) / h2
hist(galaxies,
     breaks = round(nobreaks2),
     probability = TRUE)
```

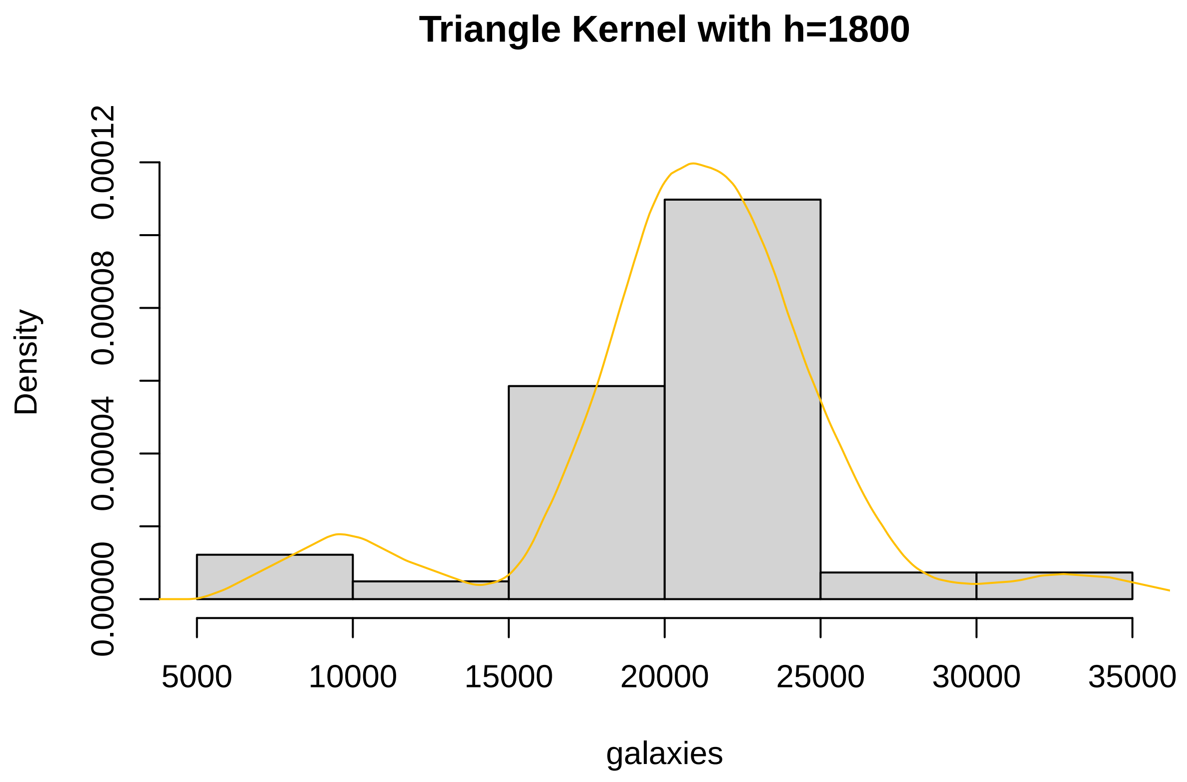


Kernel Smoothing:

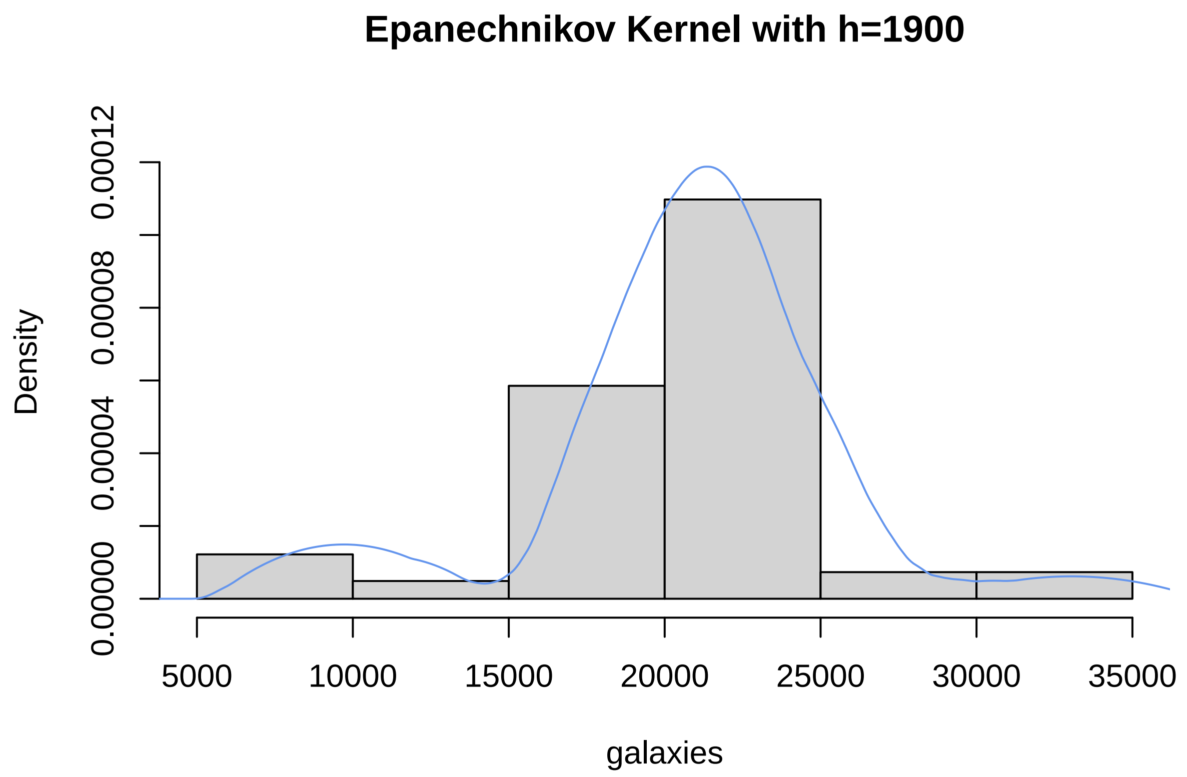
```
hist(
  galaxies,
  breaks = round(nobreaks),
  probability = TRUE,
  ylim = c(0, 13e-05),
  main = 'Uniform Kernel with h=1700'
)
