Practice: Graphics

Jay Wei

2023-09-24

Contents

1 Exercise 1. **2**

2 Exercise 2. **4**

2.1 Question a. 4

2.2 Question b. 10

2.3 Question c. 10

3 Exercise 3. **11**

1 Exercise 1.

Reproduce the barley experiment plot in the Trellis plotting section of the notes. The data is in the `barley` data frame of the `lattice` package.

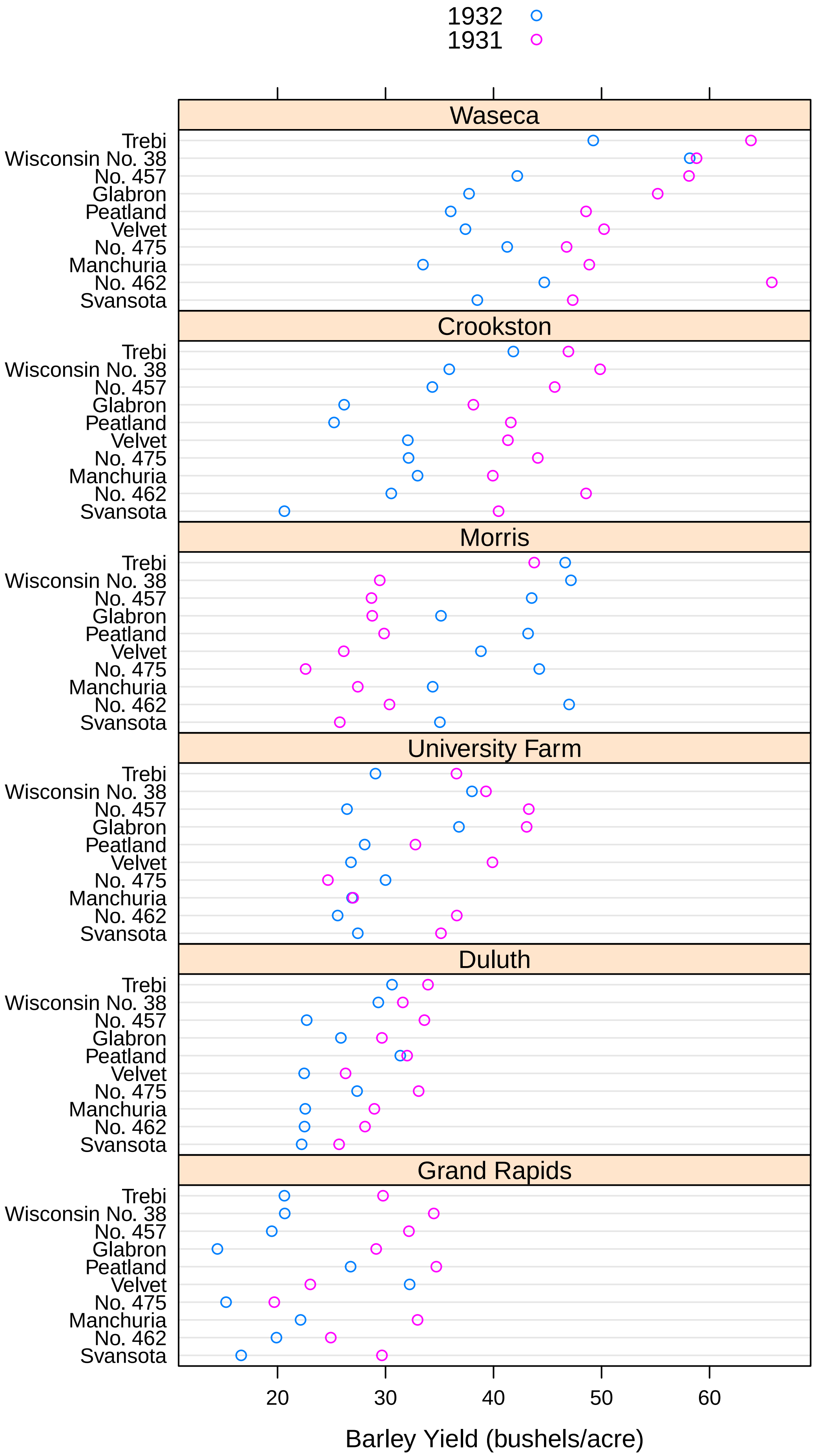
```r
library(lattice)

head(barley)
```

```
##      yield   variety year            site
## 1 27.00000 Manchuria 1931 University Farm
## 2 48.86667 Manchuria 1931          Waseca
## 3 27.43334 Manchuria 1931          Morris
## 4 39.93333 Manchuria 1931       Crookston
## 5 32.96667 Manchuria 1931    Grand Rapids
## 6 28.96667 Manchuria 1931          Duluth
```

`auto.key = TRUE`: used to automatically produce a suitable legend in conjunction with a grouping variable (which is `year` here).

