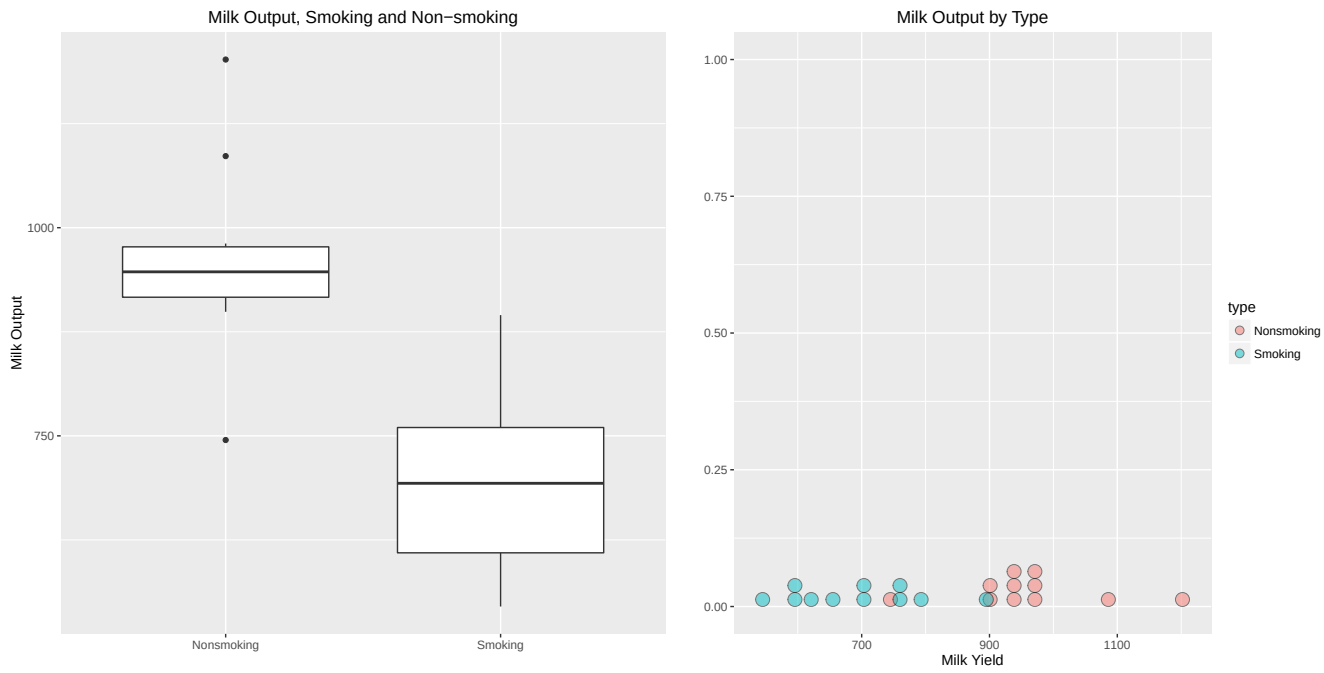
36-463 / 36-663: Multilevel & Hierarchical Models
HW02 Solution

September 19, 2016

Exercise 1

Part a

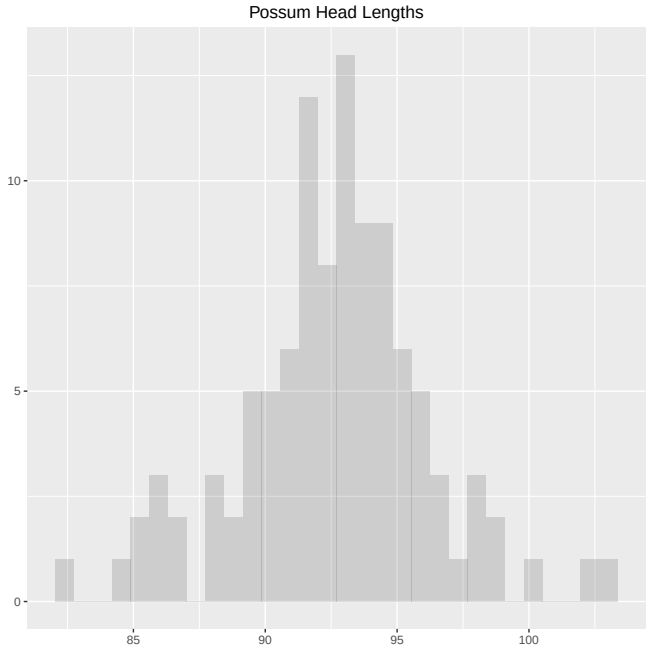
```
yieldvec <- c(621,793,593,545,753,655,895,767,714,598,693)
yield <- c(yieldvec, 947,945,1086,1202,973,981,930,745,903,899,961)
n0 <- length(yieldvec)
n1 <- length(yieldvec)
typevec <- c(rep("Smoking",n0),rep("Nonsmoking",n1))
milk.vol <- data.frame(yield = yield, type = typevec)

library(ggplot2)
milkplot <- ggplot(milk.vol, aes(type, yield))
milkboxplot  <- milkplot + geom_boxplot() + ggtitle("Milk Output, Smoking and Non-smoking") +
     xlab("") + ylab("Milk Output")
milkdotplot <- ggplot(milk.vol, aes(yield, fill = type)) + geom_dotplot(alpha = 0.5)
milkdotplot <- milkdotplot + ggtitle("Milk Output by Type") + ylab("") + xlab("Milk Yield")
milkboxplot
milkdotplot
```