lines(density(galaxies, kernel = 'rectangular', bw = 1700), col = '#DE3163')
```



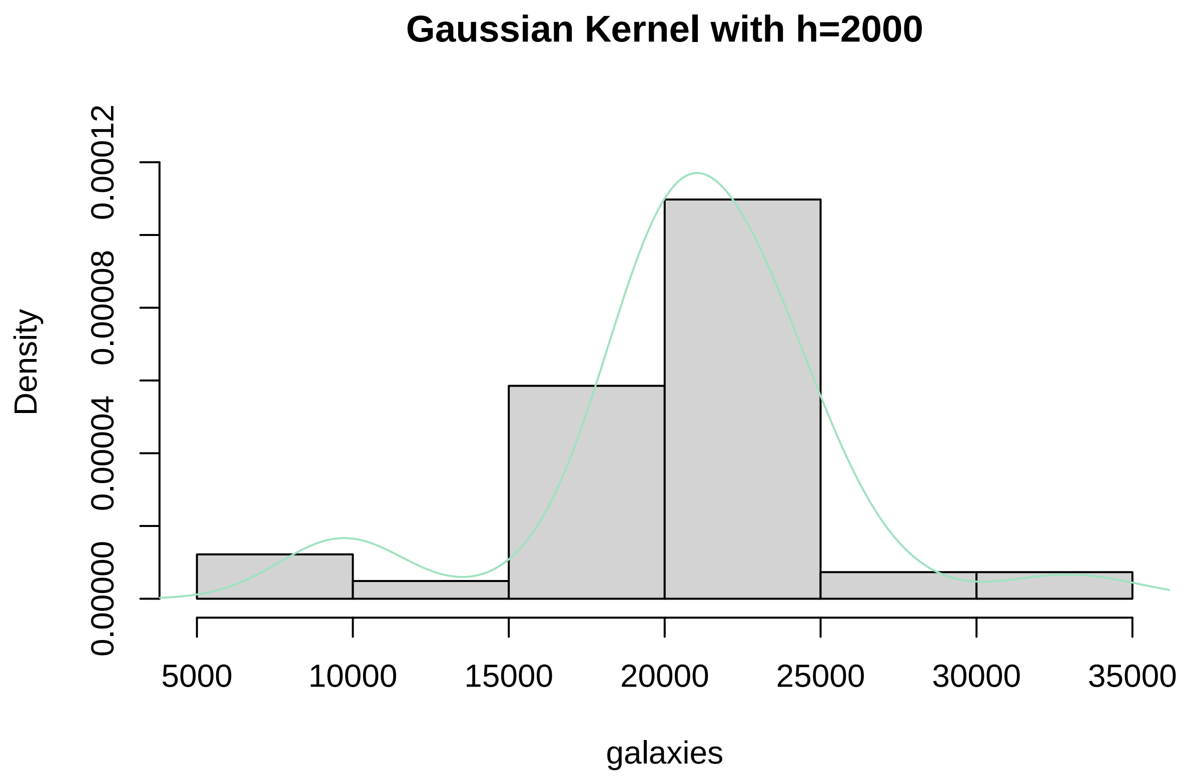
```
hist(
  galaxies,
  breaks = round(nobreaks),
  probability = TRUE,
  ylim = c(0, 13e-05),
  main = 'Triangle Kernel with h=1800'
)
lines(density(galaxies, kernel = 'triangular', bw = 1800), col = '#FFBF00')
```