`layout = c(1, 6)`: specifies the number of columns to be 1, and the number of rows to be 6. Indeed, this can be weird as in most cases, we have row first and column next.

```r
dotplot(x = variety ~ yield | site, data = barley, groups = year,
        auto.key = TRUE,
        xlab = "Barley Yield (bushels/acre)",
        layout = c(1,6))
```


1932
1931
Waseca
Crookston
Morris
University Farm
Duluth
Grand Rapids
Trebi
Wisconsin No. 38
No. 457
Glabron
Peatland
Velvet
No. 475
Manchuria
No. 462
Svansota
20
30
40
50
60
Barley Yield (bushels/acre)

2 Exercise 2.

STAT 950 course uses graphics to better understand the results of a Monte Carlo simulation. In one class project, students evaluated six methods to calculate a confidence interval for a variance.

The confidence level was set to 95% ($\alpha = 0.05$) for these intervals throughout the project.

The simulation results are given in the file `SimResults.csv`

```
set1 <- read.csv("SimResults.csv")
head(set1)
```

```
##              CI  Coverage ExpLength NA. Distribution SampleSize
## 1 Normal-based 0.7940000     16.33   0        Gamma          9
## 2   Asymptotic 0.6036217      7.98   3        Gamma          9
## 3        Basic 0.6440000      8.28   0        Gamma          9
## 4   Percentile 0.6300000      8.28   0        Gamma          9
## 5          BCa 0.6740000      9.47   0        Gamma          9
## 6  Studentized 0.9000000    128.81   0        Gamma          9
```

The columns represent:

- `CI` = Confidence interval methods
- `Coverage` = The proportion of times that the confidence interval contained σ^2
- `ExpLength` = Average length of the confidence interval across all simulated data sets
- `NA` = Number of times out of 500 that a confidence interval could not be calculated
- `Distribution` = The distribution from which the data was simulated
- `SampleSize` = The sample size used for each of the 500 simulated data sets.