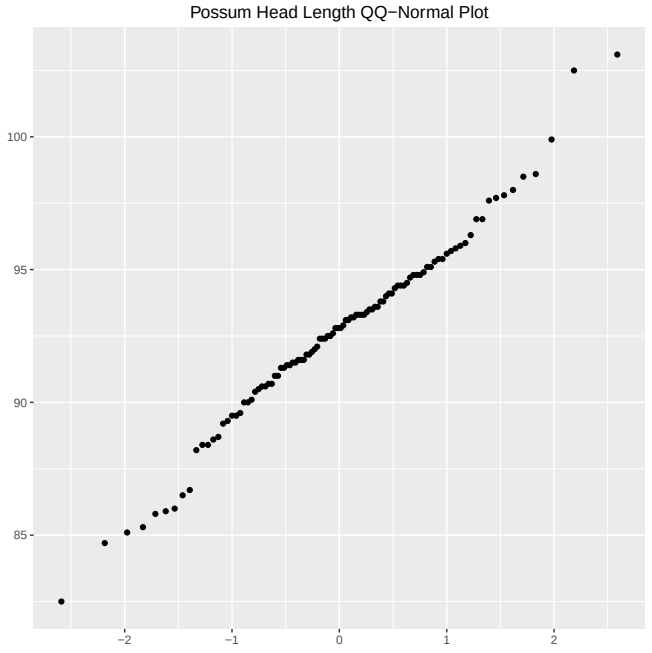


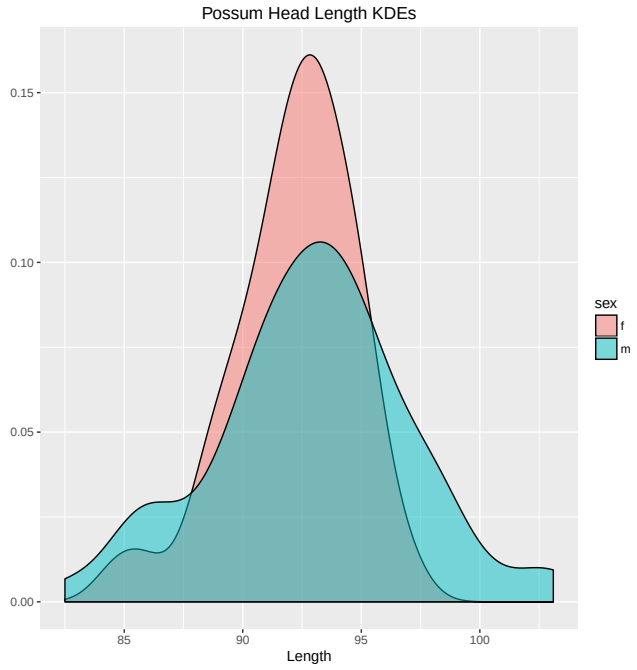
part b)

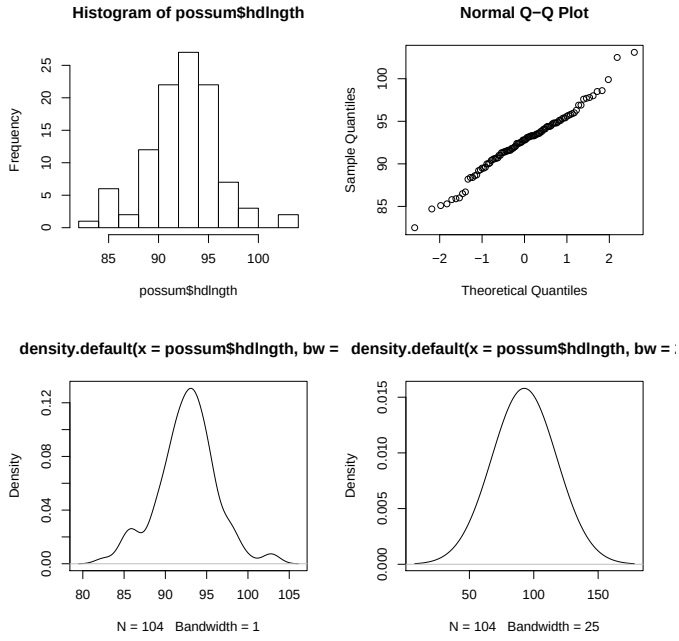
```
possumhist <- ggplot(possum, aes(x = hdlngth)) +
  geom_histogram(alpha = 0.2, position = "identity") +
  ggtitle("Possum Head Lengths") + ylab("") + xlab("")
possumhist
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.

possumqq <- ggplot(possum, aes(sample = hdlngth)) +
  geom_point(stat = 'qq') +
  ggtitle("Possum Head Length QQ-Normal Plot") + xlab("") + ylab("")
possumqq

possumdensplot <- ggplot(possum, aes(hdlngth, fill = sex)) +
  stat_density(aes(y = ..density..), position = "identity", color = "black", alpha = .5) +
  ggtitle("Possum Head Length KDEs") + xlab("Length") + ylab("")
possumdensplot

hist(possum$hdlngth)
qqnorm(possum$hdlngth)
plot(density(possum$hdlngth,bw =1))
plot(density(possum$hdlngth,bw =25))
```

These functions are very similar. Notice that the functions in part c not only generates the plots but also can return the corresponding output values.





