# Exploring Assumptions

## Self-test answers

- Plot histograms for the hygiene scores for the three days of the Download Festival. (For reasons that will become apparent, use *geom_histogram(aes(y = ..density..)* rather than *geom_histogram()*.).

The code for plotting a histogram of the hygiene scores on day 1 is in the book chapter. To draw the histograms for the hygiene scores for the other two days, we can simply adapt that command. To create a histogram of the day 2 data, we could execute:

```
hist.day2 <- ggplot(dlf, aes(day2)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Hygiene
score on day 2", y = "Density")

hist.day2
```

To create a histogram of the day 3 data, we could execute:

```
hist.day3 <- ggplot(dlf, aes(day3)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Hygiene
score on day 3", y = "Density")

hist.day3
```

Add normal curves to then histograms that you drew for **day2** and **day3**.

To add a normal curve to the histogram of the day 2 data, we could execute:

```
hist.day2 + stat_function(fun = dnorm, args = list(mean = mean(dlf$day2, na.rm =
TRUE), sd = sd(dlf$day2, na.rm = TRUE)), colour = "black", size = 1)
```

For the day 3, we execute:

```
hist.day3 + stat_function(fun = dnorm, args = list(mean = mean(dlf$day3, na.rm =
TRUE), sd = sd(dlf$day3, na.rm = TRUE)), colour = "black", size = 1)
```

Create Q-Q plots for the variables **day2** and **day3**.

To create a Q-Q plot of the day 2 data we could execute:

```
qqplot.day2 <- qplot(sample = dlf$day2, stat="qq")
qqplot.day2
```

To create a Q-Q plot of the day 3 scores we would execute:

```
qqplot.day3 <- qplot(sample = dlf$day3, stat="qq")
qqplot.day3
```

The resulting histograms and Q-Q plots are shown and explained in the book chapter.

- Using what you have learnt so far, obtain descriptive statistics and draw histograms of first-year exam scores, computer literacy, numeracy and lectures attended.

You could get the descriptives in several ways:

```
stat.desc(rexam[, c("exam", "computer", "lectures", "numeracy")], basic = FALSE, norm
= TRUE)
```

or

```
stat.desc(cbind(rexam$exam, rexam$computer, rexam$lectures, rexam$numeracy), basic =
FALSE, norm = TRUE)
```

or

```
describe(cbind(rexam$exam, rexam$computer, rexam$lectures, rexam$numeracy))
```

or

```
describe(rexam[, c("exam", "computer", "lectures", "numeracy")])
```

Alternatively, if you want to round to 3 decimal places, any one of these:

```
round(stat.desc(rexam[, c("exam", "computer", "lectures", "numeracy")], basic = FALSE,
norm = TRUE), digits = 3)

round(stat.desc(cbind(rexam$exam, rexam$computer, rexam$lectures, rexam$numeracy),
basic = FALSE, norm = TRUE), digits = 3)

round(describe(cbind(rexam$exam, rexam$computer, rexam$lectures, rexam$numeracy)),
digits = 3)

round(describe(rexam[, c("exam", "computer", "lectures", "numeracy")]), digits = 3)
```

The histograms are drawn by executing these commands. For previous exam performance:

```
hexam <- ggplot(rexam, aes(exam)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x = "First
Year Exam Score", y = "Density") + stat_function(fun = dnorm, args = list(mean =
mean(rexam$exam, na.rm = TRUE), sd = sd(rexam$exam, na.rm = TRUE)), colour = "red",
size = 1)

hexam
```

For computer Literacy:

```
hcomputer <- ggplot(rexam, aes(computer)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x = "Computer
Literacy", y = "Density") + stat_function(fun = dnorm, args = list(mean =
mean(rexam$computer, na.rm = TRUE), sd = sd(rexam$computer, na.rm = TRUE)), colour =
"red", size = 1)

hcomputer
```

For percentage of lectures attended:

```
hlectures <- ggplot(rexam, aes(lectures)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x =
"Percentage of Lectures Attended", y = "Density") + stat_function(fun = dnorm, args =
list(mean = mean(rexam$lectures, na.rm = TRUE), sd = sd(rexam$lectures, na.rm =
TRUE)), colour = "red", size = 1)

hlectures
```

For numeracy:

```
hnumeracy <- ggplot(rexam, aes(numeracy)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x =
"Numeracy", y = "Density") + stat_function(fun = dnorm, args = list(mean =
mean(rexam$numeracy, na.rm = TRUE), sd = sd(rexam$numeracy, na.rm = TRUE)), colour =
"red", size = 1)

hnumeracy
```

- Repeat these analyses for the computer literacy and percentage of lectures attended and interpret the results.

You could use *by()* to get the descriptives for computer literacy and percentage of lectures attended split by **uni**:

```
by(cbind(data=rexam$computer, data=rexam$lectures), rexam$uni, describe)
```

```
INDICES: Duncetown
  var  n  mean    sd  median trimmed   mad min max  range skew kurtosis  se
1   1 50 50.26  8.07   49.0    50.05  8.90  35  67     32  0.21    -0.51 1.14
2   2 50 56.26 23.77   60.5    56.90 20.02   8 100     92 -0.29    -0.38 3.36
-------------------------------------------------------------------------------------
INDICES: Sussex
  var  n  mean    sd  median  trimmed  mad  min max range  skew kurtosis  se
1   1 50 51.16  8.51  54.00   51.62  5.93 27.0  73  46.0 -0.51     1.38 1.20
2   2 50 63.27 18.97  65.75   63.99 20.76 12.5 100  87.5 -0.34    -0.22 2.68
```