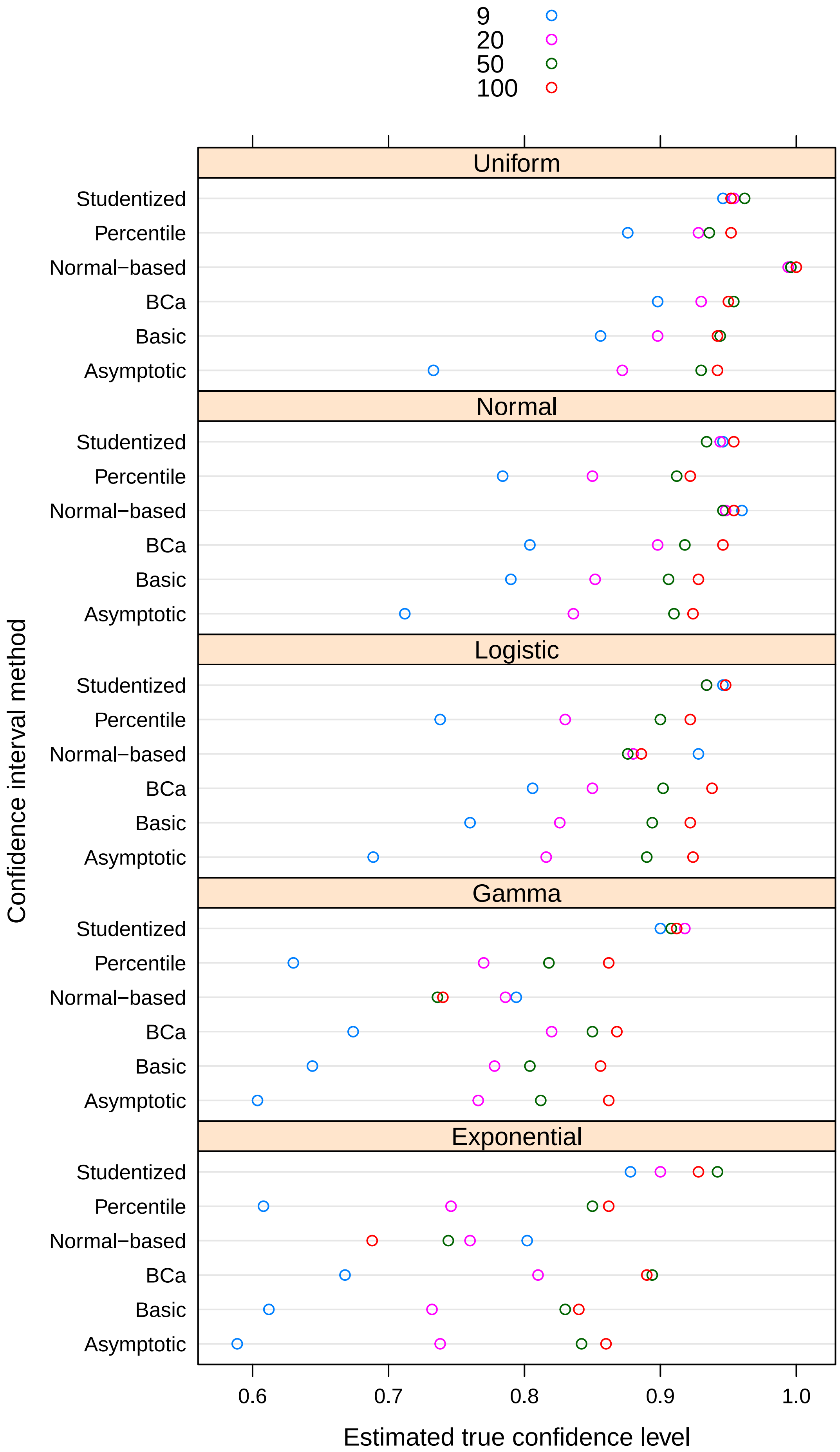
Students simulated n data sets from a specified probability distribution (gamma, logistic, uniform, exponential, or normal) with a known value of σ^2 which was the same for each distribution. The confidence interval methods were applied to each of the data sets and the proportion of times that the interval contained σ^2 was recorded.

2.1 Question a.

Examine the results using the graphical methods discussed in the notes. Discuss which plots are the best to use in this situation.

2.1.1 Very simple plot for confidence level

```
dotplot(CI ~ Coverage | Distribution, data = set1,
        groups = SampleSize,
        auto.key = TRUE,
        xlab = "Estimated true confidence level",
        layout = c(1,5),
        ylab = "Confidence interval method")
```



9
20
50
100
Uniform
Normal
Logistic
Gamma
Exponential
Studentized
Percentile
Normal−based
BCa
Basic
Asymptotic
Confidence interval method
0.6
0.7
0.8
0.9
1.0
Estimated true confidence level

