# Problem Set 4

Ian Sims

March 9, 2019

Load packages:

## 1. Recall that a Poisson distribution with parameter ?? has probability mass function f(x) = ??xe????? x!

**Does the number of children a woman has follow a Poisson distribution? We collect data from 1761 adult German women, and count their children:**

**Number of children 0 1 2 3 4 5 6 7 8 9 10 Women with this number of children 398 455 575 227 65 28 9 3 0 1 0**

**Perform a goodness-of-fit test to see whether this data can be well-modeled using a Poisson distribution, stating the value of your test statistic, your P-value, and your conclusion.**

```
#Using the Chi-Square test statistic

dd = data.frame("children" = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10), "n" =
c(398, 455, 575, 227, 65, 28, 9, 3, 0, 1, 0))

N = sum(dd$n)

observed = dd$n
lambda.obs = sum((dd$children * dd$n))/N

expected = N * dpois(dd$children, lambda.obs)

extra = N - sum(expected)

chi.stat = sum((observed - expected)^2/expected) + extra

poisson.chisq.sim = function(N, lambda.obs){
  sim.sample = data.frame(x = rpois(N, lambda.obs))
  lambda.sim = mean(sim.sample$x)
  x.tally = tally(group_by(sim.sample, x))
  observed = x.tally$n
  expected = N * dpois(x.tally$x, lambda.sim)
  extra = N - sum(expected)
  chi.stat = sum((observed - expected)^2/expected) + extra
  return(chi.stat)
}
```

```
chi.list = replicate(10000, poisson.chisq.sim(N, lambda.obs))
p.value = mean(chi.list >= chi.stat)
```

Using the Chi-Square test statistic and simulation the P-Value is approximatley equal to 0.01. This is a fairly small p-value and would lead us to conclude that this data is not consistent with a Poisson model.

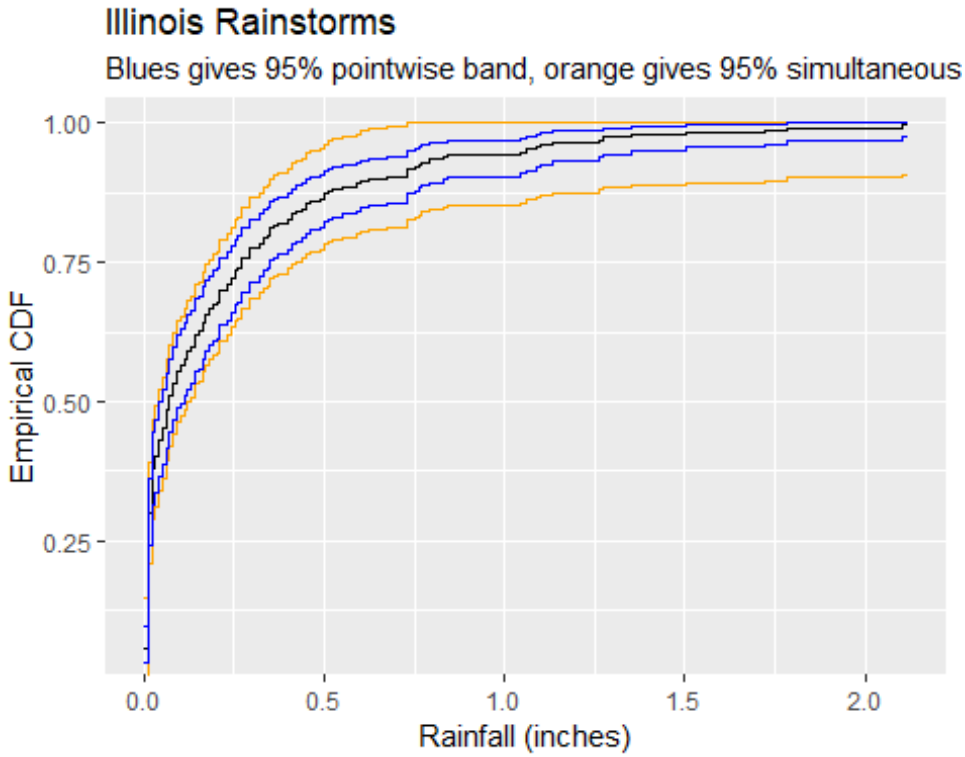
**2. The data set illinois-rainstorms.txt gives the rainfall (in inches) in a sample of 227 rainstorms in Illinois.On the same graph, plot:**

**(a) A 95% pointwise confidence band for the CDF of rainfall;**

**(b) A 95% simultaneous confidence band for the CDF of rainfall. and clearly indicate on the graph which band is which.**

```
dd = scan("illinois-rainstorms.txt")
F.hat = ecdf(dd)

x.grid = seq(min(dd), max(dd), 0.01)
point.lower = rep(NA, length(x.grid))
point.upper = rep(NA, length(x.grid))
n = length(dd)

for(J in 1:length(x.grid)){
  CI = binom.test(sum(dd <= x.grid[J]), n)$conf.int
  point.lower[J] = CI[1]
  point.upper[J] = CI[2]
}

alpha = 0.05
epsilon = sqrt(log(2 / alpha) / (2 * n))

L = function(y) {
  raw = F.hat(y) - epsilon
  return(ifelse(raw < 0, 0, raw))
}

U = function(y) {
  raw = F.hat(y) + epsilon
  return(ifelse(raw > 1, 1, raw))
}

simul.lower = L(x.grid)
simul.upper = U(x.grid)

y = F.hat(x.grid)
plot.df = data.frame(x = x.grid, y)

gg = ggplot(plot.df, aes(x, y)) + geom_step() + geom_step(y = simul.lower,
```

```
col = "orange") + geom_step(y = simul.upper, col = "orange") + geom_step(y =
point.lower, col = "blue") + geom_step(y = point.upper, col = "blue")

gg + ggtitle("Illinois Rainstorms") + labs(subtitle = "Blues gives 95%
pointwise band, orange gives 95% simultaneous band") + xlab("Rainfall
(inches)") + ylab("Empirical CDF")
```