```
hist(
  galaxies,
  breaks = round(nobreaks),
  probability = TRUE,
  ylim = c(0, 13e-05),
  main = 'Epanechnikov Kernel with h=1900'
)
lines(density(galaxies, kernel = 'epanechnikov', bw = 1900), col = '#6495ED')
```



```
hist(
  galaxies,
  breaks = round(nobreaks),
  probability = TRUE,
  ylim = c(0, 13e-05),
  main = 'Gaussian Kernel with h=2000'
)
lines(density(galaxies, kernel = 'gaussian', bw = 2000), col = '#9FE2BF')
```



There are at least 2 peaks of the distribution of velocities. Thus, the multimodality of the distribution of velocities implies the existence of superclusters.

3.3

```
library(HSAUR3)
```

```
##     tools
```

```
data(foster)
attach(foster)
```

(a)

```
table(litgen, motgen)
```

```
##       motgen
## litgen A B I J
##      A 5 3 4 5
##      B 4 5 4 2
##      I 3 3 5 3
##      J 4 3 3 5
```

```
# group means
aggregate(weight, by = list(litgen, motgen), FUN = mean)
```

```
##    Group.1 Group.2        x
## 1        A       A 63.68000
## 2        B       A 52.32500
## 3        I       A 47.10000
## 4        J       A 54.35000
## 5        A       B 52.40000
## 6        B       B 60.64000
## 7        I       B 64.36667
## 8        J       B 56.10000
## 9        A       I 54.12500
## 10       B       I 53.92500
## 11       I       I 51.60000
## 12       J       I 54.53333
## 13       A       J 48.96000
## 14       B       J 45.90000
## 15       I       J 49.43333
## 16       J       J 49.06000
```

```
# group standard deviations
aggregate(weight, by = list(litgen, motgen), FUN = sd)
```