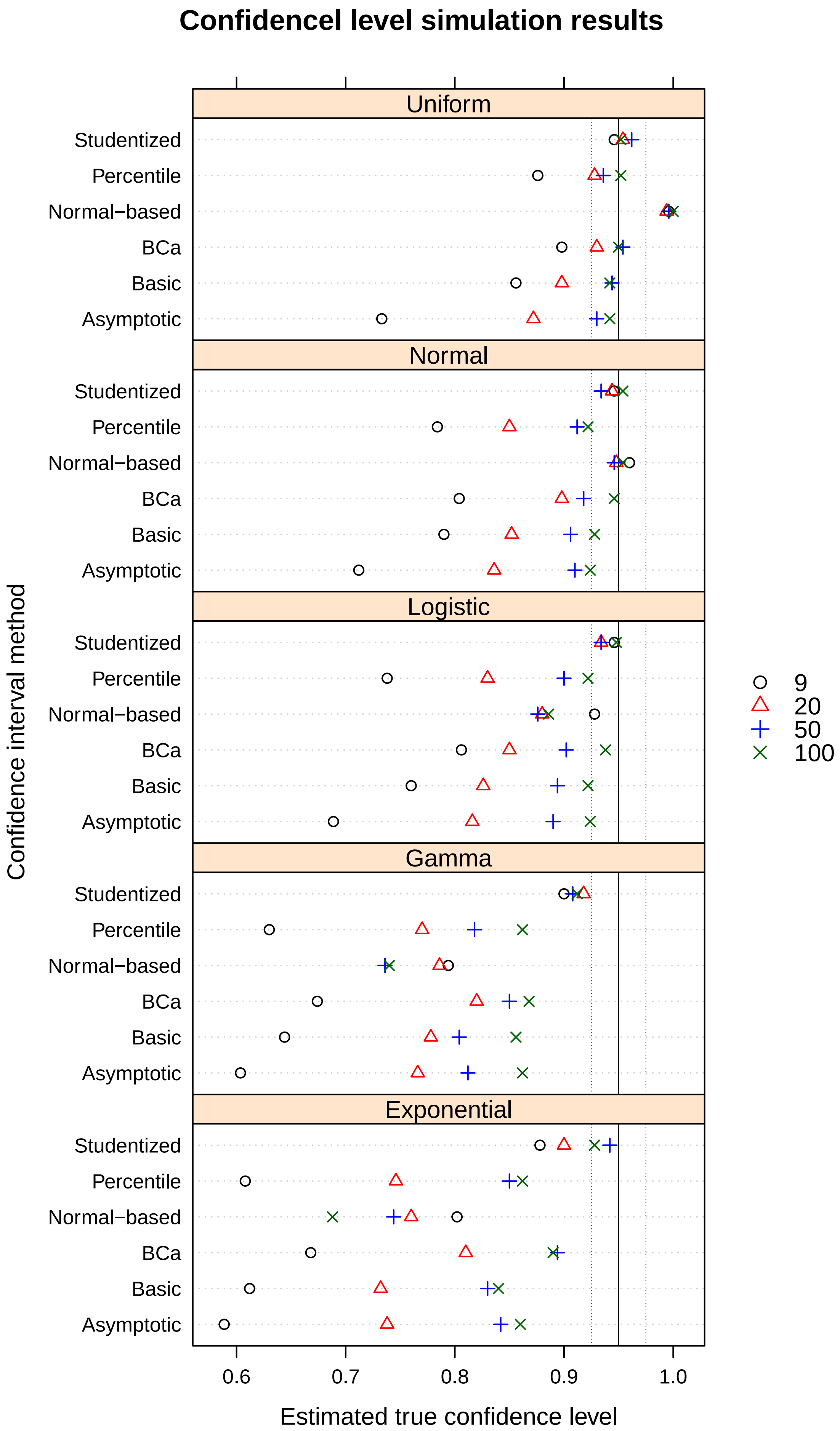
2.1.2 A nicer plot for confidence level

```
plot.levels <- levels(factor(set1$SampleSize))
plot.levels
```

```
## [1] "9"   "20"  "50"  "100"
```

This is one way to obtain all of the sample sizes and put into a vector where the elements are characters. A more simple (but less general) way is to just manually enter the sample size levels as `plot.levels <- c("9", "20", "50", "100")`