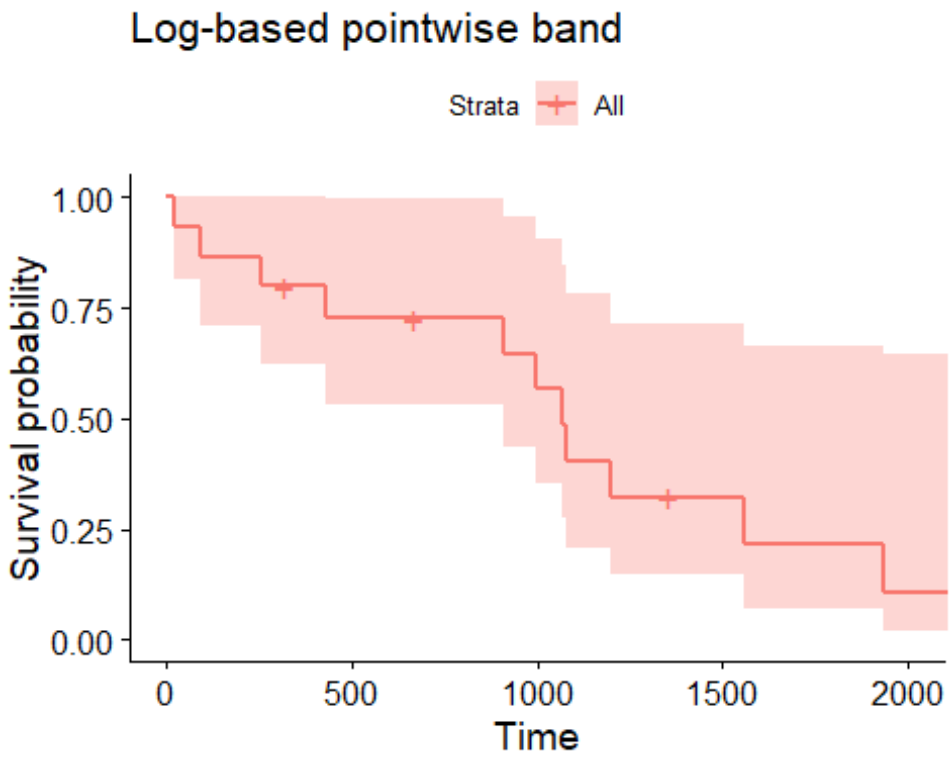



**3. Here are survival times (in days) for a sample of HIV patients (a "+" indicates the patient was still alive at the last time of observation):**

**22, 90, 256, 320+, 428, 670+, 910, 997, 1070, 1081, 1197, 1355+, 1560, 1933, 2202**

**Use R to obtain the Kaplan-Meier estimate of the survival function, and plot it along with pointwise confidence limits calculated using a method of your choice.**

```
time = c(22, 90, 256, 320, 428, 670, 910, 997, 1070, 1081, 1197, 1355, 1560,
1933, 2202)
event = c(1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1)

surv.data = Surv(time = time, event = event)

surv.log = survfit(surv.data ~ 1, conf.type = "log")

ggsurvplot(surv.log, data = surv.data) + ggtitle("Log-based pointwise band")
```




**4. For the Illinois rainstorms data:**

**(a) Find the sample mean $\bar{x}$, and use the bootstrap to estimate the mean squared error of $\bar{x}$ as an estimate of the population mean.**

```
dd = scan("illinois-rainstorms.txt")

x.bar = mean(dd)
x.bar

## [1] 0.2243921

mean.boot = replicate(10000, mean(sample(dd, replace = TRUE)))

mean((mean.boot - x.bar)^2)

## [1] 0.0005864679
```

The sample mean is approximatley .224 and the boostrap estimate of the mean squared error is approximatley .001.

**(b) Find the sample standard deviation s, and use the bootstrap to estimate the mean squared error of s as an estimate of the population standard deviation.**

```
dd = scan("illinois-rainstorms.txt")

s = sd(dd)
s
```

```
## [1] 0.3658212
```

```
s.boot = replicate(10000, sd(sample(dd, replace = TRUE)))

mean((s.boot - s)^2)
```

```
## [1] 0.001614204
```

The sample sd is approximatley .366 and the boostrap estimate of the mean squared error is approximatley .002.