You could also use *stat.desc()*:

```
by(rexam[, c("computer", "lectures")], rexam$uni, stat.desc, basic = FALSE, norm = TRUE)
```

```
rexam$uni: Duncetown University
               computer    lectures
median       49.0000000  60.5000000
mean         50.2600000  56.2600000
SE.mean       1.1410021   3.3619491
CI.mean.0.95  2.2929295   6.7560897
var          65.0942857 565.1351020
std.dev       8.0681030  23.7725704
coef.var      0.1605273   0.4225484
skewness      0.2121230  -0.2904291
skew.2SE      0.3150960  -0.4314149
kurtosis     -0.6779460  -0.5634849
kurt.2SE     -0.5121147  -0.4256518
normtest.W    0.9776352   0.9697414
normtest.p    0.4571125   0.2259081
-------------------------------------------------------------------------------------
rexam$uni: Sussex University
                computer    lectures
median       54.00000000  65.7500000
mean         51.16000000  63.2700000
SE.mean       1.20284018   2.6827191
CI.mean.0.95  2.41719783   5.3911258
var          72.34122449 359.8490816
std.dev       8.50536445  18.9696885
coef.var      0.16625028   0.2998212
skewness     -0.50635339  -0.3429407
skew.2SE     -0.75215735  -0.5094177
kurtosis      0.96404781  -0.4233827
kurt.2SE      0.72823358  -0.3198197
normtest.W    0.94392282   0.9817164
normtest.p    0.01931478   0.6262669
```

The output is split into two sections: first, the results for students at Duncetown University, then the results for those attending Sussex University. Variable 1 id **computer** and variable 2 id **lectures**. From these tables it is clear that Sussex and Duncetown students scored similarly on computer literacy (both means are very similar). Sussex students attended slightly more lectures (63.27%) than their Duncetown counterparts (56.26%).

To plot histograms of the computer literacy and number of lectures attended for each university, we would again create two dataframes, one for each university:

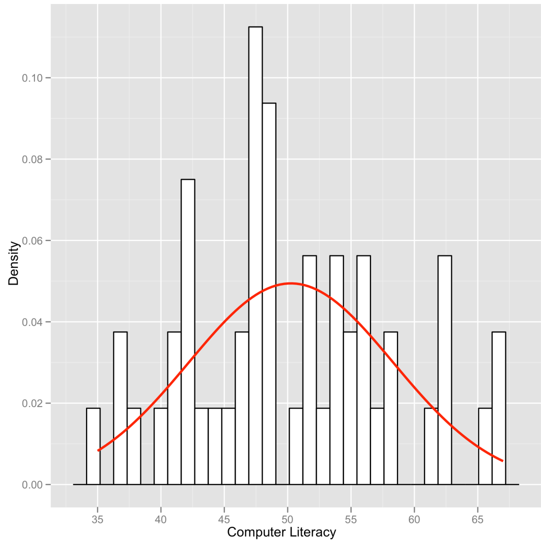
```
dunceData<-subset(rexam, rexam$uni=="Duncetown University")

sussexData<-subset(rexam, rexam$uni=="Sussex University")
```

Therefore, to plot a histogram for computer literacy for Duncetown University we would execute:

```
hist.computer.duncetown <- ggplot(dunceData, aes(computer)) + opts(legend.position =
"none") + geom_histogram(aes(y = ..density..), fill = "white", colour = "black") +
labs(x = "Computer Literacy", y = "Density") + stat_function(fun=dnorm, args=list(mean
= mean(dunceData$computer, na.rm = TRUE), sd = sd(dunceData$computer, na.rm = TRUE)),
colour = "red", size=1)
```

```
hist.computer.duncetown
```

The resulting histogram should look like this:

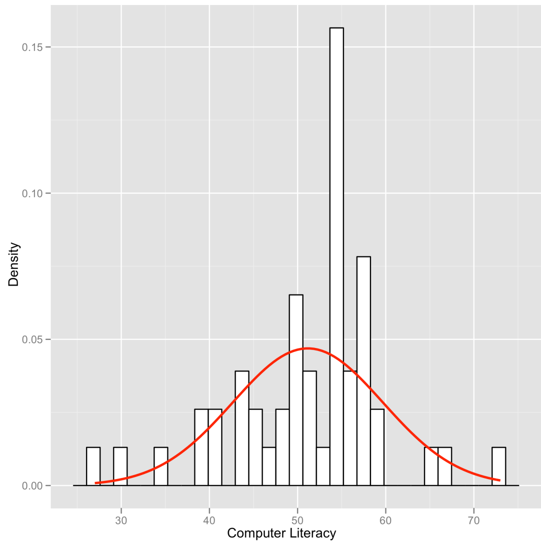



To plot a histogram  for computer literacy for Sussex University we would execute:

```
hist.computer.sussex <- ggplot(sussexData, aes(computer)) + opts(legend.position =
"none") + geom_histogram(aes(y = ..density..), fill = "white", colour = "black") +
labs(x = "Computer Literacy", y = "Density") + stat_function(fun=dnorm, args=list(mean
= mean(sussexData$computer, na.rm = TRUE), sd = sd(sussexData$computer, na.rm =
TRUE)), colour = "red", size=1)
```

```
hist.computer.sussex
```

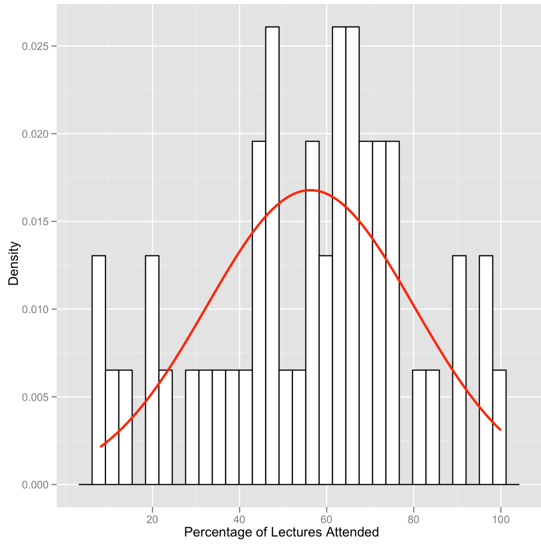
The resulting histogram should look like this:




To plot a histogram  for percentage of lectures attended for Duncetown University we would execute:

```
hist.lectures.duncetown <- ggplot(dunceData, aes(lectures)) + opts(legend.position = "none") + geom_histogram(aes(y = ..density..), fill = "white", colour = "black") + labs(x = "Percentage of Lectures Attended", y = "Density") + stat_function(fun=dnorm, args=list(mean = mean(dunceData$lectures, na.rm = TRUE), sd = sd(dunceData$lectures, na.rm = TRUE)), colour = "red", size=1)

hist.lectures.duncetown
```

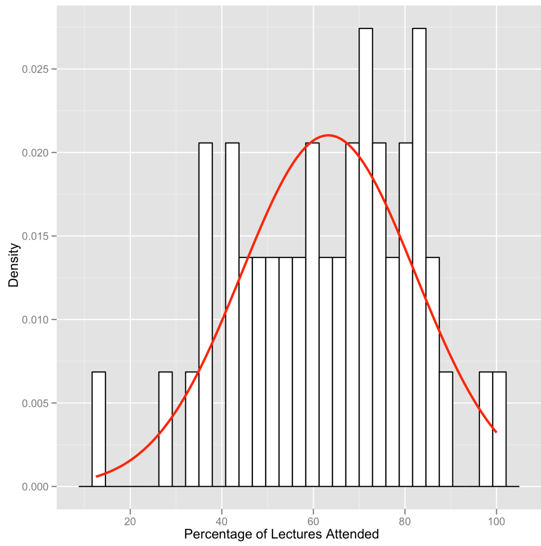
The resulting histogram should look like this:




To plot a histogram for percentage of lectures attended for Sussex University we would execute:

```
hist.lectures.sussex <- ggplot(sussexData, aes(lectures)) + opts(legend.position = "none") + geom_histogram(aes(y = ..density..), fill = "white", colour = "black") + labs(x = "Percentage of Lectures Attended", y = "Density") + stat_function(fun=dnorm, args=list(mean = mean(sussexData$lectures, na.rm = TRUE), sd = sd(sussexData$lectures, na.rm = TRUE)), colour = "red", size=1)

hist.lectures.sussex
```

The resulting histogram should look like this:



All of the distributions look fairly normal. The only exception is the computer literacy scores for the Sussex students. This is a fairly flat distribution apart from a huge peak between 50% and 60%. It's slightly heavy-tailed (right at the very ends of the curve the bars come above the line) and very pointy. This suggests positive kurtosis.

- Have a go at creating similar variables **logday2** and **logday3** for the day 2 and day 3 data. Plot histograms of the transformed scores for all three days.

To transform the variable **day2** and create a new variable **logday2**, we would execute:

```
dlf$logday2 <- log(dlf$day2 + 1)
```

To transform the variable **day3** and create a new variable **logday3**, we would execute

```
dlf$logday3 <- log(dlf$day3 + 1)
```

If you then execute:

```
dlf
```

you will see the new dataframe with the added variables. I have pasted in a section below:

```
    ticknumb gender day1 day2 day3    logday1    logday2    logday3
1       2111      0 2.64 1.35 1.61 1.29198368 0.85441533 0.95935022
2       2229      1 0.97 1.41 0.29 0.67803354 0.87962675 0.25464222
3       2338      0 0.84   NA   NA 0.60976557         NA         NA
4       2384      1 3.03   NA   NA 1.39376638         NA         NA
5       2401      1 0.88 0.08   NA 0.63127178 0.07696104         NA
6       2405      0 0.85   NA   NA 0.61518564         NA         NA
7       2467      1 1.56   NA   NA 0.94000726         NA         NA
8       2478      1 3.02   NA   NA 1.39128190         NA         NA
9       2490      0 2.29   NA   NA 1.19088756         NA         NA
10      2504      1 1.11 0.44 0.55 0.74668795 0.36464311 0.43825493
11      2509      0 2.17   NA   NA 1.15373159         NA         NA
12      2510      1 0.82 0.20 0.47 0.59883650 0.18232156 0.38526240
```

To create the histogram for **logday1** we could execute:

```
hist.logday1 <- ggplot(dlf, aes(logday1)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Hygiene
score on day 1", y = "Density") + stat_function(fun = dnorm, args = list(mean =
mean(dlf$logday1, na.rm = TRUE), sd = sd(dlf$logday1, na.rm = TRUE)), colour = "red",
size = 1)

hist.logday1
```

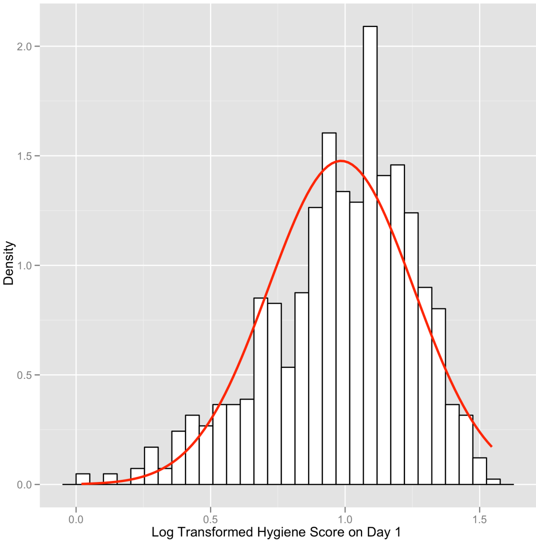
To create the histogram for **logday2** we could execute:

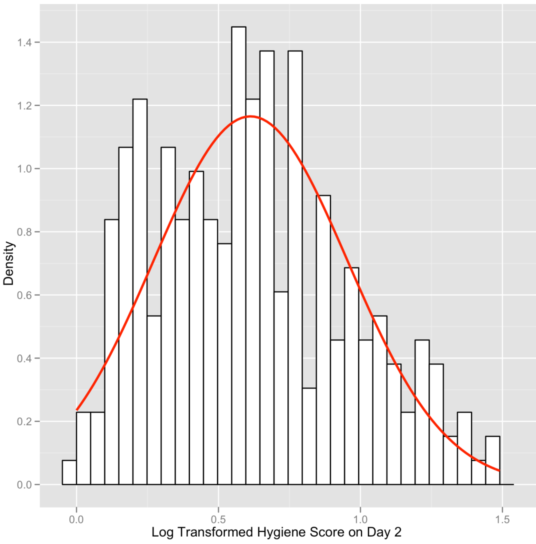
```
hist.logday2 <- ggplot(dlf, aes(logday2)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Hygiene
score on day 2", y = "Density") + stat_function(fun = dnorm, args = list(mean =
mean(dlf$logday2, na.rm = TRUE), sd = sd(dlf$logday2, na.rm = TRUE)), colour = "red",
size = 1)

hist.logday2
```

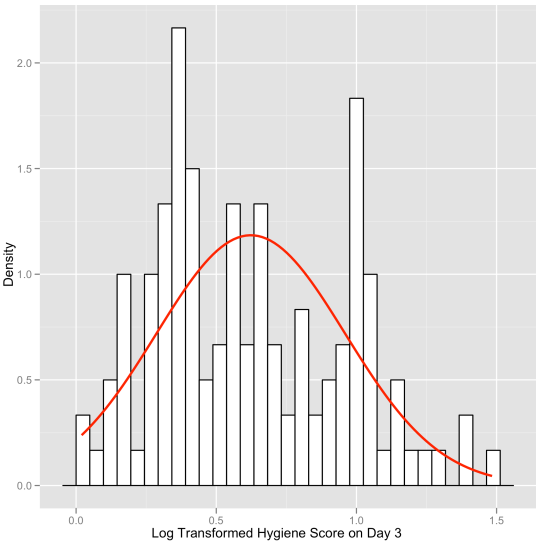
To create the histogram for **logday3** we could execute:

```
hist.logday3 <- ggplot(dlf, aes(logday3)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Hygiene
score on day 3", y = "Density") + stat_function(fun = dnorm, args = list(mean =
mean(dlf$logday3, na.rm = TRUE), sd = sd(dlf$logday3, na.rm = TRUE)), colour = "red",
size = 1)

hist.logday3
```

The resulting graphs should look like this:


Density
Log Transformed Hygiene Score on Day 1


Density
Log Transformed Hygiene Score on Day 2


Density
Log Transformed Hygiene Score on Day 3

- Repeat this process for **day2** and **day3** to create variables called **sqrtday2** and **sqrtday3**. Plot histograms of the transformed scores for all three days.

To do a square root transformation on all three days of the Download Festival we would execute the following commands:

```
dlf$sqrtday1 <- sqrt(dlf$day1)
dlf$sqrtday2 <- sqrt(dlf$day2)
dlf$sqrtday3 <- sqrt(dlf$day3)
```

Executing these commands will produce three new columns in the Download Festival dataframe called *sqrtday1*, *sqrtday2* and *sqrtday3*.
To plot histograms of the transformed scores for all three days, we would execute, for **sqrtday1**:

```
hist.sqrtday1 <- ggplot(dlf, aes(sqrtday1)) + opts(legend.position = "none") + geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Square Root of Hygiene Score on Day 1", y = "Density") + stat_function(fun = dnorm, args = list(mean = mean(dlf$sqrtday1, na.rm = TRUE), sd = sd(dlf$sqrtday1, na.rm = TRUE)), colour = "red", size = 1)

hist.sqrtday1
```

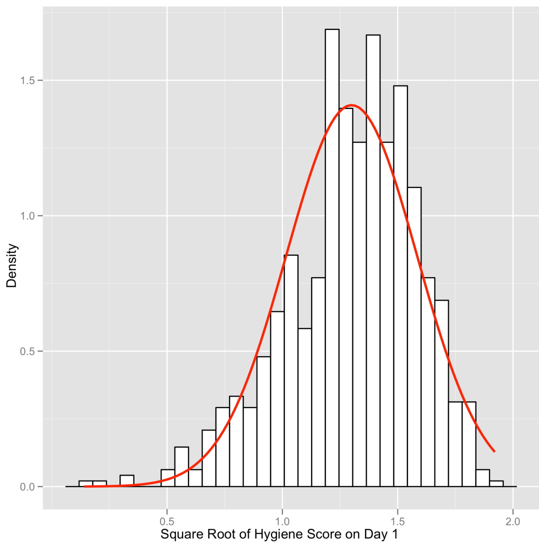
for **sqrtday2**:

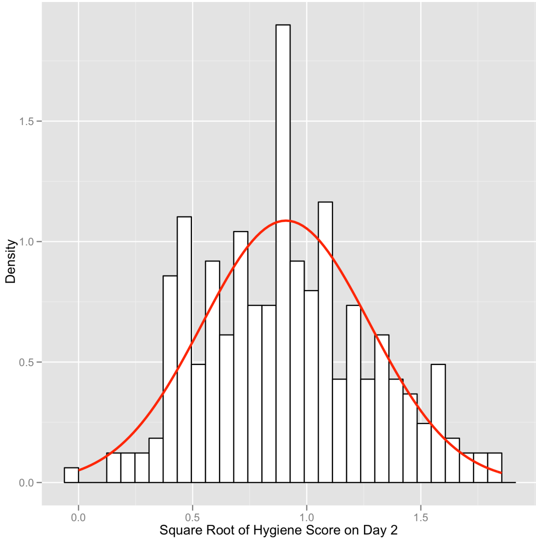
```
hist.sqrtday2 <- ggplot(dlf, aes(sqrtday2)) + opts(legend.position = "none") + geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Square Root of Hygiene Score on Day 2", y = "Density") + stat_function(fun = dnorm, args = list(mean = mean(dlf$sqrtday2, na.rm = TRUE), sd = sd(dlf$sqrtday2, na.rm = TRUE)), colour = "red", size = 1)

hist.sqrtday2
```