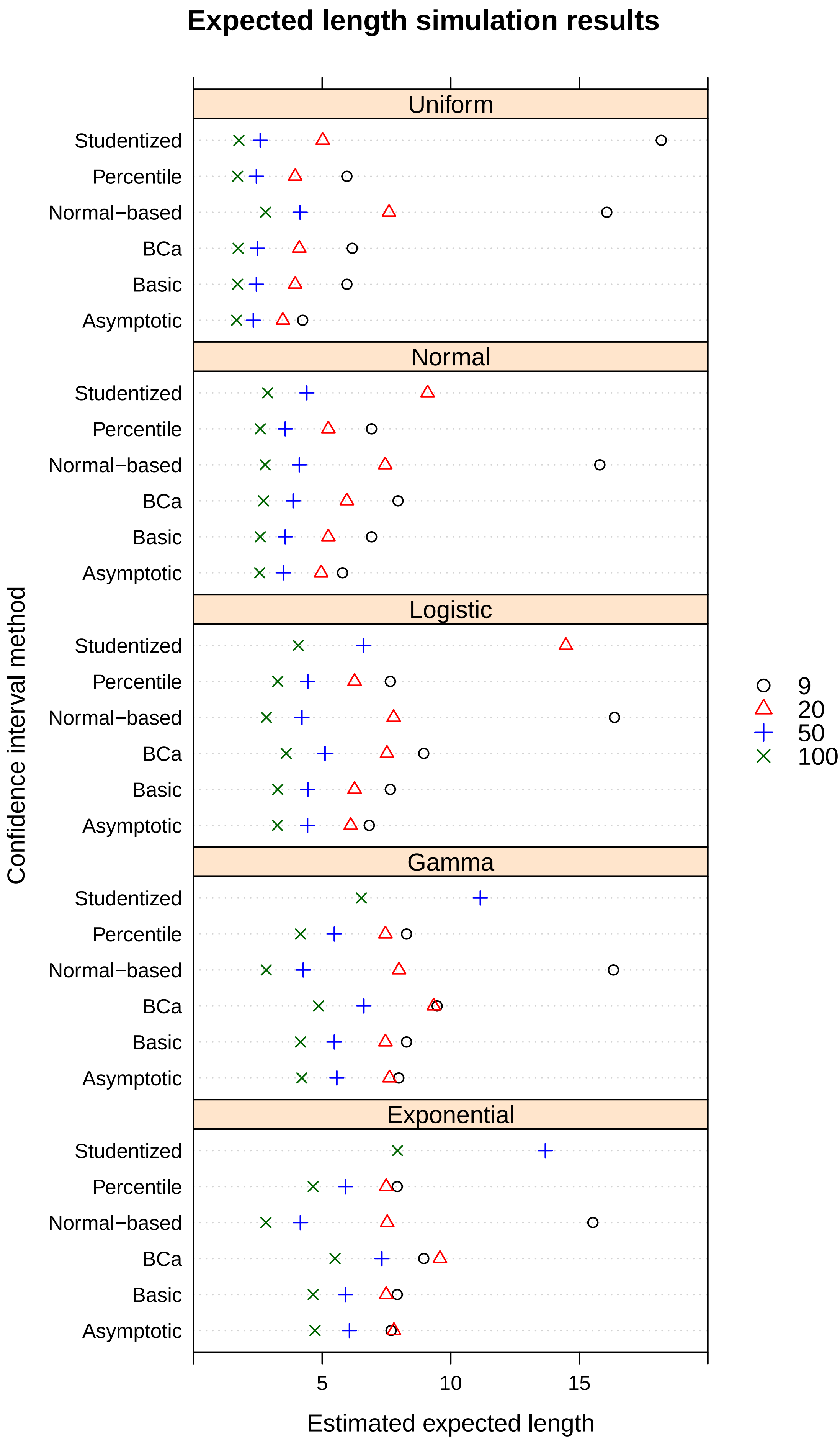
```
dotplot(CI ~ Coverage | Distribution, data = set1,
        groups = SampleSize,
        main = "Confidencel level simulation results",
        key = list(space = "right",
                   points = list(pch = 1:4,
                                 col = c("black", "red",
                                         "blue", "darkgreen")),
                   text = list(lab = plot.levels)),
        panel = function(x, y) {
           panel.grid(h = -1, v = 0,
                      lty = "dotted", lwd = 1, col="lightgray")
           panel.abline(v = 0.95, lty = "solid", lwd = 0.5)
           panel.abline(v = c(0.925, 0.975),
                        lty = "dotted", lwd = 0.5)
           panel.xyplot(x = x, y = y, col = c(rep("black", times = 6),
                                              rep("red", times = 6),
                                              rep("blue", times = 6),
                                              rep("darkgreen", times = 6)),
                        pch = c(rep(1,6), rep(2,6),
                                rep(3, 6), rep(4, 6)))
         },
         xlab = "Estimated true confidence level",
         layout = c(1,5),
         ylab = "Confidence interval method")
```


Confidencel level simulation results
Uniform
Normal
Logistic
Gamma
Exponential
Studentized
Percentile
Normal−based
BCa
Basic
Asymptotic
Confidence interval method
Estimated true confidence level
0.6
0.7
0.8
0.9
1.0
9
20
50
100

2.1.3 Expected length

I restricted the x-axis here due to some VERY large lengths (thus, some lengths may not be shown on the plot).

```
dotplot(CI ~ ExpLength | Distribution, data = set1,
        groups = SampleSize,
        main = "Expected length simulation results",
        key = list(space = "right",
                   points = list(pch = 1:4, col = c("black", "red",
                                                 "blue","darkgreen")),
                   text = list(lab = plot.levels)),
        xlim = c(0, 20), # restricted the x-axis
        panel = function(x, y) {
          panel.grid(h = -1, v = 0,
                     lty = "dotted", lwd = 1, col="lightgray")
          panel.xyplot(x = x, y = y, col = c(rep("black", times = 6),
                                             rep("red", times = 6),
                                             rep("blue", times = 6),
                                             rep("darkgreen", times = 6)),
                       pch = c(rep(1,6), rep(2,6),
                               rep(3, 6), rep(4, 6)))
          },
        xlab = "Estimated expected length",
        layout = c(1,5),
        ylab = "Confidence interval method")
```



Many other types of plots can be examined here.