**(c) Find the sample coefficient of variation s/¯x, and use the bootstrap to estimate the mean squared error of s/¯x as an estimate of the population coefficient of variation.**

```
dd = scan("illinois-rainstorms.txt")

dd.cv = cv(dd) / 100
dd.cv
```

```
## [1] 1.630277
```

```
dd.cv.boot = replicate(10000, cv(sample(dd, replace = TRUE)) / 100)

mean((dd.cv.boot - dd.cv)^2)
```

```
## [1] 0.0100056
```

The sample cv is approximatley 1.630 and the boostrap estimate of the mean squared error is approximatley .010.

## 5. The file rabbits.txt contains the eosinophil counts of a bunch of rabbits.

**(a) Find a 95% bootstrap Studentized t-pivot confidence interval for the mean eosinophil count of rabbits, without using the boot package.**

```
dd =
c(55,140,91,122,111,185,203,101,76,145,95,101,196,45,299,226,65,70,196,72,121
,171,151,113,112,67,276,125,100,81,122,71,158,78,162,128,96,79,67,119)

n = length(dd)
x.bar = mean(dd)

t.pivot = function(x){
  x.boot = sample(x, replace = TRUE)
  t.boot = (mean(x.boot) - mean(x)) /
           (sd(x.boot) / sqrt(length(x)))
  return(t.boot)
}
```

```
t.boot = replicate(10000, t.pivot(dd))

quantile(t.boot, c(0.025, 0.975))

##      2.5%     97.5%
## -2.492650  1.771479

qt(c(.025, .975), df = n-1)

## [1] -2.022691  2.022691

ci = mean(dd) - quantile(t.boot, c(0.975, 0.025)) * sd(dd)/sqrt(n)

x.bar

## [1] 124.775

ci

##    97.5%     2.5%
## 108.4047 147.8097
```

The 95% bootstrap Studentized t-pivot ci for the mean eosinophil is approximatley (107.96, 148.17).

**(b) Find a 95% bootstrap BCA confidence interval for the mean eosinophil count of rabbits, using whatever packages you wish.**

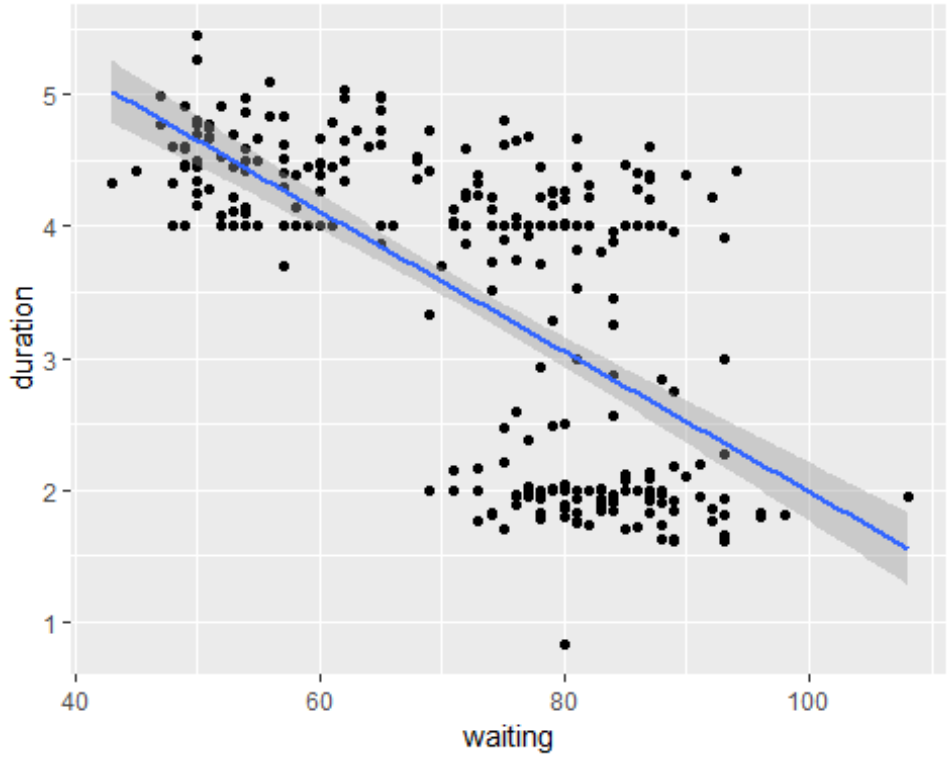
```
dd =
c(55,140,91,122,111,185,203,101,76,145,95,101,196,45,299,226,65,70,196,72,121
,171,151,113,112,67,276,125,100,81,122,71,158,78,162,128,96,79,67,119)

bootmean = function(x, indices){
  return(mean(x[indices]))
}

bootmean.dist = boot(dd, bootmean, R = 10000)
boot.ci(bootmean.dist)

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootmean.dist)
##
## Intervals :
## Level      Normal              Basic
## 95%   (106.8, 142.5 )   (105.9, 141.7 )
##
## Level     Percentile            BCa
## 95%   (107.8, 143.7 )   (109.2, 145.6 )
## Calculations and Intervals on Original Scale
```

The 95% bootstrap BCa ci for the mean eosinophil is approximatley (109.2, 145.6).

## 6. The file geyser.txt contains two variables: waiting, which gives the waiting times (in minutes) between eruptions of Old Faithful, and duration, which gives the duration (in minutes) of the eruption following that waiting time.