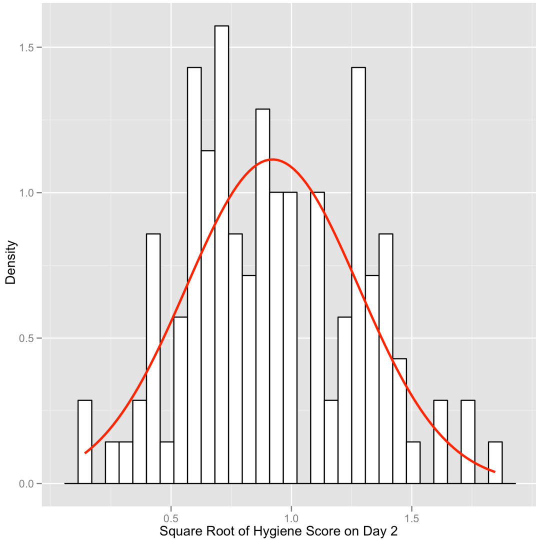
for **sqrtday3**:

```
hist.sqrtday3 <- ggplot(dlf, aes(sqrtday3)) + opts(legend.position = "none") + geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Square Root of Hygiene Score on Day 2", y = "Density") + stat_function(fun = dnorm, args = list(mean = mean(dlf$sqrtday3, na.rm = TRUE), sd = sd(dlf$sqrtday3, na.rm = TRUE)), colour = "red", size = 1)

hist.sqrtday3
```

The resulting histograms should look like this:






- Repeat this process for **day2** and **day3**. Plot histograms of the transformed scores for all three days.

To do a reciprocal transformation on the data from all three days of the Download Festival we would execute the following commands:

```
dlf$recday1 <- 1/(dlf$day1 + 1)
dlf$recday2 <- 1/(dlf$day2 + 1)
dlf$recday3 <- 1/(dlf$day3 + 1)
```

To plot histograms of the reciprocal transformed scores for all three days we would execute, for **recday1**:

```
hist.recday1 <- ggplot(dlf, aes(recday1)) + opts(legend.position = "none") +
geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Reciprocal
of of Hygiene Score on Day 1", y = "Density") + stat_function(fun = dnorm, args =
list(mean = mean(dlf$recday1, na.rm = TRUE), sd = sd(dlf$recday1, na.rm = TRUE)),
colour = "red", size = 1)

hist.recday1
```

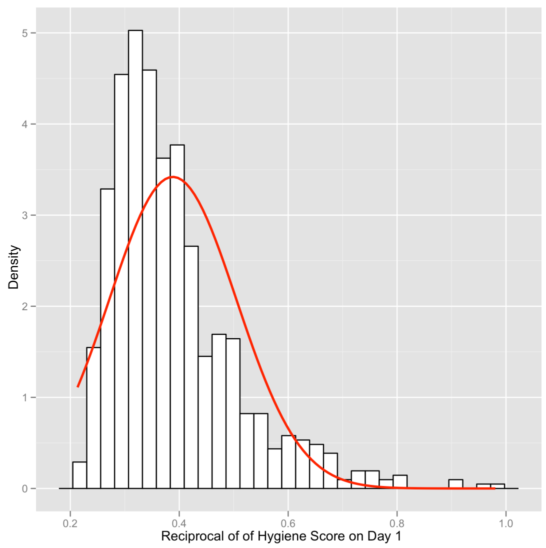
for **recday2**:

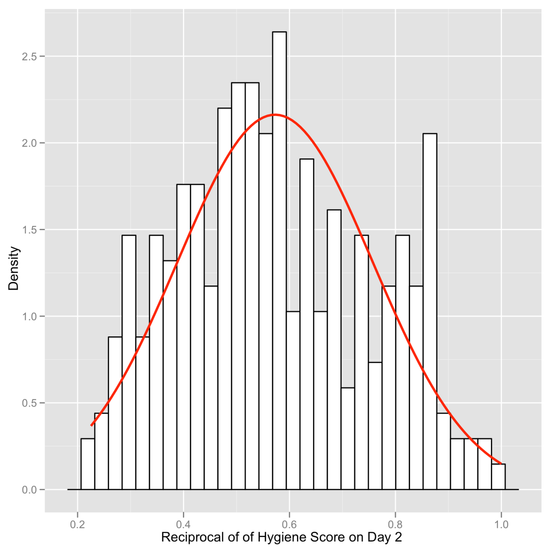
```
hist.recday2 <- ggplot(dlf, aes(recday2)) + opts(legend.position = "none") + geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Reciprocal of of Hygiene Score on Day 2", y = "Density") + stat_function(fun = dnorm, args = list(mean = mean(dlf$recday2, na.rm = TRUE), sd = sd(dlf$recday2, na.rm = TRUE)), colour = "red", size = 1)

hist.recday2
```

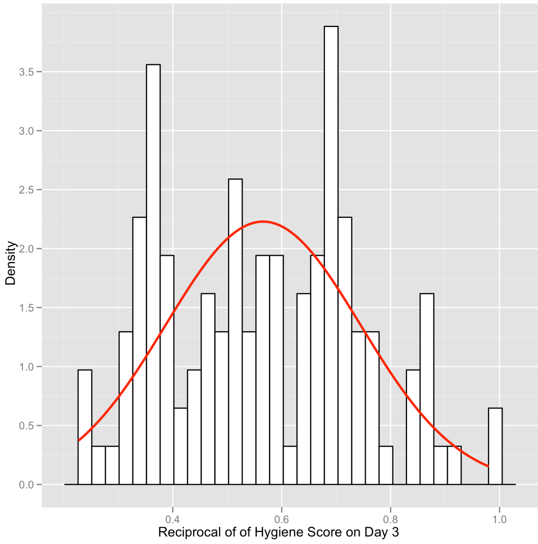
for **recday3**:

```
hist.recday3 <- ggplot(dlf, aes(recday3)) + opts(legend.position = "none") + geom_histogram(aes(y=..density..), colour="black", fill="white") + labs(x="Reciprocal of of Hygiene Score on Day 3", y = "Density") + stat_function(fun = dnorm, args = list(mean = mean(dlf$recday3, na.rm = TRUE), sd = sd(dlf$recday3, na.rm = TRUE)), colour = "red", size = 1)

hist.recday3
```