```
##    Group.1 Group.2         x
## 1        A       A  3.273683
## 2        B       A  5.533158
## 3        I       A 18.103315
## 4        J       A  5.325098
## 5        A       B  9.374433
## 6        B       B  5.647389
## 7        I       B  7.124839
## 8        J       B  3.351119
## 9        A       I  5.321889
## 10       B       I  5.114277
## 11       I       I  8.624964
## 12       J       I  8.376953
## 13       A       J  8.760594
## 14       B       J  7.636753
## 15       I       J  5.372461
## 16       J       J  5.335541
```

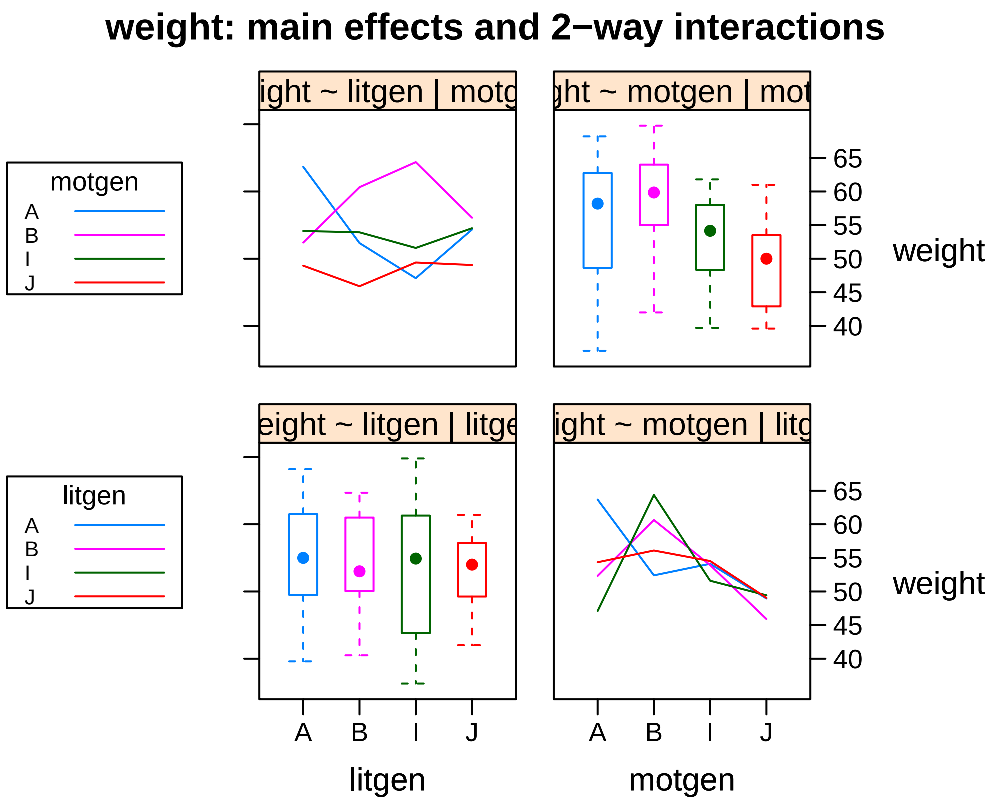
(b)

```
library(HH)
```

```
##      lattice
```

```
##      grid
##      latticeExtra

##
##    'latticeExtra'

## The following object is masked from 'package:ggplot2':
##
##     layer

##      multcomp

##      mvtnorm

##      survival

##      TH.data

##
##    'TH.data'

## The following object is masked from 'package:HSAUR3':
##
##     birds

## The following object is masked from 'package:MASS':
##
##     geyser

##      gridExtra
```

```
interaction2wt(weight ~ litgen * motgen)
```



From the plot in the upper left corner, the slope of different *motgen* from one *litgen* type to another *litgen* type is different. Similarly, from the plot in the lower right corner, the slope of different *litgen* with respect to *motgen* is not the same. Therefore, there seems to exist some interaction between *litgen* and *motgen*.

(c)

```
fit1 <- aov(weight ~ litgen * motgen)
fit2 <- aov(weight ~ motgen * litgen)
fit3 <- aov(weight ~ litgen + motgen)
fit4 <- aov(weight ~ motgen + litgen)
summary(fit1)
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)
## litgen         3   60.2   20.05   0.370 0.77522
## motgen         3  775.1  258.36   4.763 0.00574 **
## litgen:motgen  9  824.1   91.56   1.688 0.12005
## Residuals     45 2440.8   54.24
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(fit2)
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)
## motgen         3  771.6  257.20   4.742 0.00587 **
## litgen         3   63.6   21.21   0.391 0.76000
## motgen:litgen  9  824.1   91.56   1.688 0.12005
## Residuals     45 2440.8   54.24
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(fit3)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## litgen       3     60   20.05   0.332 0.80247
## motgen       3    775  258.36   4.273 0.00886 **
## Residuals   54   3265   60.46
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(fit4)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## motgen       3    772  257.20   4.254 0.00905 **
## litgen       3     64   21.21   0.351 0.78870
## Residuals   54   3265   60.46
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All the results indicate that the *motgen* main effect is significant and the *litgen* main effect is not significant. For analysis with interaction term, the results indicate that there exists some interaction between *motgen* and *litgen*, but this interaction is not significant(p=0.12005).