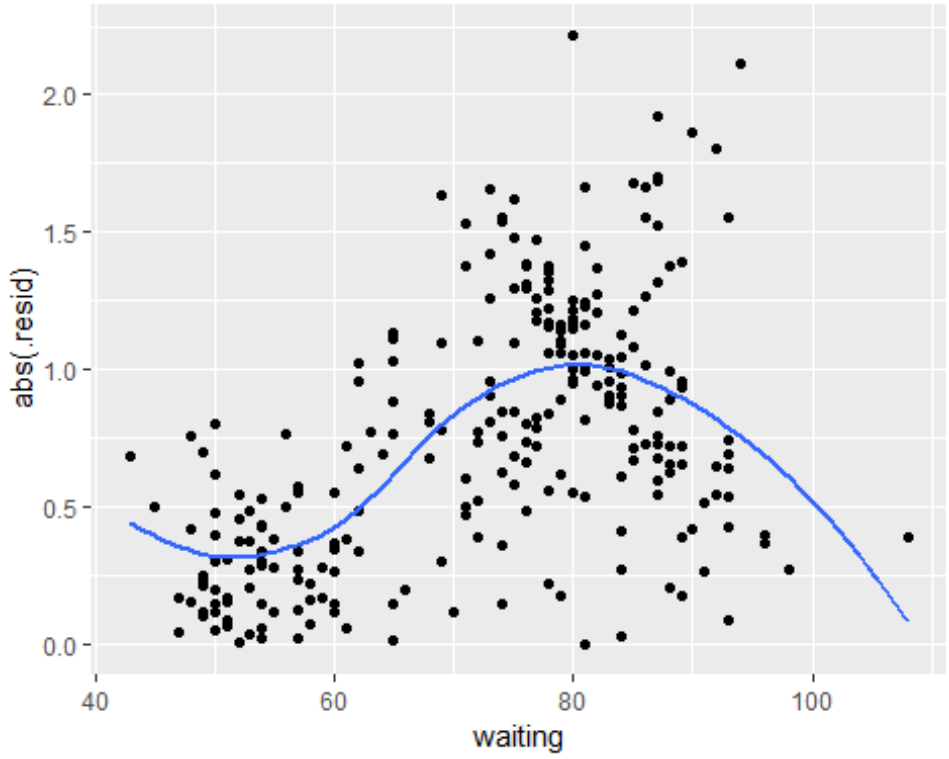
**(a) Plot the data and add a regression line to predict duration from waiting time. Does it look like the key assumptions required for classical regression inference (linearity and homoskedasticity) are met?**

```
dd = read.csv("geyser.txt", sep = " ", header = TRUE)

ggplot(dd, aes(x = waiting, y = duration)) + geom_point() +
geom_smooth(method = "lm")
```




```
eruption.model = lm(duration ~ waiting, data = dd)
get_regression_table(eruption.model)

## # A tibble: 2 x 7
##   term      estimate std_error statistic p_value lower_ci upper_ci
##   <chr>        <dbl>     <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
## 1 intercept    7.31      0.27       27.1       0     6.78    7.84
## 2 waiting     -0.053     0.004     -14.5       0    -0.06   -0.046
```

```
eruption.model.df = augment(eruption.model)
ggplot(eruption.model.df, aes(x = waiting, y = abs(.resid))) + geom_point() +
geom_smooth(se = FALSE)

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




Based on the residual plot there is significant heteroskedascity in the residuals.

**(b) Even if classical assumptions are not met, we can still use the bootstrap to do inference. Find a 95% bootstrap confidence interval for the slope parameter of the linear regression, and carefully explain what this interval means.**

```
dd = read.csv("geyser.txt", sep = " ", header = TRUE)

lm.pairs.boot = function(data, indices){
    newdata = data[indices,]
    return(lm(newdata[,2] ~ newdata[,1])$coef[2])
}

boot.lm.dist = boot(cbind(dd$waiting, dd$duration), lm.pairs.boot, R = 10000)

boot.ci(boot.lm.dist)

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

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.lm.dist)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.0590, -0.0474 )   (-0.0591, -0.0475 )
##
## Level     Percentile            BCa
## 95%   (-0.0590, -0.0475 )   (-0.0588, -0.0472 )
## Calculations and Intervals on Original Scale
```

The BCa boostrap estimate of the 95% ci for the slope of the regression line is approxamately (-0.0588, -0.0472). This means that if the assumptions of this method are met and you created a band based on this interval about 95% of the time it would encompass the true value of the regression function.

## 7. (Computationally expensive; set aside lots of computing time.) Suppose we wish to find a 95% confidence interval for the mean from an IID sample of size 20 from a chi-square distribution with 1 degree of freedom. (You can generate such a sample using rchisq().) Perform simulations to estimate the level of coverage of

### (a) The percentile bootstrap

Sorry I could not figure out how to translate the simulation for the coverage for the Studentized to these other bootstrapping methods.

### (b) The residual (basic) bootstrap

Sorry I could not figure out how to translate the simulation for the coverage for the Studentized to these other bootstrapping methods.

### (c) The BCa bootstrap

Sorry I could not figure out how to translate the simulation for the coverage for the Studentized to these other bootstrapping methods.

### (d) The Studentized (t-pivot) bootstrap

```
n = 20
chi.sim = function(n, B) {
  x = rchisq(n, 1)
  m = mean(x)
  s = sd(x)
  s2 = var(x)
  chi.cis = function(x, B) {
```

