QBS 120 - Problem Set 7

Rob Frost

Grading: problem 1 for all 10 points; see details below for guidelines

1. **Rice, Chapter 10, Problem 26: Hampson and Walker also made measurements of the heats of sublimation of rhodium and iridium. Do the following calculations for each of the two given sets of data:**

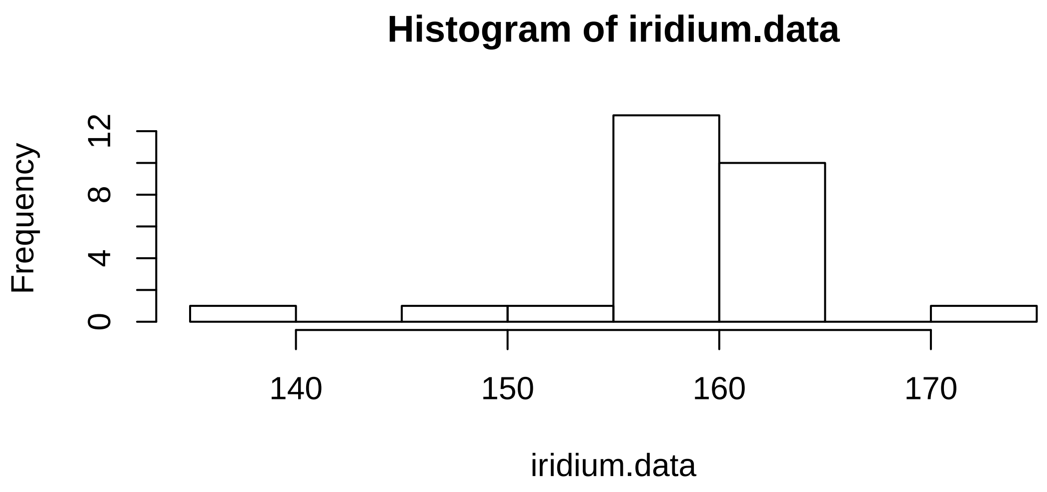
```
> iridium.data = c(136.6, 145.2, 151.5, 162.7, 159.1, 159.8, 160.8, 173.9, 160.1,
+                          160.4, 161.1, 160.6, 160.2, 159.5, 160.3, 159.2, 159.3, 159.6,
+                          160.0, 160.2, 160.1, 160.0, 159.7, 159.5, 159.5, 159.6, 159.5)
> rhodium.data = c(126.4, 135.7, 132.9, 131.5, 131.1, 131.1, 131.9, 132.7,
+                            133.3, 132.5, 133.0, 133.0, 132.4, 131.6, 132.6, 132.2,
+                            131.3, 131.2, 132.1, 131.1, 131.4, 131.2, 131.1, 131.1,
+                            134.2, 133.8, 133.3, 133.5, 133.4, 133.5, 133.0, 132.8,
+                            132.6, 133.3, 133.5, 133.5, 132.3, 132.7, 132.9, 134.1)
```

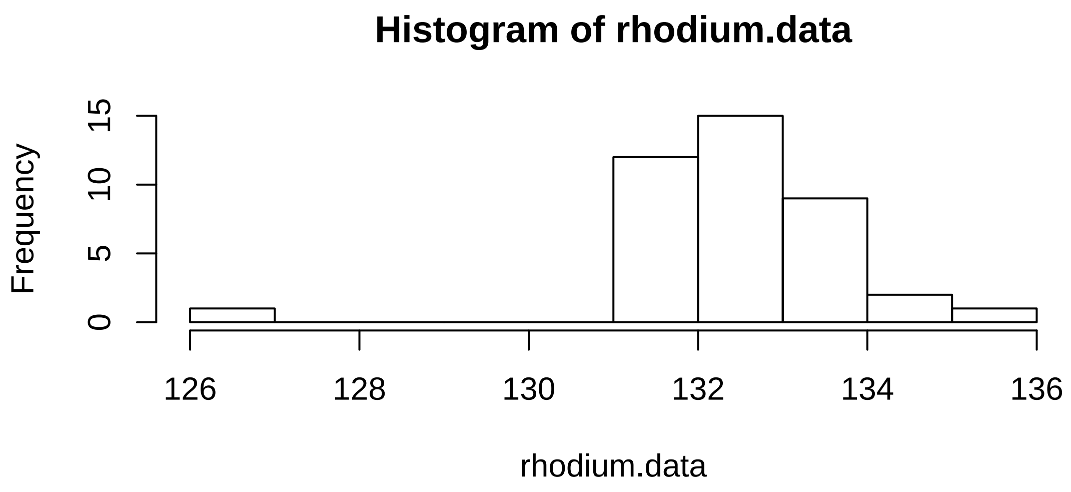
   (a) **Make a histogram.**

   Grading: 0.5 pts to make the plots

   We'll let hist() automatically select the bin sizes:

```
> par(mfrow=c(2,1))
> hist(iridium.data)
> hist(rhodium.data)
```