The resulting histograms should look like this:






## Please Sir, can I have some more … Hartley's $F_{max}$?

Critical values for Hartley's test ($\alpha$ = .05).

| ($n$ – 1) per group | Number of Variances Compared | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 2 | 39.00 | 87.50 | 142.00 | 202.00 | 266.00 | 333.00 | 403.00 | 475.00 | 550.00 | 626.00 | 704.00 |
| 3 | 15.40 | 27.80 | 39.20 | 50.70 | 62.00 | 72.90 | 83.50 | 93.90 | 104.00 | 114.00 | 124.00 |
| 4 | 9.60 | 15.50 | 20.60 | 25.20 | 29.50 | 33.60 | 37.50 | 41.40 | 44.60 | 48.00 | 51.40 |
| 5 | 7.15 | 10.80 | 13.70 | 16.30 | 18.70 | 20.80 | 22.90 | 24.70 | 26.50 | 28.20 | 29.90 |
| 6 | 5.82 | 8.38 | 10.40 | 12.10 | 13.70 | 15.00 | 16.30 | 17.50 | 18.60 | 19.70 | 20.70 |
| 7 | 4.99 | 6.94 | 8.44 | 9.70 | 10.80 | 11.80 | 12.70 | 13.50 | 14.30 | 15.10 | 15.80 |
| 8 | 4.43 | 6.00 | 7.18 | 8.12 | 9.03 | 9.80 | 10.50 | 11.10 | 11.70 | 12.20 | 12.70 |
| 9 | 4.03 | 5.34 | 6.31 | 7.11 | 7.80 | 8.41 | 8.95 | 9.45 | 9.91 | 10.30 | 10.70 |
| 10 | 3.72 | 4.85 | 5.67 | 6.34 | 6.92 | 7.42 | 7.87 | 8.28 | 8.66 | 9.01 | 9.34 |
| 12 | 3.28 | 4.16 | 4.79 | 5.30 | 5.72 | 6.09 | 6.42 | 6.72 | 7.00 | 7.25 | 7.48 |
| 15 | 2.86 | 3.54 | 4.01 | 4.37 | 4.68 | 4.95 | 5.19 | 5.40 | 5.59 | 5.77 | 5.93 |
| 20 | 2.46 | 2.95 | 3.29 | 3.54 | 3.76 | 3.94 | 4.10 | 4.24 | 4.37 | 4.49 | 4.59 |
| 30 | 2.07 | 2.40 | 2.61 | 2.78 | 2.91 | 3.02 | 3.12 | 3.21 | 3.29 | 3.36 | 3.39 |
| 60 | 1.67 | 1.85 | 1.96 | 2.04 | 2.11 | 2.17 | 2.22 | 2.26 | 2.30 | 2.33 | 2.36 |
| x | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

# Smart Alex's solutions

## Task 1

- Using the **ChickFlick.dat** data, check the assumptions of normality and homogeneity of variance for the two films (ignore gender): are the assumptions met?

First of all read in the **chickFlick.dat** data:

```
chickFlick <- read.delim(file="ChickFlick.dat", header=TRUE)
```

If we want to obtain separate descriptive statistics for each of the films, we can use the *by()* function.

To get descriptive statistics for the variable **arousal** for each film separately using *describe*, we could execute:

```
by(chickFlick$arousal, chickFlick$film, describe)
```

```
chickFlick$film: Bridget Jones' Diary
  var n  mean  sd   median trimmed  mad  min max  range  skew  kurtosis   se
   1  20 14.8  5.73  15     15.06    5.93  3   24   21    -0.32  -0.73      1.28
------------------------------------------------------------------------------------
chickFlick$film: Memento
  var  n  mean   sd  median  trimmed  mad min max range skew kurtosis   se
   1   20 25.25 7.13   24.5    25.25   8.9 14  37   23   0.03  -1.26    1.59
```

Or we could use stat.desc (I have edited the output below to save space):

```
by(chickFlick$arousal, chickFlick$film, stat.desc, basic = FALSE, norm = TRUE)
```

```
chickFlick$film: Bridget Jones' Diary
median   mean  SE.mean  var    std.dev  coef.var  skewness   skew.2SE    kurtosis     kurt.2SE
15.000  14.800   1.281  32.800  5.727     0.387    -0.323    -0.315       -0.726      -0.366
------------------------------------------------------------------------------------------------
chickFlick$film: Memento
median  mean    SE.mean  var    std.dev coef.var skewness    skew.2SE     kurtosis      kurt.2SE
24.500  25.250   1.594   50.829  7.129   0.282     0.034      0.033          -1.259       -0.634
```

The values of skewness and kurtosis should be zero in a normal distribution. Positive values of skewness indicate a pile-up of scores on the left of the distribution, whereas negative values indicate a pile-up on the right. Positive values of kurtosis indicate a pointy and heavy-tailed distribution, whereas negative values indicate a flat and light-tailed distribution. The further the value is from zero, the more likely it is that the data are not normally distributed. *For Bridget Jones's Diary* the skew value is fairly close to zero (which is good) and kurtosis is a little negative. For *Memento*, the skew value is very close to zero but the kurtosis is a more negative than for *Bridget Jones's Diary*.

For the arousal scores, the values of *skew.2SE* are –0.315 for *Bridget Jones' Diary* and 0.033 for *Memento*, indicating no significant skewness in either film; the values of *kurt.2SE* are –0.366 for *Bridget Jones's Diary* and –0.634 for *Memento*, which also show no significant kurtosis.