```
    chi.boot = function(x, m, s) {
      x.boot = sample(x, replace = T)
      t = (mean(x.boot) - m)/(sd(x.boot)/sqrt(n))
      chi2 = (n - 1) * var(x.boot)/s2
      return(c(t, chi2))
    }
    boot.dist = replicate(B, chi.boot(x, m, s))
    m.ci = m - quantile(boot.dist[1, ], c(0.975, 0.025)) * s/sqrt(n)
    v.ci = (n - 1) * s2/quantile(boot.dist[2, ], c(0.975, 0.025))
    return(c(m.ci, v.ci))
    }
    return(chi.cis(x, B))
  }
chi.rep = replicate(1000, chi.sim(n, B = 1000))
1 - mean(chi.rep[1, ] > 1) - mean(chi.rep[2, ] < 1)

## [1] 0.944
```

The coverage of this simulation is 0.944.

**Note: The number of bootstrap replications has little effect on relative accuracy, so keep B to a moderate value like 1000.**

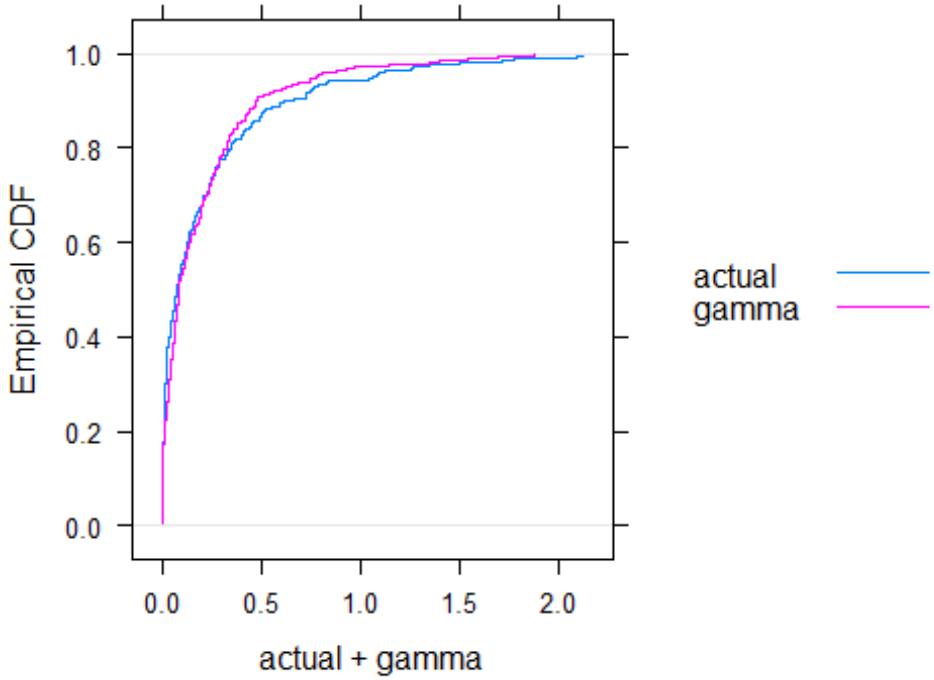
## 8. For the Illinois rainstorms data:

**(a) Find the maximum likelihood estimates of the shape and rate parameters of a gamma distribution fitted to the data.**

```
dd = scan("illinois-rainstorms.txt")

fit = fitdistr(dd, "gamma")
fit

##      shape         rate
##   0.44080171   1.96476293
##  (0.03376322) (0.24743990)
```

**(b) Plot (on the same graph): . The empirical CDF of the data; . The CDF of the gamma distribution you estimated.**

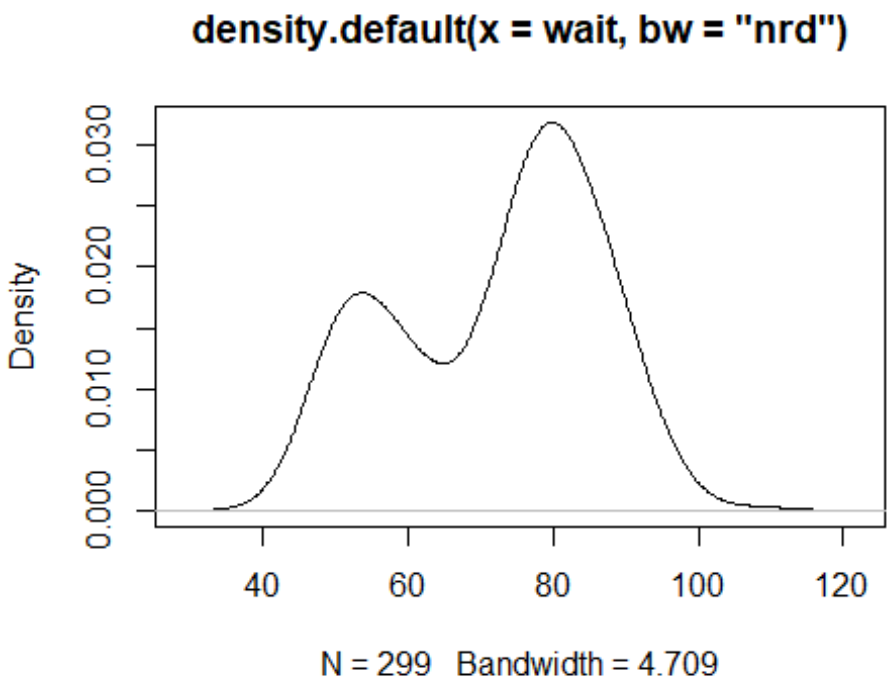
```
dd = scan("illinois-rainstorms.txt")