(d)

The dependent variable is assumed to be normally distributed, and have equal variance in each group. outlier

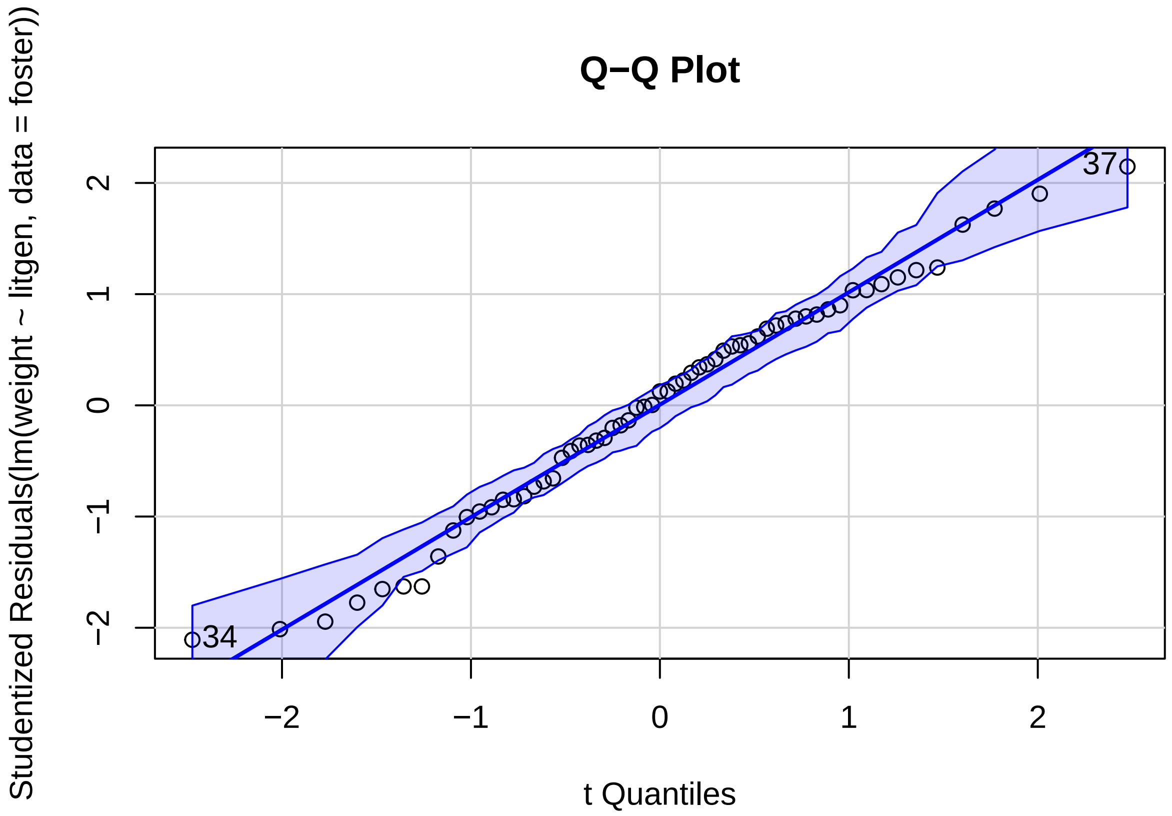
```
# Normally distributed:
library(car)
```

```
##      carData
```

```
##
##    'car'
```

```
## The following objects are masked from 'package:HH':
##
##     logit, vif
```

```
qqPlot(
  lm(weight ~ litgen, data = foster),
  simulate = TRUE,
  main = 'Q-Q Plot',
  labels = FALSE
)
```



```
## [1] 34 37
```

```
# equality of variances
bartlett.test(weight ~ litgen, data = foster)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  weight by litgen
## Bartlett's K-squared = 6.1503, df = 3, p-value = 0.1045
```

```
# outlier
fit <- aov(weight ~ litgen)
outlierTest(fit)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
```

```
##    rstudent unadjusted p-value Bonferroni p
## 37 2.147241           0.036118           NA
```

Yes. These assumptions are satisfied.