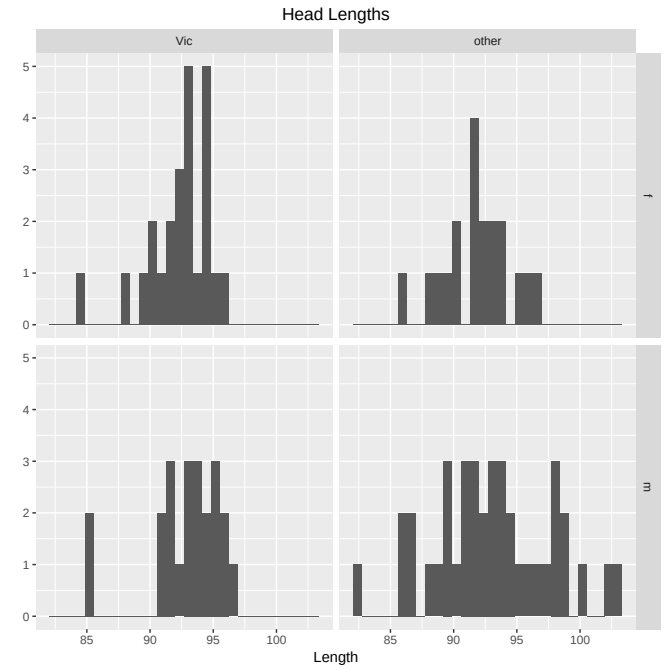


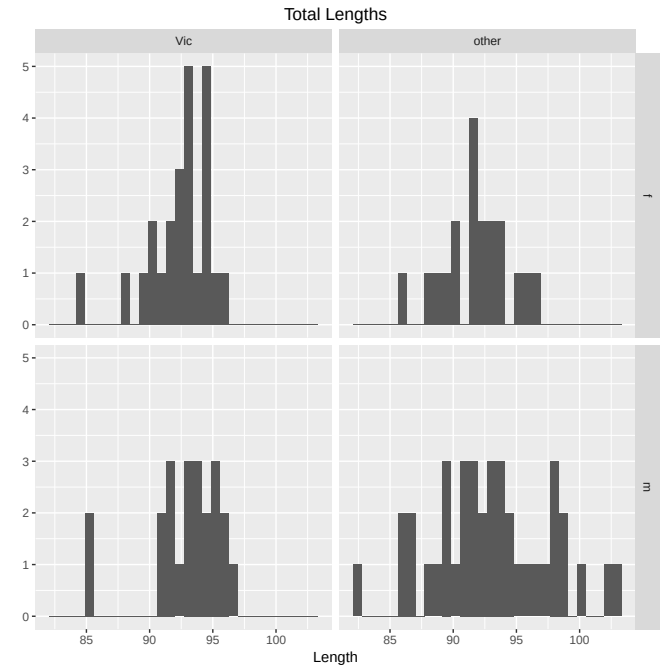


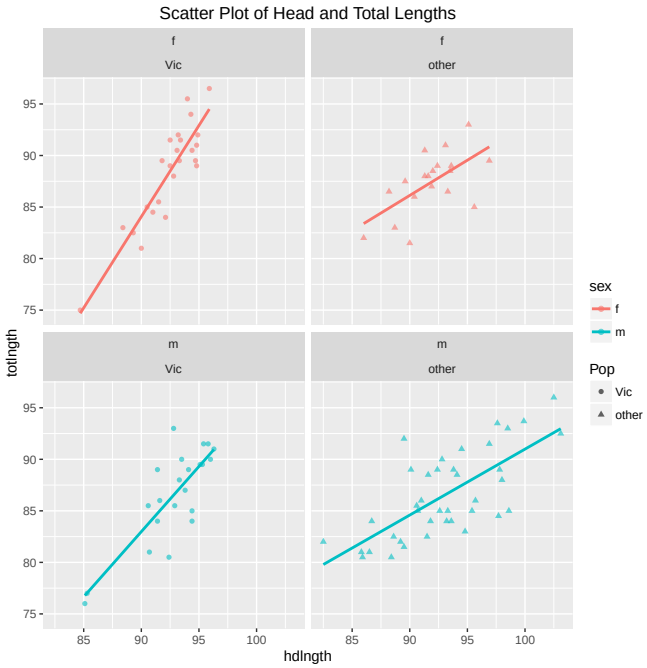
part d)

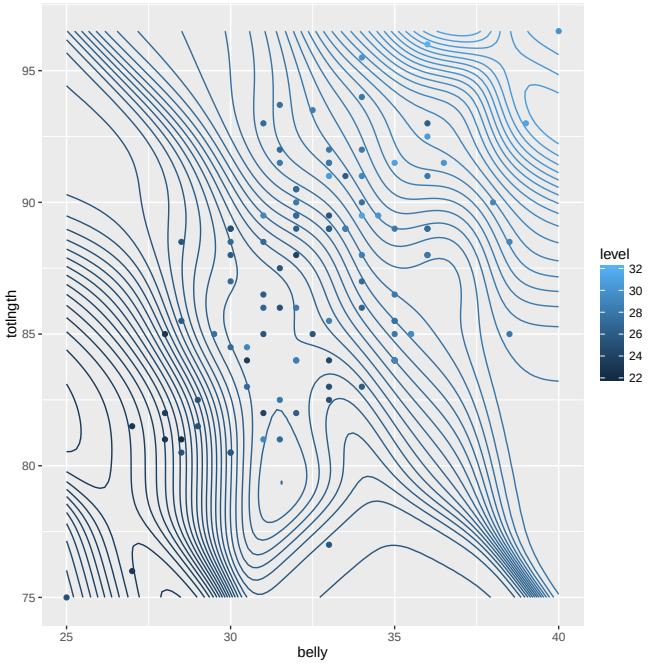
```
hdhist  <- ggplot(possum, aes(x = hdlngth)) +
  geom_histogram() + facet_grid(sex ~ Pop) +
  ggtitle("Head Lengths") + xlab("Length") + ylab("")
tothist <- ggplot(possum, aes(x = hdlngth)) +
  geom_histogram() + facet_grid(sex ~ Pop) +
  ggtitle("Total Lengths") + xlab("Length") + ylab("")
```