fit = fitdistr(dd, "gamma")

dd_gamma = rgamma(length(dd), fit$estimate[1], fit$estimate[2])

vals <- data.frame(actual=dd, gamma=dd_gamma)

ecdfplot(~ actual + gamma, data=vals, auto.key=list(space='right'))
```




## 9. For the geyser.txt data:

**(a) Choose a bandwidth for a Gaussian kernel to estimate the PDF of waiting time, stating how how you (or R) calculated the bandwidth. Plot a Gaussian kernel density estimate of the PDF of waiting time using this bandwidth.**

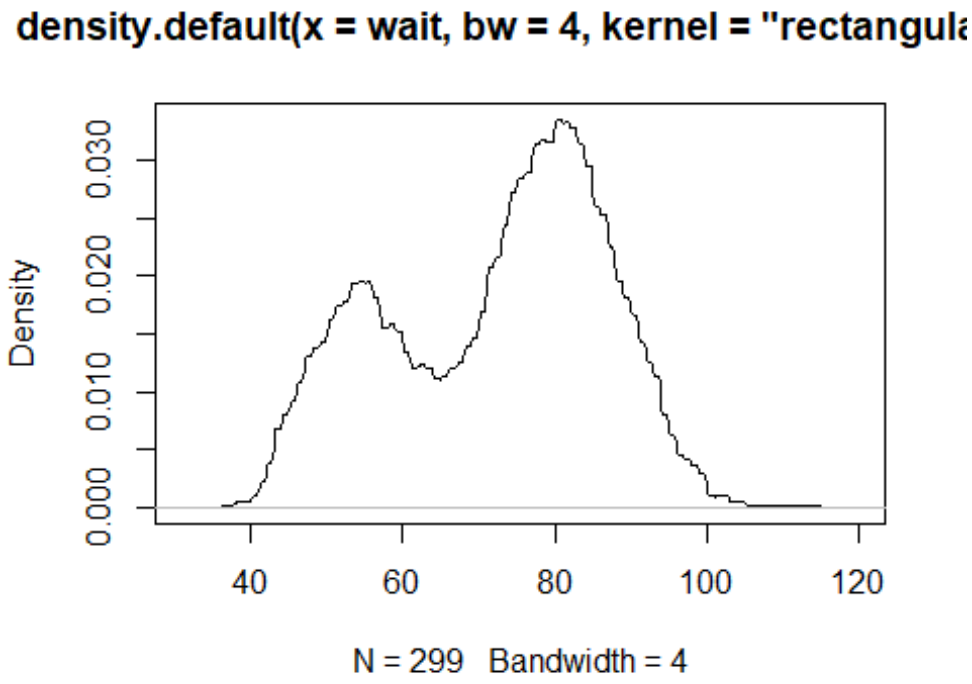
```
dd = read.csv("geyser.txt", sep = " ", header = TRUE)

wait = dd$waiting

f.hat = density(wait, bw = "nrd")

plot(f.hat)
```




I used the standard R function for plotting this. This function uses the standard formula for the bandwidth for the Gaussian kernal of 1.06 * sd * n ^ (-.2).

**(b) Your friend is allergic to normal distributions, and asks you to instead create a kernel density estimate using a uniform kernel with bandwidth 4 (meaning the kernel stretches two minutes up and down from each observation.) Plot such an estimate. Hint: Make sure the area under the curve is 1.**

```
dd = read.csv("geyser.txt", sep = " ", header = TRUE)

wait = dd$waiting

f.hat = density(wait, bw = 4, kernel = "rectangular")