Next, we could run Levene's test to check for homogeneity of variance by executing:

```
leveneTest(chickFlick$arousal, chickFlick$film, center=median)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  1  1.8051 0.1871
      38
```

The output for Levene's test shows that the variances of arousal for the two films were not significantly different, $F(1, 38) = 1.81$, $p > .05$.

To compute the Shapiro–Wilk test for arousal split by film we would execute:

```
by(data=chickFlick$arousal, INDICES=chickFlick$film, FUN=shapiro.test)
```

```
chickFlick$film: Bridget Jones' Diary

      Shapiro-Wilk normality test

data:  dd[x, ]
W = 0.9716, p-value = 0.7876

------------------------------------------------------------------------------------
chickFlick$film: Memento

      Shapiro-Wilk normality test

data:  dd[x, ]
W = 0.9604, p-value = 0.5516
```

Remember that a significant value (*p*-value less than .05) indicates a deviation from normality. For both *Bridget Jones's Diary* and *Memento*, the Shapiro–Wilk test is nonsignificant, indicating that both distributions are not significantly different from a normal distribution.

Finally, let's plot some histograms. We want to plot separate histograms for the two films (*Bridget Jones's Diary* and *Memento*), therefore we need to use the *subset()* function to create two new dataframes:

```
bridgetData<-subset(chickFlick, chickFlick$film=="Bridget Jones' Diary")
mementoData<-subset(chickFlick, chickFlick$film=="Memento")
```

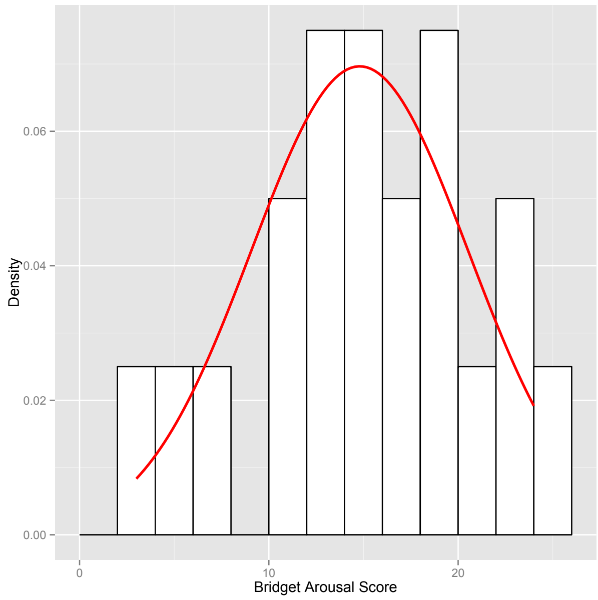
Then we can plot the histograms, remembering to specify the correct new dataframe. For *Bridget Jones's Diary* we would execute:

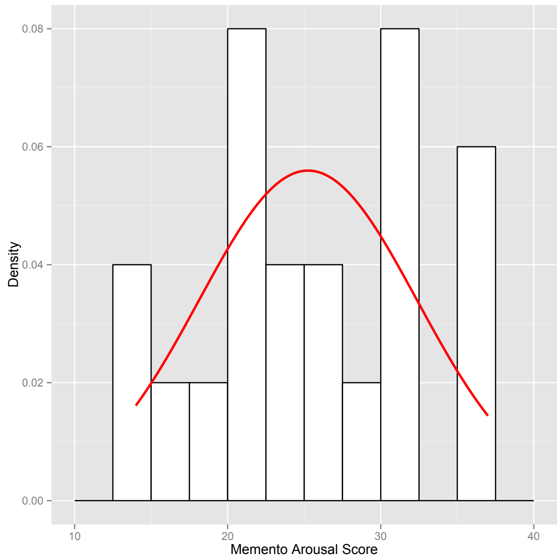
```
hist.arousal.bridget <- ggplot(bridgetData,
 aes(arousal)) + opts(legend.position = "none") +
 geom_histogram(aes(y = ..density..), fill =
 "white", colour = "black", binwidth = 2) +
 labs(x = "Arousal Score", y = "Density") +
 stat_function(fun=dnorm, args=list(mean =
 mean(bridgetData$arousal, na.rm = TRUE), sd =
 sd(bridgetData$arousal, na.rm = TRUE)), colour =
 "red", size=1)
hist.arousal.bridget
```

For *Memento* we would execute:

```
hist.arousal.memento <- ggplot(mementoData,
 aes(arousal)) + opts(legend.position = "none") +
 geom_histogram(aes(y = ..density..), fill =
 "white", colour = "black", binwidth = 2.5) +
 labs(x = "Arousal Score", y = "Density") +
 stat_function(fun=dnorm, args=list(mean =
 mean(mementoData$arousal, na.rm = TRUE), sd =
 sd(mementoData$arousal, na.rm = TRUE)), colour =
 "red", size=1)
hist.arousal.memento
```

The resulting histograms should look like this:




# Task 2

- The numeracy scores were positively skewed in the **RExam.dat** data (see Figure 5.5). Transform these data using one of the transformations described in this chapter: do the data become normal?