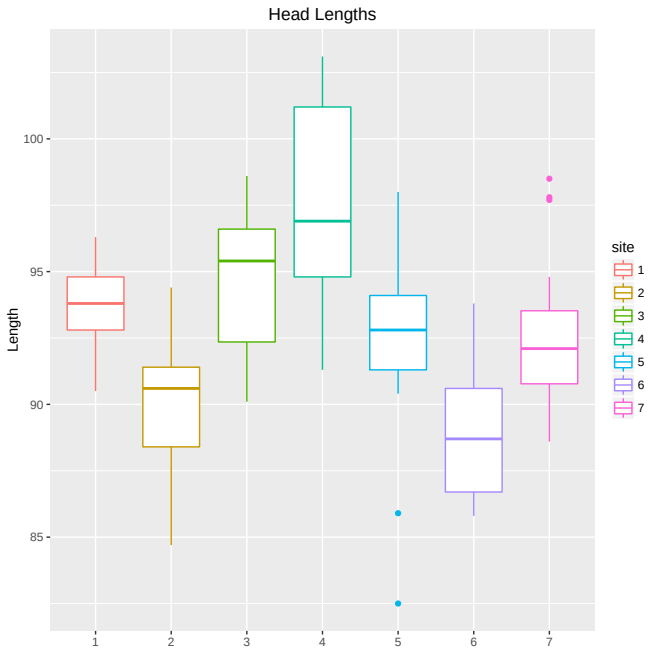
```
hdhist
tothist
```

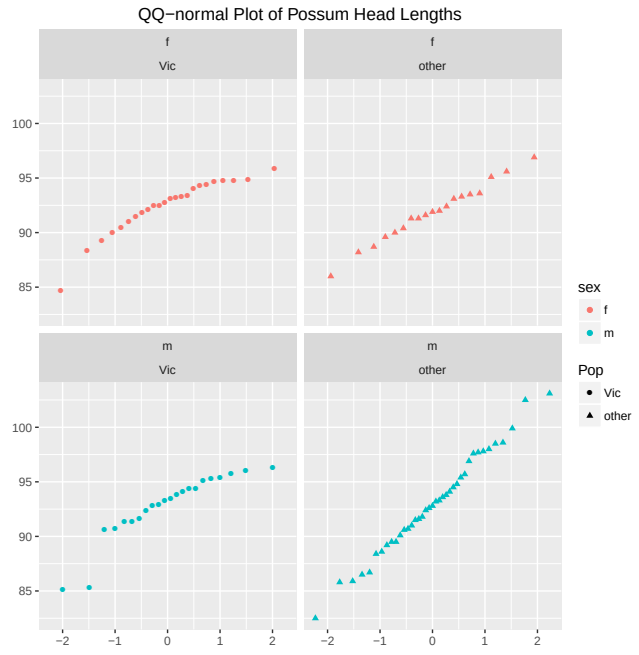












part e)

```
possumscatter <- ggplot(possum, aes(x = hdlngth, y = totlngth, color = sex, shape = Pop)) +
  stat_smooth(method = "lm", se = FALSE) +
  geom_point(alpha = 0.6) +
  facet_wrap(sex ~ Pop) +
  ggtitle("Scatter Plot of Head and Total Lengths")
possumscatter
```

Making a contour plot turns out to be a little tricky, because the data are irregularly spaced. See the appendix of these solutions for one way to do that.

```
headdboxplot <- ggplot(possum, aes(x = site, y = hdlngth, color = site)) +
  geom_boxplot() +
  ggtitle("Head Lengths") + xlab("") + ylab("Length")
headdboxplot

possumqq <- ggplot(possum, aes(sample = hdlngth, group = sex, col = sex, shape = Pop)) +
  geom_point(stat = "qq") + facet_wrap(sex ~ Pop) +
  ggtitle("QQ-normal Plot of Possum Head Lengths") + xlab("") + ylab("")
possumqq
```

Exercise 2

part a)

The standard deviation of these proportions is 0.0064. The average of these proportions is 0.4857 To calculate the expected standard deviation:

$$X \sim Binoml(n,p), n = 3900, p = 0.4857, Var(\frac{X}{n}) = \frac{1}{n^2}Var(X) = \frac{1}{n^2}(np(1-p))$$

so the expected standard deviation:

$$\sqrt{\frac{p(1-p)}{n}} = 0.008$$

Now, we want to test whether the difference between the observed standard deviation and expected standard deviation is statistically significant. so we have

$$H_0 : observedSD = expectedSD, H_1 : observedSD \neq expectedSD$$

The usual test statistics for this test is $(n-1)s^2/\sigma^2$ and this follows a χ^2_{23} distribution; the value of the test statistic in our case is 18.4. Now there are two ways to reach the conclusion:

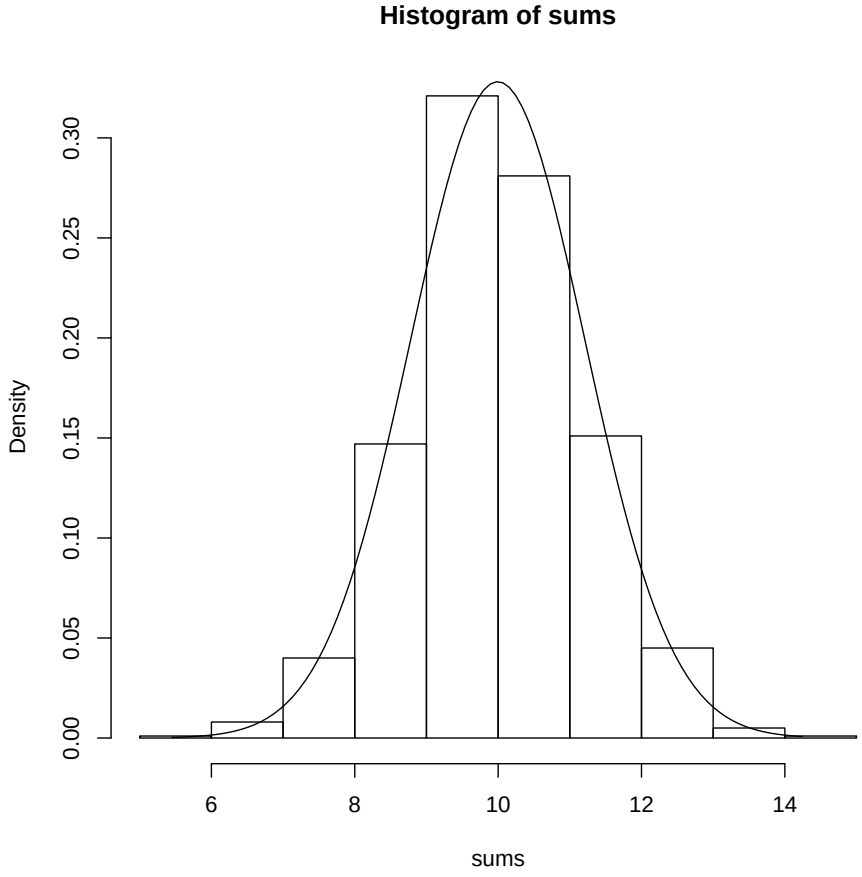
- Calculate a critical value for the test and compare our test statistic to the critical value (if our value is beyond the critical value, that indicates our data was unusual, compared to the χ^2_{23} distribution under the null hypothesis): $qchisq(0.95, df = 23) = 35.17$; and $18.4 < 35.17$ so 18.4 is not unusual for the null hypothesis. or
- Calculate a p-value for our test statistic; if the p-value is small it would be hard to get more unusual data from the null hypothesis than our test statistic; this would show that our test statistic is unusual under the null hypothesis: $1 - pchisq(18.4, df = 23) = 0.735$, and this is greater than 0.05, indicating that 18.4 is not unusal for the null hypothesis.

Either way, we conclude that there is not a significant difference between the observed SD and the expected SD.

part b)

for the code on the left side:
Line 1: randomly generate a $1000by20$ uniform valued matrix
Line 2: now sum over all the rows in the matrix
Line 3: generate a histogram of the 1000 sums
Line 4: find the mean of the 1000 sums
Line 5: find the standard deviation of the 1000 sums
Line 6: Generate 100 evenly spaced numbers between the minimum
and maximum of the sums
Line 7: Add the normal densities to the histogram with the mean and sd calculated above

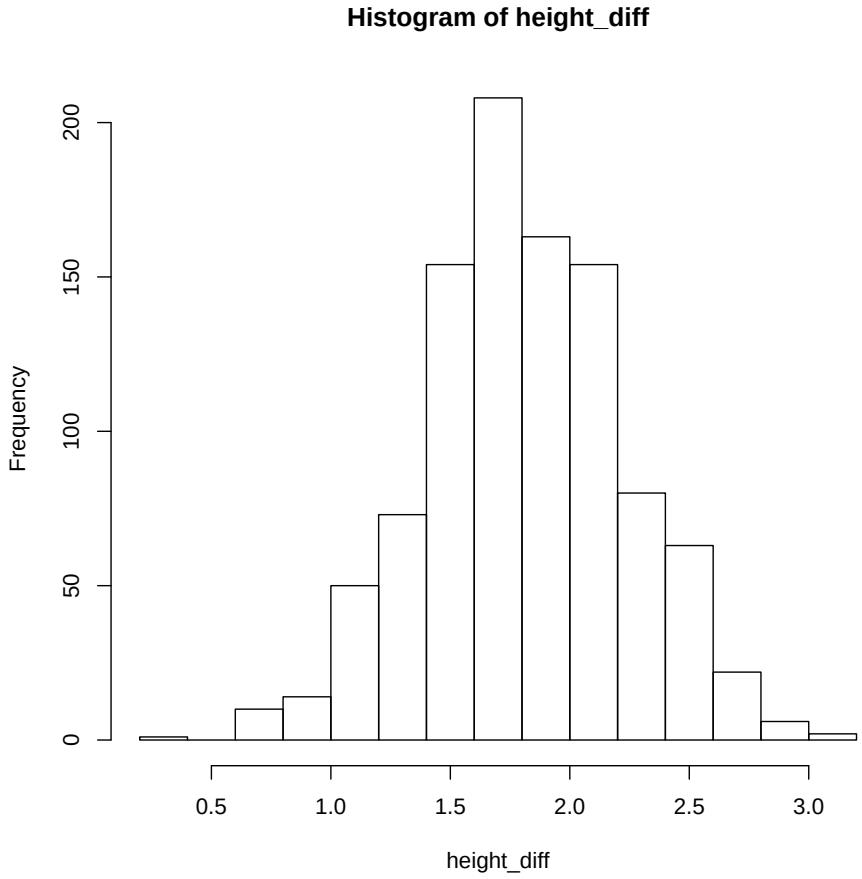
for the code on the right hand side:
first 6 lines: use a loop to generate 1000 sums of 20 random uniform variables.
the rest of the lines: the same as above



part c

```
height_diff <- NULL
for ( i in 1:1000){

mens_heights <- rnorm(n= 100,mean = 69.1,sd =2.9)
womens_heights <- rnorm(n= 100,mean = 63.7, sd = 2.7)
x <- mean(mens_heights)
y <- mean(womens_heights)
height_diff <- c(height_diff,x -y)

}

hist(height_diff)
```

The exact mean of x - y is roughly in the middle of the histogram which is around 5.4. It is roughly equal to the mean of the 1000 values of height difference (5.41) The exact standard deviation is $SD((\bar{X}) - (\bar{Y})) = \sqrt{\frac{Var((\bar{X}))+Var((\bar{Y}))}{n}} = 0.396$ which is close to 0.4 of that of the sample SD of the 1000 values of height difference. Yes, the histogram is approximately six SD wide. This is an appropriate question because the 1000 height differences are normally distributed and six SD covers around 97.5 % of the data.