plot(f.hat)
```



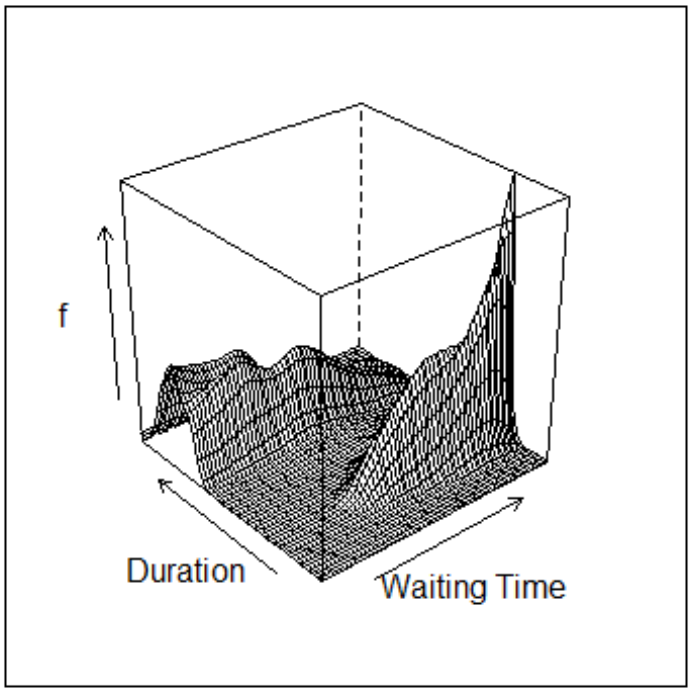

I'm not sure this is what you were looking for, but I couldn't find anything in the notes to help with this.

## 10. For the geyser.txt data:

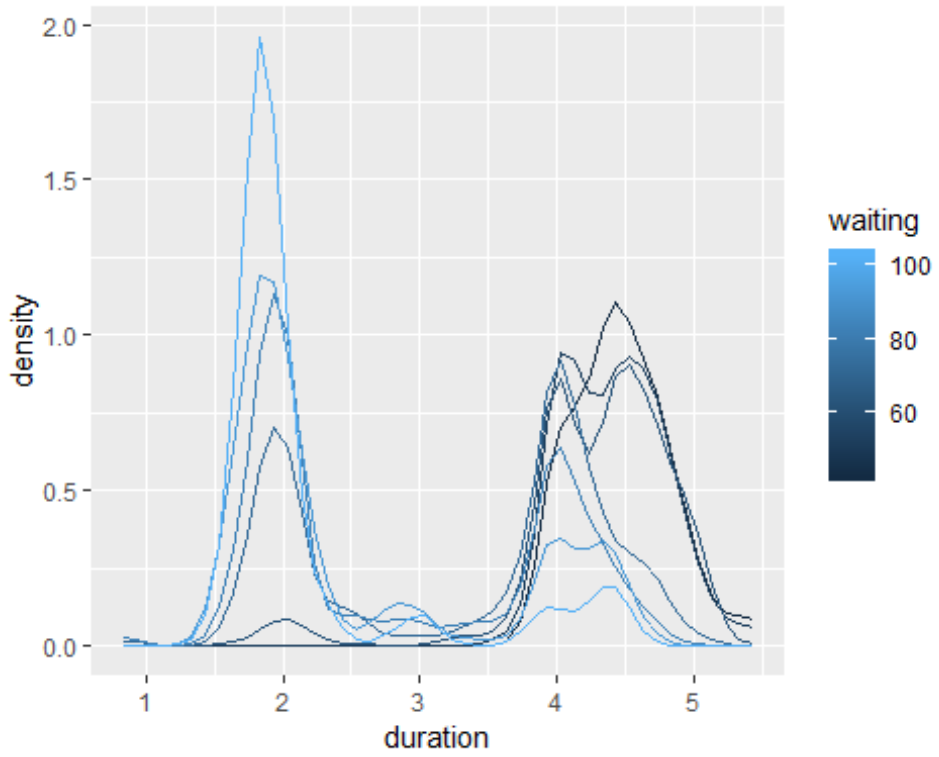
**(a) Plot a conditional density estimate of eruption duration given the previous waiting time. Describe what your plot tells you.**

```
dd = read.csv("geyser.txt", sep = " ", header = TRUE)

dur.cdens = npcdens(duration ~ waiting, data = dd)

##
Multistart 1 of 2 |
Multistart 1 of 2 |
Multistart 1 of 2 |
Multistart 1 of 2 /
Multistart 1 of 2 -
Multistart 1 of 2 |
Multistart 1 of 2 |
Multistart 2 of 2 |
Multistart 2 of 2 |
Multistart 2 of 2 /
Multistart 2 of 2 -
Multistart 2 of 2 |
```

```
Multistart 2 of 2 |


grid = expand.grid(waiting = seq(min(dd$waiting), max(dd$waiting), 1),
duration = seq(min(dd$duration), max(dd$duration), 0.1))

f.hat = predict(dur.cdens, newdata = grid)

wireframe(f.hat ~ grid$waiting * grid$duration, xlab = "Waiting Time", ylab =
"Duration", zlab = "f")
```




```
grid2 = expand.grid(waiting = seq(min(dd$waiting), max(dd$waiting), 10),
duration = seq(min(dd$duration), max(dd$duration), 0.1))
f.hat2 = predict(dur.cdens, newdata = grid2)
predict.df = data.frame(grid2, density = f.hat2)
ggplot(predict.df, aes(x = duration, y = density, group = waiting, color =
waiting)) + geom_line()
```



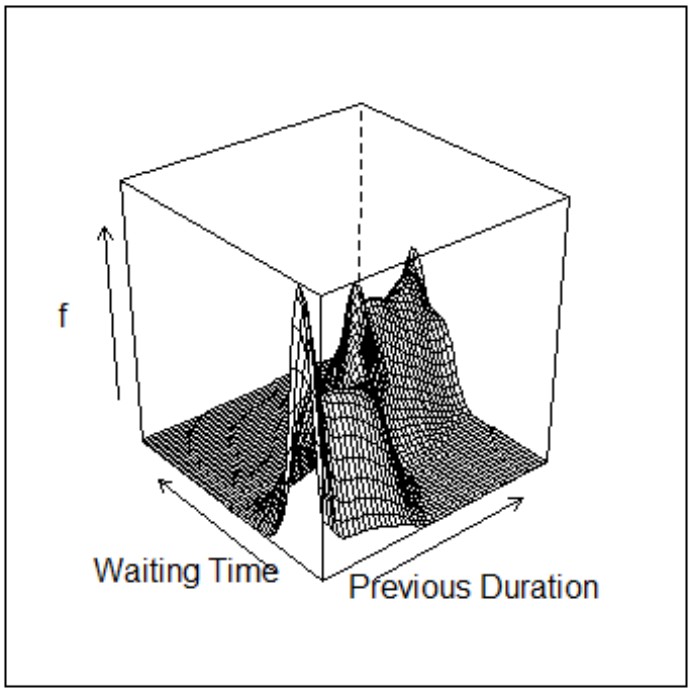

These plots indicate that the probablilty density changes for various levels of the predictor variable. Specifically that for higher levels of waiting time the density function for the duration shows higher probablilties for lower duration. Similarly as the waiting time increases the density function shifts to give higher probabilities to shorter durations.

**(b) Plot a conditional density estimate of waiting time given the previous eruption duration. (You'll have to manipulate the data to get the previous duration to line up with the following waiting time.) Describe what your plot tells you.**