2.2 Question b.

Develop an overall conclusion about which method(s) are best.

Overall, the studentized bootstrap interval appears to be the best in terms of the true confidence level, but it can be exceptionally long in length. (You can see it by excluding `xlim = c(0, 20)` from the code)

2.3 Question c.

Do you think the normal-based method typically taught in STAT 801 is good to use in practice? Explain your answer.

The normal-based interval is what one typically learns about in STAT 801. Specifically, if s^2 denotes the sample variance, σ^2 denotes the population variance, and n denotes the sample size, the interval is

$$\frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}} < \sigma^2 < \frac{(n-1)s^2}{\chi^2_{\alpha/2,n-1}}$$

For this problem, many of the confidence intervals methods often do not work well.

3 Exercise 3.

Use the graphical methods discussed in the notes to examine the `goblet` data. This data set is in the file `goblet.csv` on my course website.

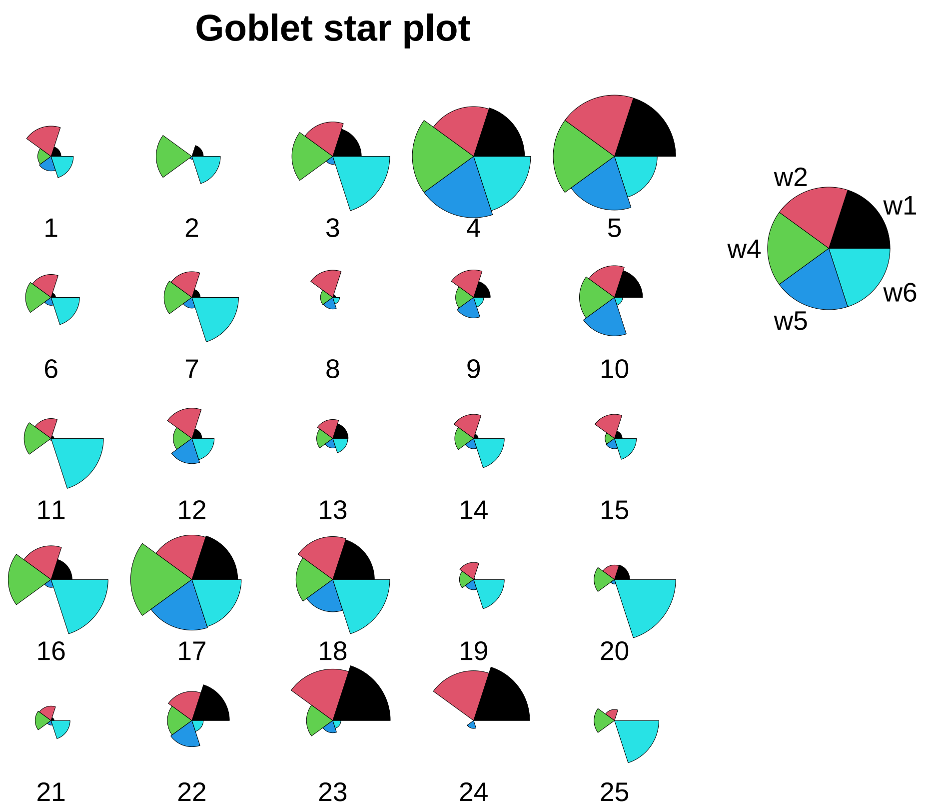
```
goblet <- read.csv("../../Chapter1-Background/3-dist/goblet.csv")
head(goblet)
```

```
##   goblet x1 x2 x3 x4 x5 x6
## 1      1 13 21 23 14  7  8
## 2      2 14 14 24 19  5  9
## 3      3 19 23 24 20  6 12
## 4      4 17 18 16 16 11  8
## 5      5 19 20 16 16 10  7
## 6      6 12 20 24 17  6  9
```