part d

$$Var\left(\frac{X+Y}{2}\right) = \left(\frac{1}{2}\right)^2 Var(X) + 2 * \frac{1}{4} * Cov(X,Y) + \left(\frac{1}{2}\right)^2 Var(Y)$$

and

$$Cov(X,Y) = Cor(X,Y) * SD(X) * SD(Y) = 0.3 * 2.9 * 2.7 = 2.349$$

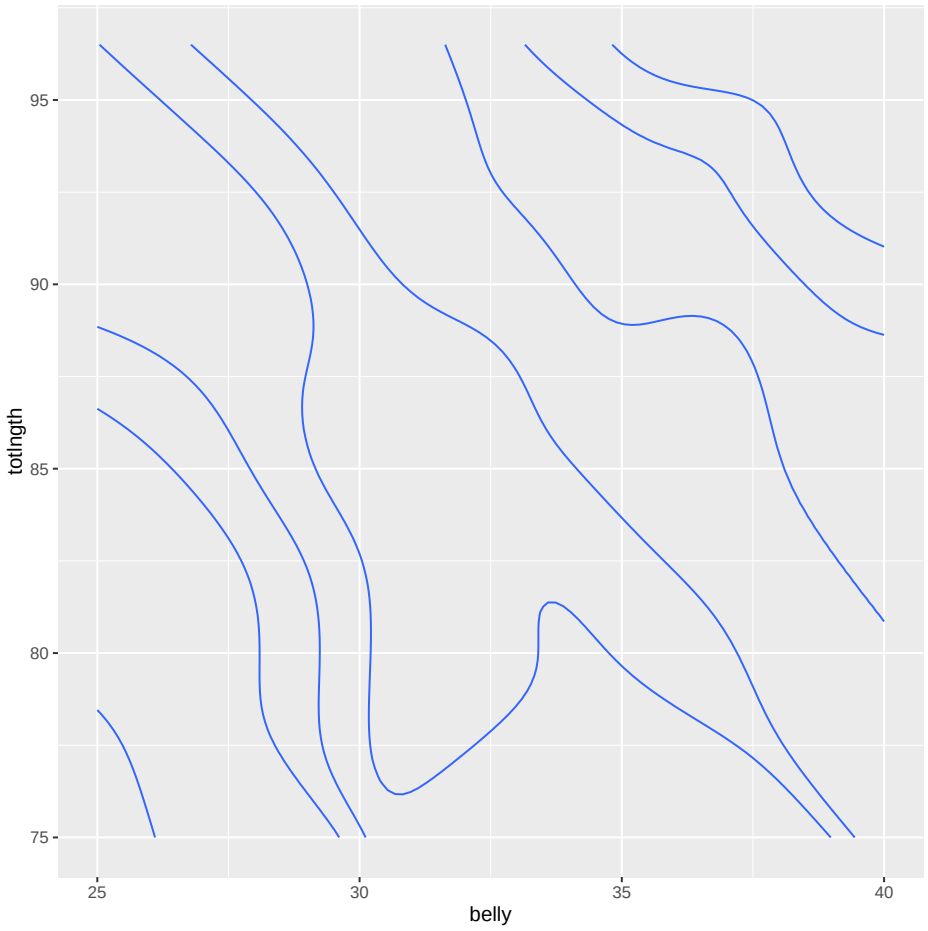
from this we can calculate that the variance equals 5.09.

Appendix: Making a contour plot from irregularly-spaced data with ggplot2 (or lattice)...