```
dd = read.csv("geyser.txt", sep = " ", header = TRUE)

shift <- function(x, n){
  c(x[-(seq(n))], rep(NA, n))
}

dd$waiting <- shift(dd$waiting, 1)

dd = head(dd,-1)

wait.cdens = npcdens(waiting ~ duration, data = dd)

##
Multistart 1 of 2 |
Multistart 1 of 2 |
Multistart 1 of 2 |
Multistart 1 of 2 /
```

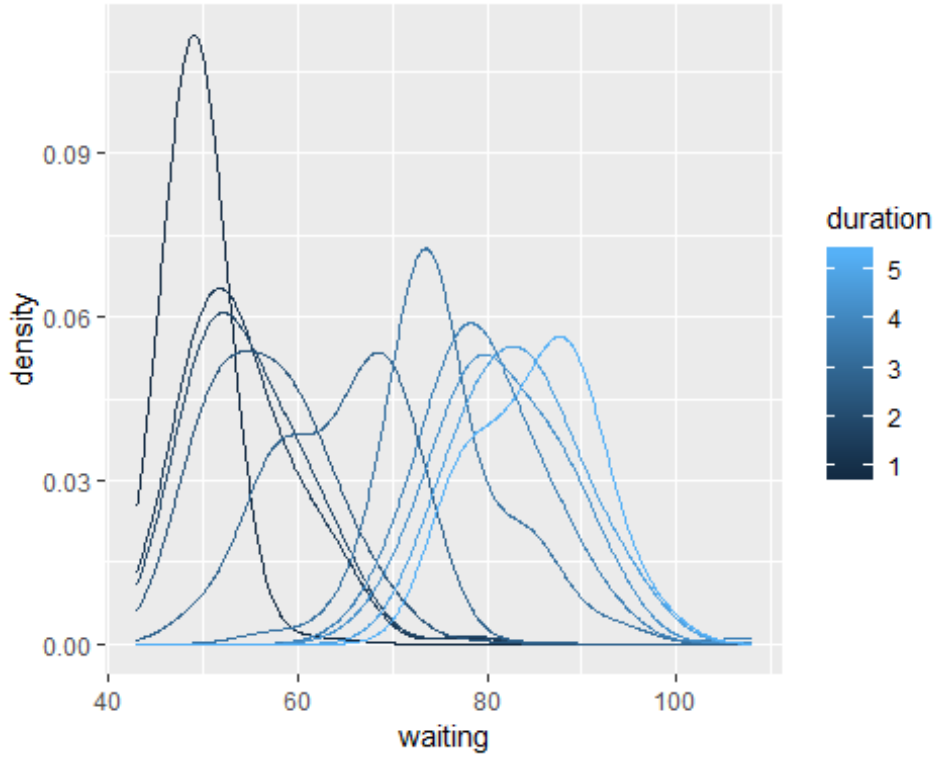
```
Multistart 1 of 2 -
Multistart 1 of 2 \
Multistart 1 of 2 |
Multistart 1 of 2 |
Multistart 2 of 2 |
Multistart 2 of 2 |
Multistart 2 of 2 /
Multistart 2 of 2 -
Multistart 2 of 2 |
Multistart 2 of 2 |
Multistart 2 of 2 /
Multistart 2 of 2 -
```

```
grid = expand.grid(duration = seq(min(dd$duration), max(dd$duration), .1),
waiting = seq(min(dd$waiting), max(dd$waiting), 1))

f.hat = predict(wait.cdens, newdata = grid)

wireframe(f.hat ~ grid$duration * grid$waiting, xlab = "Previous Duration",
ylab = "Waiting Time", zlab = "f")
```




```
grid2 = expand.grid(duration = seq(min(dd$duration), max(dd$duration), .5),
waiting = seq(min(dd$waiting), max(dd$waiting), 0.1))
f.hat2 = predict(wait.cdens, newdata = grid2)
predict.df = data.frame(grid2, density = f.hat2)
```

```
ggplot(predict.df, aes(x = waiting, y = density, group = duration, color =
duration)) + geom_line()
```




These plots indicate that the probablilty density changes for various levels of the predictor variable. Specifically that for higher levels of previous duration the density function for the waiting time shows higher probablilties for longer waiting times. Similarly as the previous duration decreases the density function shifts to give higher probabilities to shorter waiting times.