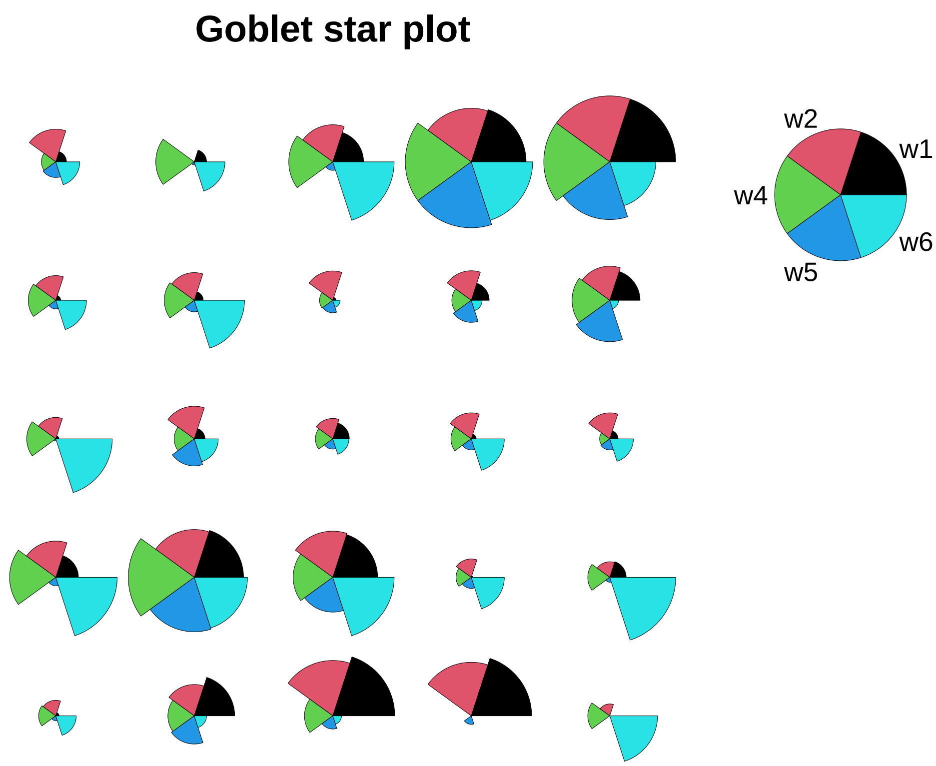
```
goblet2 <- data.frame(ID = goblet$goblet,
                      w1 = goblet$x1/goblet$x3,
                      w2 = goblet$x2/goblet$x3,
                      w4 = goblet$x4/goblet$x3,
                      w5 = goblet$x5/goblet$x3,
                      w6 = goblet$x6/goblet$x3)
head(goblet2)
```

```
##   ID        w1        w2        w4        w5        w6
## 1  1 0.5652174 0.9130435 0.6086957 0.3043478 0.3478261
## 2  2 0.5833333 0.5833333 0.7916667 0.2083333 0.3750000
## 3  3 0.7916667 0.9583333 0.8333333 0.2500000 0.5000000
## 4  4 1.0625000 1.1250000 1.0000000 0.6875000 0.5000000
## 5  5 1.1875000 1.2500000 1.0000000 0.6250000 0.4375000
## 6  6 0.5000000 0.8333333 0.7083333 0.2500000 0.3750000
```

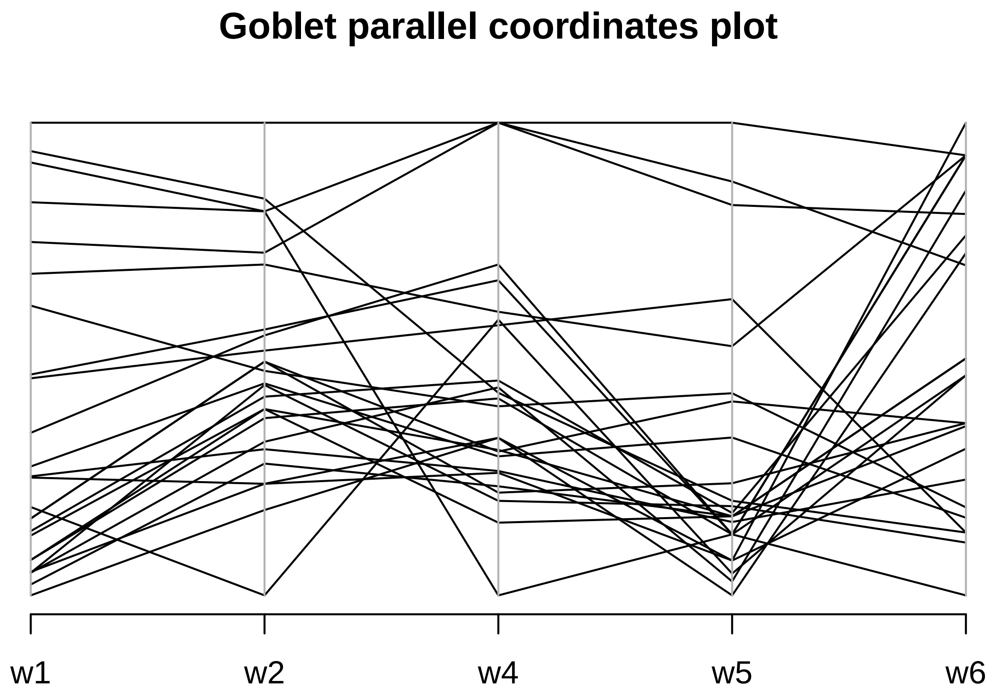
```
#Stars plot; The "-1" index is a quick way to remove the first column here
#  I used the labels argument because the goblet numbers were not included
stars(x = goblet2[,-1], draw.segments = TRUE, key.loc = c(15,10),
      main = "Goblet star plot", labels = goblet2$ID)
```