(e)

```
library(lmPerm)
set.seed(1234)
aovobject <- aovp(weight ~ litgen * motgen, data = foster, perm = "Prob")
```

```
## [1] "Settings:  unique SS "
```

```
summary(aovobject)
```

```
## Component 1 :
##                Df R Sum Sq R Mean Sq Iter Pr(Prob)
## litgen          3    27.66     9.219  114   0.9737
## motgen          3   671.74   223.913 5000   0.0084 **
## litgen:motgen   9   824.07    91.564 2158   0.1348
## Residuals      45  2440.82    54.240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
### result in (c)
summary(fit1)
```

```
##                Df Sum Sq Mean Sq F value  Pr(>F)
## litgen          3   60.2   20.05   0.370 0.77522
## motgen          3  775.1  258.36   4.763 0.00574 **
## litgen:motgen   9  824.1   91.56   1.688 0.12005
## Residuals      45 2440.8   54.24
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Compared to the result in (c), the *litgen* main effect, the *motgen* main effect and the interaction are all less significant.

```
detach(foster)
```

3.4

```
library(ISLR)
data(Default)
```

(a)

```r
summary(Default)
```

```
##  default    student       balance           income     
##  No :9667   No :7056   Min.   :   0.0   Min.   :  772  
##  Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340  
##                        Median : 823.6   Median :34553  
##                        Mean   : 835.4   Mean   :33517  
##                        3rd Qu.:1166.3   3rd Qu.:43808  
##                        Max.   :2654.3   Max.   :73554
```

```r
logit_fit <-
  glm(default ~ student + balance + income,
      family = binomial(link = "logit"),
      data = Default)
summary(logit_fit)
```

```
## 
## Call:
## glm(formula = default ~ student + balance + income, family = binomial(link = "logit"), 
##     data = Default)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 ** 
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
## 
## Number of Fisher Scoring iterations: 8
```

The estimated standard error for the estimated coefficient of studentYes is 2.363×10^{-1}, while the estimated standard error for the estimated coefficient of balance and income are 2.319×10^{-4} and 8.203×10^{-6} respectively.

(b)

```r
boot.fn <- function(formula, data, indices) {
  d <- data[indices, ]
  fit <- glm((formula),
             family = binomial(link = "logit"),
             data = d)
  return(coef(fit))
}
```

(c)

```
library(boot)
```

```
##
##     'boot'

## The following object is masked from 'package:car':
##
##     logit

## The following object is masked from 'package:HH':
##
##     logit

## The following object is masked from 'package:survival':
##
##     aml

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
set.seed(1234)
results <-
  boot(
    data = Default,
    statistic = boot.fn,
    R = 1000,
    formula = default ~ student + balance + income
  )
print(results)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000, formula = default ~
##     student + balance + income)
##
##
## Bootstrap Statistics :
```

```
##           original        bias     std. error
## t1* -1.086905e+01 -2.846982e-02 5.022490e-01
## t2* -6.467758e-01 -1.151472e-02 2.390398e-01
## t3*  5.736505e-03  1.953943e-05 2.330627e-04
## t4*  3.033450e-06 -1.787413e-07 8.595409e-06
```

The estimated standard error for the estimated coefficient of studentYes is 2.390398×10^{-1}, while the estimated standard error for the estimated coefficient of balance and income are 2.330627×10^{-4} and 8.595409×10^{-6} respectively.

(d)

The standard errors obtained by the bootstrap appear to be a quite close to those obtained using the statistical formulas underlying the glm() function. This suggests that the data satisfies the underlying assumptions of a logistic regression model: the responses Y_i are independent random variables coming from Bernoulli distributions with probabilities P_i, and the log-odds corresponding to P_i is a linear combination of the predictors.