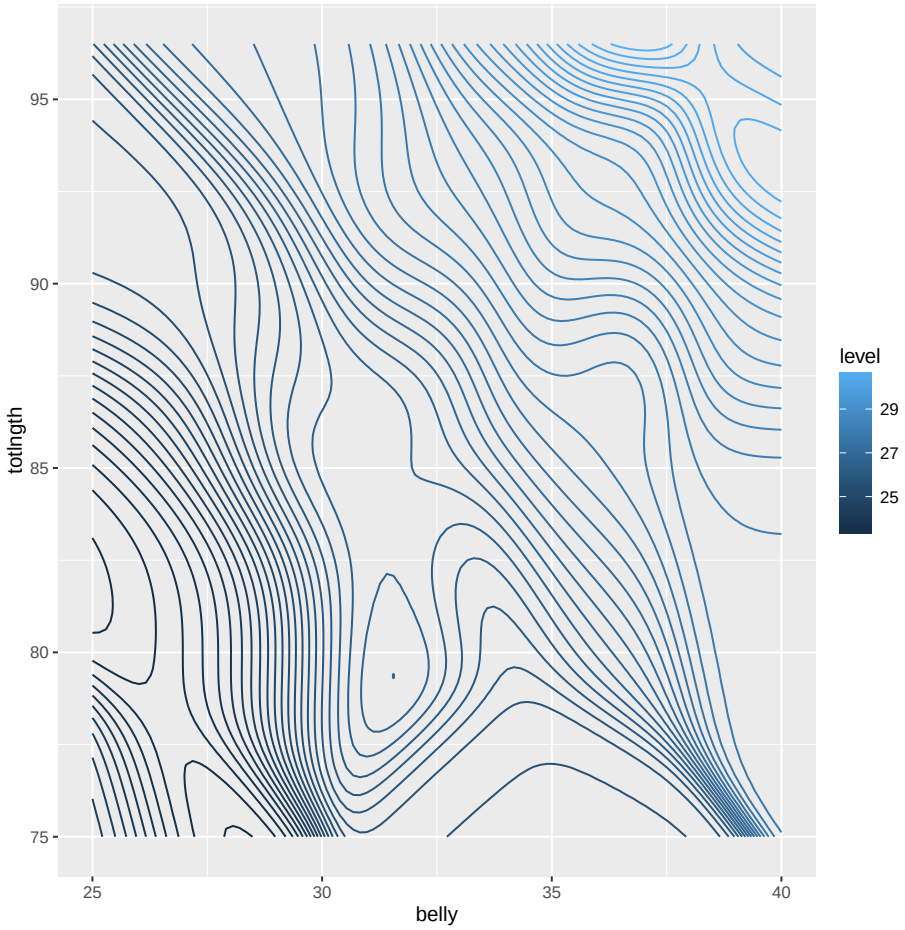
```
# Notes on exercise 12(b), Chapter 4, of the "Using R" pdf.


# The problem asks us to make a level plot or contour plot of "chest"
# as a function of "belly" and "totlngth", using the data set "possum".

# many people found "stat_contour()" (btw all "stat" and "geom" layers
# in ggplot2 are listed at http://docs.ggplot2.org/), and tried

load("usingR.RData")

library(ggplot2)

g <- ggplot(possum,aes(x=belly, y=totlngth, z=chest))

g + stat_contour()

# but they got a mysterious error message like this:
#
# Error in contourLines(x = sort(unique(data$x)), y = sort(unique(data$y)),
#  (list) object cannot be coerced to type 'double'
#
# Whatever generated this error, it does not tell us much about where the
# *real* problem lies...

# It turns out that the problem is that the "stat_contour" function
# expects to find data on a regular grid of (x,y) values, and he data
# in "possum" is not on any grid at all.

# before we can use the "stat_contour" function, we have to define a
# regular grid of x (belly) and y (totlength) values, and estimate z
# (chest) at each of those grid points.

# Suddenly, this is a rather involved little exercise!!

############################################################

# I wrote a function "gridder()" that will estimate the z values on a
# regular grid, from any old irregular (x,y,z) triples.  Before we can
# make the contour plot, we have to pass the belly, totlngth and chest
# variable through the"gridder" function.

# here's gridder():

gridder <- function(x, y, z, grid.x, grid.y, h.x, h.y) {

  # error checking and filling in default values
```

```
  n <- length(x)
  stopifnot(length(y)==n, length(z)==n)

  if(missing(grid.x)) {
    grid.x <- seq(min(x),max(x),length=n)
  }

  if(missing(grid.y)) {
    grid.y <- seq(min(y),max(y),length=n)
  }

  if(missing(h.x)) {
    h.x <- 1.06*sd(x)/n^(1/5) # silverman's univariate rule of thumb
  }

  if(missing(h.y)) {
    h.y <- 1.06*sd(y)/n^(1/5) # silverman's univariate rule of thumb
  }

  # the guts of the function come next; they make a 2-dimensional
  # kernel-smoothed regression using independent gaussian kernels in
  # each coordinate, from which we can estimate values of z above the
  # regular grid:

  grid.out <- expand.grid(grid.x,grid.y)  # make a regular grid
  x.out <- grid.out[,1]                   # extract the x values
  y.out <- grid.out[,2]                   # extract the y values
  n.out <- length(x.out)
  z.out <- c(NA,n.out)                    # set up a vector for the z values

  for (i in 1:n.out) {                    # fill in the z's using a 2-dim
    xdiffs <- x - x.out[i]                # gaussian kernel regression est
    ydiffs <- y - y.out[i]
    weights <- dnorm(xdiffs,sd=h.x)*dnorm(ydiffs,sd=h.y)
    z.out[i] <- sum(z*weights)/sum(weights)
  }

  return(data.frame(x=x.out,y=y.out,z=z.out))

}

############################################################

# now let's use the function:

gridded.data <- gridder(possum$belly, possum$totlngth, possum$chest)

names(gridded.data) <- c("belly","totlngth","chest")

g <- ggplot(gridded.data,aes(x=belly,y=totlngth,z=chest))
```