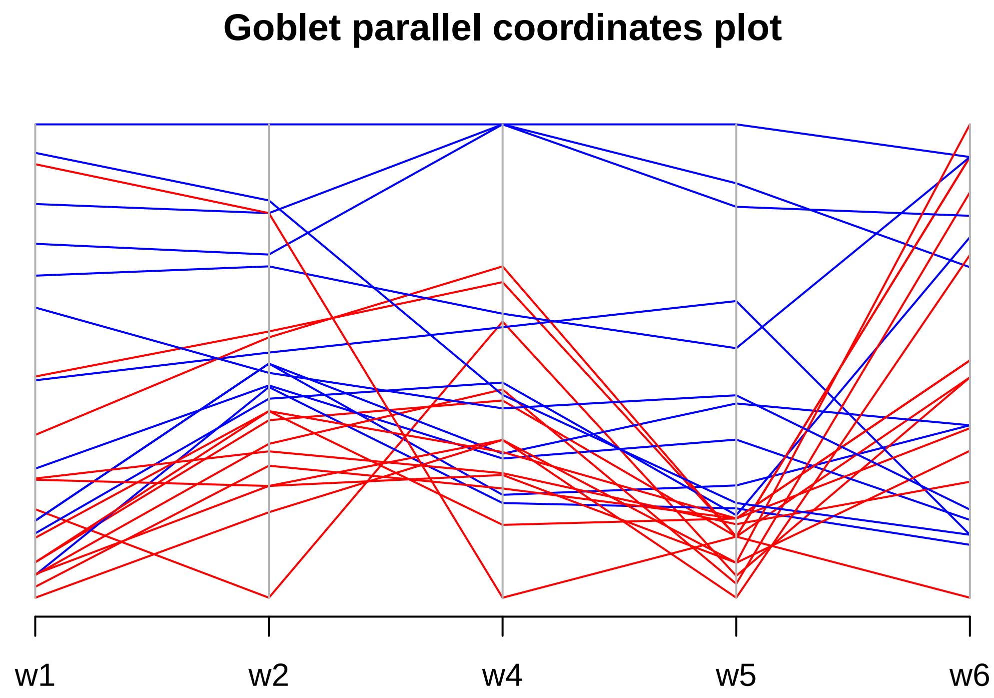

```
#Interestingly, the goblet numbers are not included when I use 2:6 as the column number
stars(x = goblet2[,2:6], draw.segments = TRUE, key.loc = c(14,10),
      main = "Goblet star plot")
```



```r
library(MASS)
parcoord(x = goblet2[,-1], main = "Goblet parallel coordinates plot")
```




```r
# A crude way to do brushing
col.w5 <- ifelse(test = goblet2$w5 <= median(goblet2$w5),
                 yes = "red", no = "blue")
parcoord(x = goblet2[,-1], col = col.w5,
         main = "Goblet parallel coordinates plot")
```



These plots should be interpreted. One interesting finding here is that it appears a large connection between the base and the cup leads to larger values of w1, w2, w4, and w6. This “trend” that we see in the data could lead to possible classifications that we could put the goblets in. We will discuss this more in later sections.

Many other types of plots can be examined here.