Read in the **R** exam data:

```
rexam <- read.table(file="rexam.dat", header=TRUE)
```

Set the variable **uni** to be a factor:

```
rexam$uni<-factor(rexam$uni, levels = c(0:1), labels = c("Duncetown University",
 "Sussex University"))
```

To plot a histogram of the *numeracy* scores, we would execute:

```
hist.numeracy <- ggplot(rexam, aes(numeracy)) +
             opts(legend.position = "none")+
              geom_histogram(aes(y=..density..),
               colour="black", fill="white", binwidth = 1.5) +
             labs(x="Numeracy scores", y = "Density") +
             stat_function(fun=dnorm,
               args=list (mean=mean (rexam$numeracy, na.rm=TRUE),
                sd=sd(rexam$numeracy, na.rm=TRUE) ),
                colour="black", size=1)
hist.numeracy
```

To log-transform the *numeracy* variable and create a new variable *lognumeracy*, we would execute:

```
rexam$lognumeracy <- log(rexam$numeracy)
```

To create a histogram for *lognumeracy* we would execute:

```
hist.lognumeracy <- ggplot(rexam, aes(lognumeracy)) +
             opts(legend.position = "none")+
              geom_histogram(aes(y=..density..),
               colour="black", fill="white", binwidth = .7) +
             labs(x="Log Transformed Numeracy scores", y = "Density") +
             stat_function(fun=dnorm,
               args=list (mean=mean (rexam$lognumeracy, na.rm=TRUE),
                sd=sd(rexam$lognumeracy, na.rm=TRUE) ),
                colour="black", size=1)
 hist.lognumeracy
```

To do a square root transformation, we run through the same process, by using a name such as *sqrtnumeracy*:

```
rexam$sqrtnumeracy <- sqrt(rexam$numeracy)
```

To create a histogram for *sqrtnumeracy* we would execute:

```
hist.sqrtnumeracy <- ggplot(rexam, aes(sqrtnumeracy)) +
             opts(legend.position = "none")+
              geom_histogram(aes(y=..density..),
               colour="black", fill="white", binwidth = .3) +
             labs(x="Square Root of Numeracy scores", y = "Density") +
             stat_function(fun=dnorm,
               args=list (mean=mean (rexam$sqrtnumeracy, na.rm=TRUE),
                sd=sd(rexam$sqrtnumeracy, na.rm=TRUE) ),
                colour="black", size=1)
hist.sqrtnumeracy
```

To do a reciprocal transformation on the numeracy data, we don't use a function, we use an arithmetic expression:

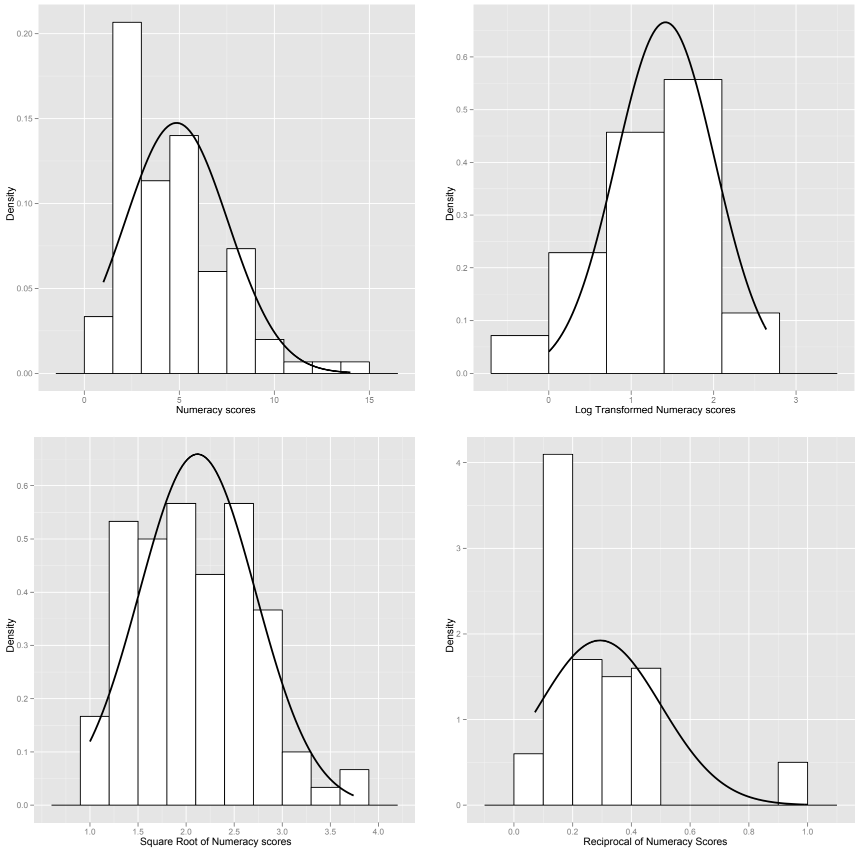
```
rexam$recnumeracy <- 1/(rexam$numeracy)
```

To create a histogram for *recnumeracy* we would execute:

```
hist.recnumeracy <- ggplot(rexam, aes(recnumeracy)) +
             opts(legend.position = "none")+
              geom_histogram(aes(y=..density..),
               colour="black", fill="white", binwidth = .1) +
             labs(x="Reciprocal of Numeracy Scores", y = "Density") +
             stat_function(fun=dnorm,
               args=list (mean=mean (rexam$recnumeracy, na.rm=TRUE),
                sd=sd(rexam$recnumeracy, na.rm=TRUE) ),
                colour="black", size=1)
```

```
hist.recnumeracy
```

The resulting histograms should look like this:




None of these histograms appear to be normal.
Let's have a look at the Shapiro–Wilk test for the original and transformed numeracy scores to further check for normality. For *numeracy*:

```
shapiro.test(rexam$numeracy)
```

```
 Shapiro-Wilk normality test

 data:  rexam$numeracy
 W = 0.9244, p-value = 2.424e-05
```

For *lognumeracy*:

```
shapiro.test(rexam$lognumeracy)
```

```
 Shapiro-Wilk normality test

 data:  rexam$lognumeracy
 W = 0.9591, p-value = 0.003474
```

For *sqrtnumeracy*:

```
shapiro.test(rexam$sqrtnumeracy)
```

```
 Shapiro-Wilk normality test

 data:  rexam$sqrtnumeracy
 W = 0.9695, p-value = 0.02035
```

For *recnumeracy*:

```
shapiro.test(rexam$recnumeracy)
```

```
Shapiro-Wilk normality test
```

```
data:  rexam$recnumeracy
W = 0.7633, p-value = 2.135e-11
```

The results from all of the Shapiro–Wilk tests above are significant. The only conclusion is that although the square root transformation does the best job of normalizing the data, none of these transformations actually works!