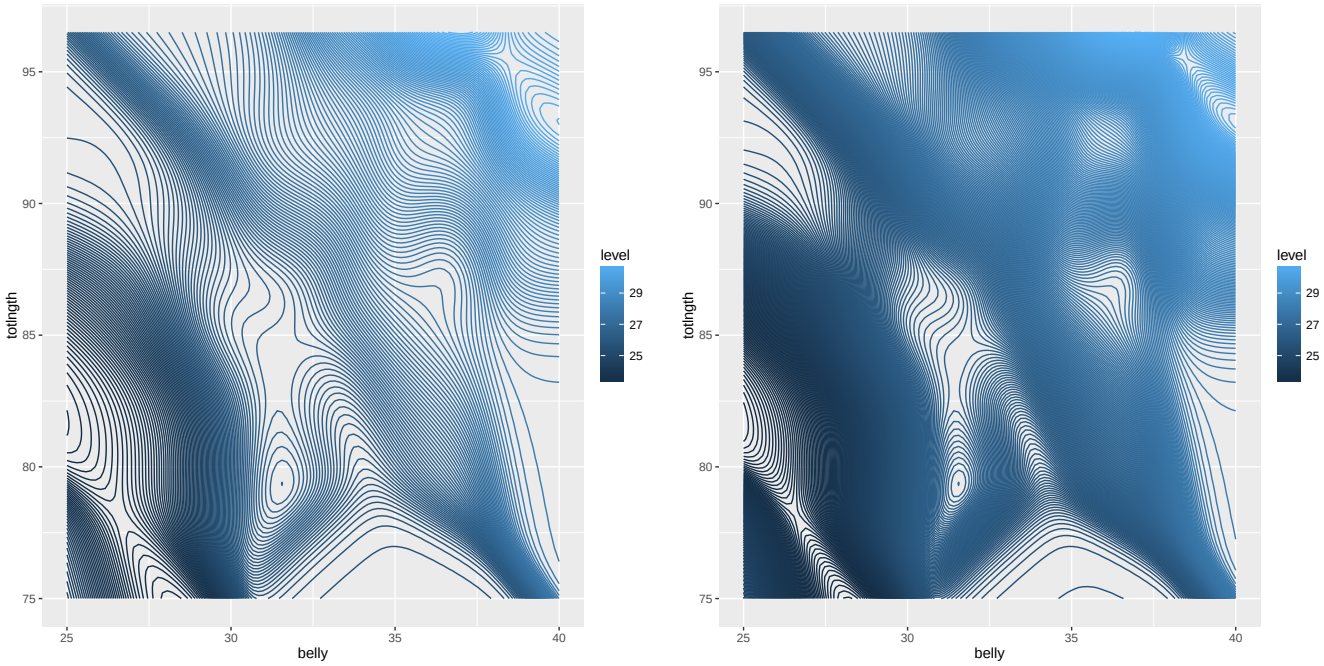
```
g + stat_contour()
```



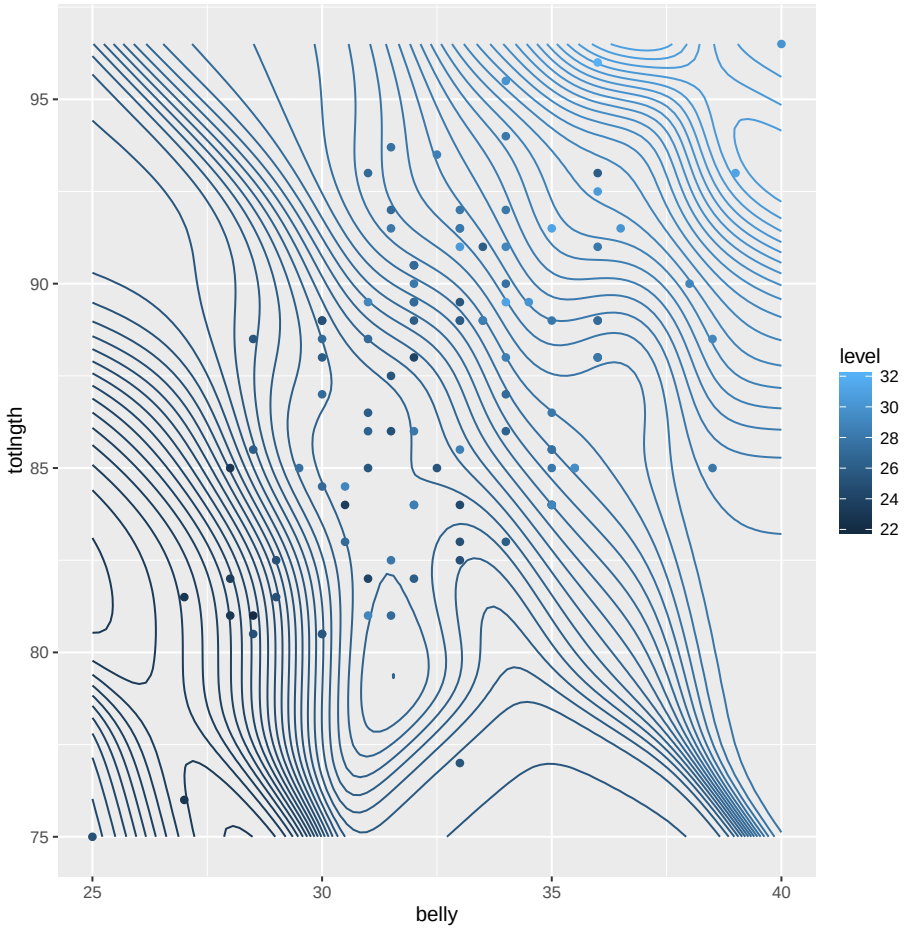

```
# it's a little hard to see what is going on with this plot, so let's
# color the grid lines and add more of them:

g + stat_contour(bins=50,aes(color=..level..))
```




```
# ("..level.." is a variable that gives the level of each contour line)

# hmm, we could do better with more contour lines.  Let's try a lot:

g + stat_contour(bins=200,aes(color=..level..))

g + stat_contour(bins=500,aes(color=..level..))
```




```
# you can see that the chest measurement generally go from low values
# in the lower left corner, to higher values in the upper right corner

# we can also plot the original data on top of the contour lines:

g + stat_contour(bins=50,aes(color=..level..)) +
  geom_point(data=possum,aes(x=belly,y=totlngth,color=chest))
```




```
# note that I have colored the original data according to chest
# measurement, on just the same scale as the countour line levels.

# we can see that the data also go from low chest measurements in the
# lower left to higher chest measurements in the upper right.  However
# we can also see that the countours show a lot of artifacting in the
# upper left and lower right, that is due to the fact that there is no
# data out there in those corners of the graph, for "gridder()" to
# work with in estimating chest measurements (the extrapolation is
# pretty bad out there).

##########################################################

# Finally, some might try to do this problem using the levelplot()
# or contourplot() functions from the "lattice" package, like this:

library(lattice)

contourplot(chest ~ belly*totlngth, data=possum)
```

```
# this doesn't produce an error message, but the plot is pretty useless!


levelplot(chest ~ belly*totlngth, data=possum)

# this basically plots each (belly,totlngth) pair and colors it
# according to the chest measurement.  It is similar to this ggplot2
# plot:

ggplot(possum,aes(x=belly,y=totlngth)) +
  geom_point(size=5,shape=15,aes(color=chest))

# note that if we use the gridded data, contourplot() and levelplot()
# produce results that are similar to the ggplot2 plots:

contourplot(chest ~ belly*totlngth, data=gridded.data)

levelplot(chest ~ belly*totlngth, data=gridded.data)

############################################################

# Feel free to use "gridder()" when you need to make other contour
# plots from irregularly spaced data...

############################################################
```