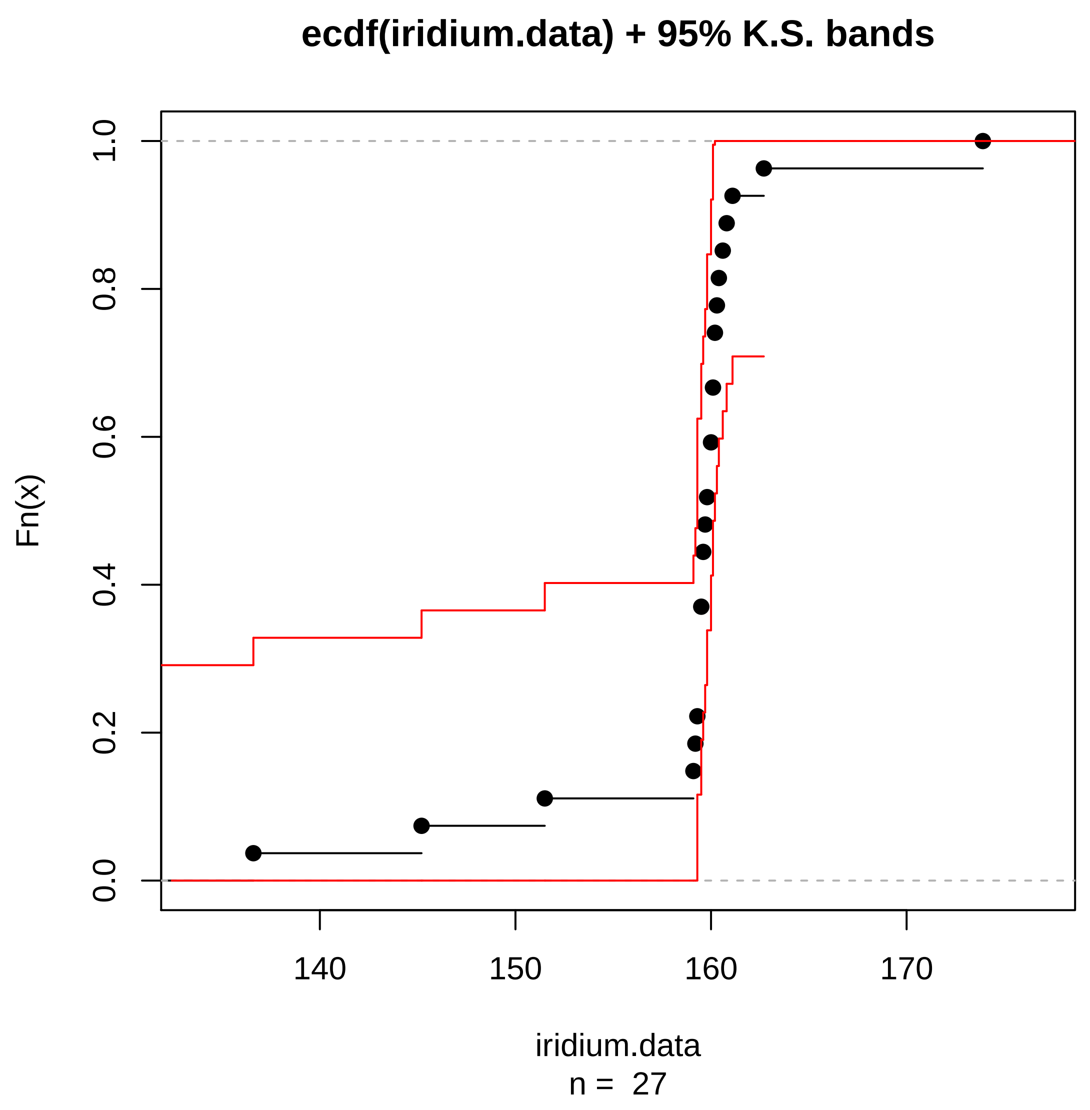




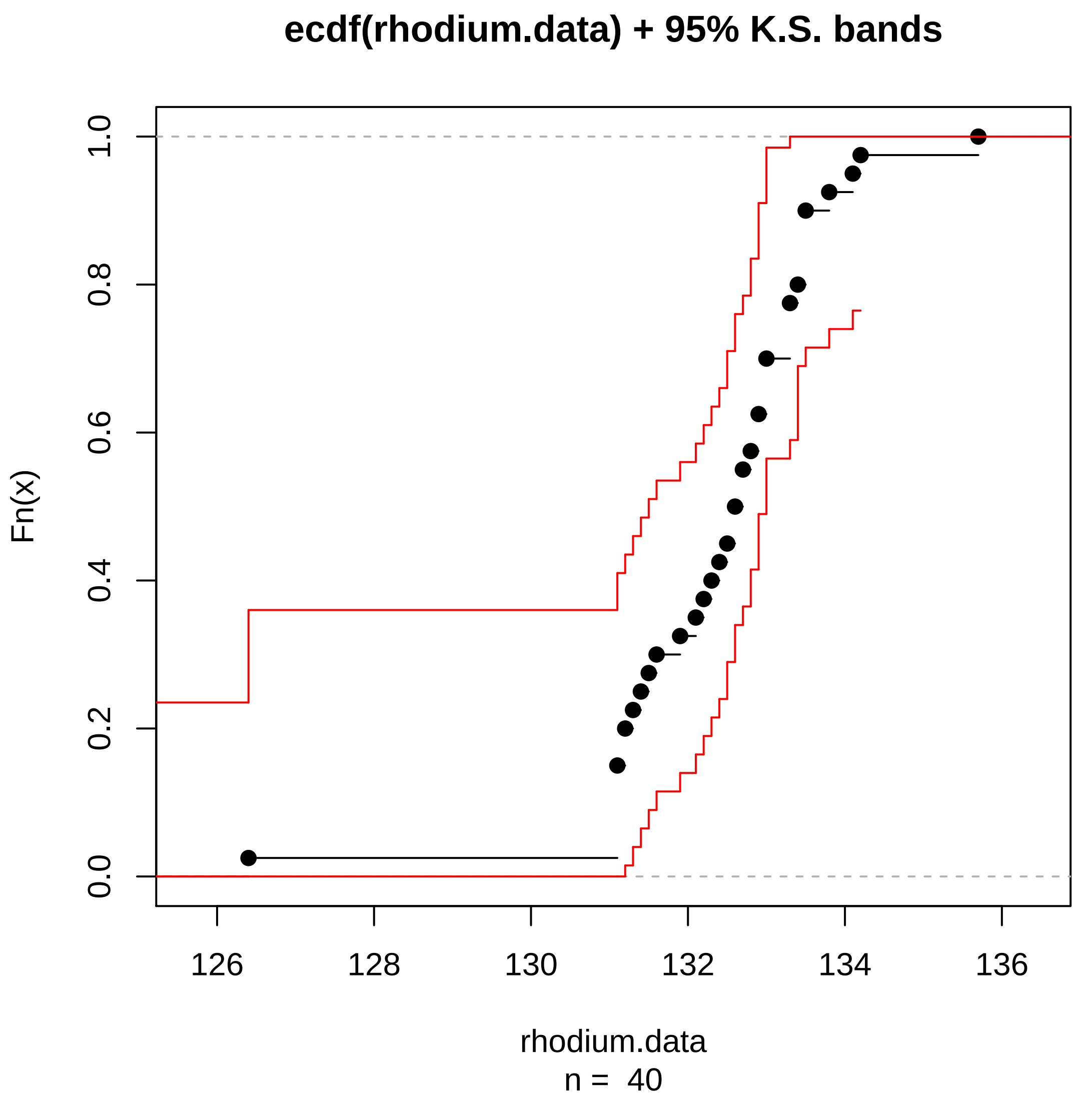
Looks like there are some outliers in both cases (high and low for iridium and low for rhodium), however, the default histogram bins are quite wide so tough to tell.

(b) **Plot the eCDFs with 95% confidence bands.** ***Grading: 0.5 pts to make the plots***

```
> library(sfsmisc)
> ecdf.ksCI(iridium.data)
```

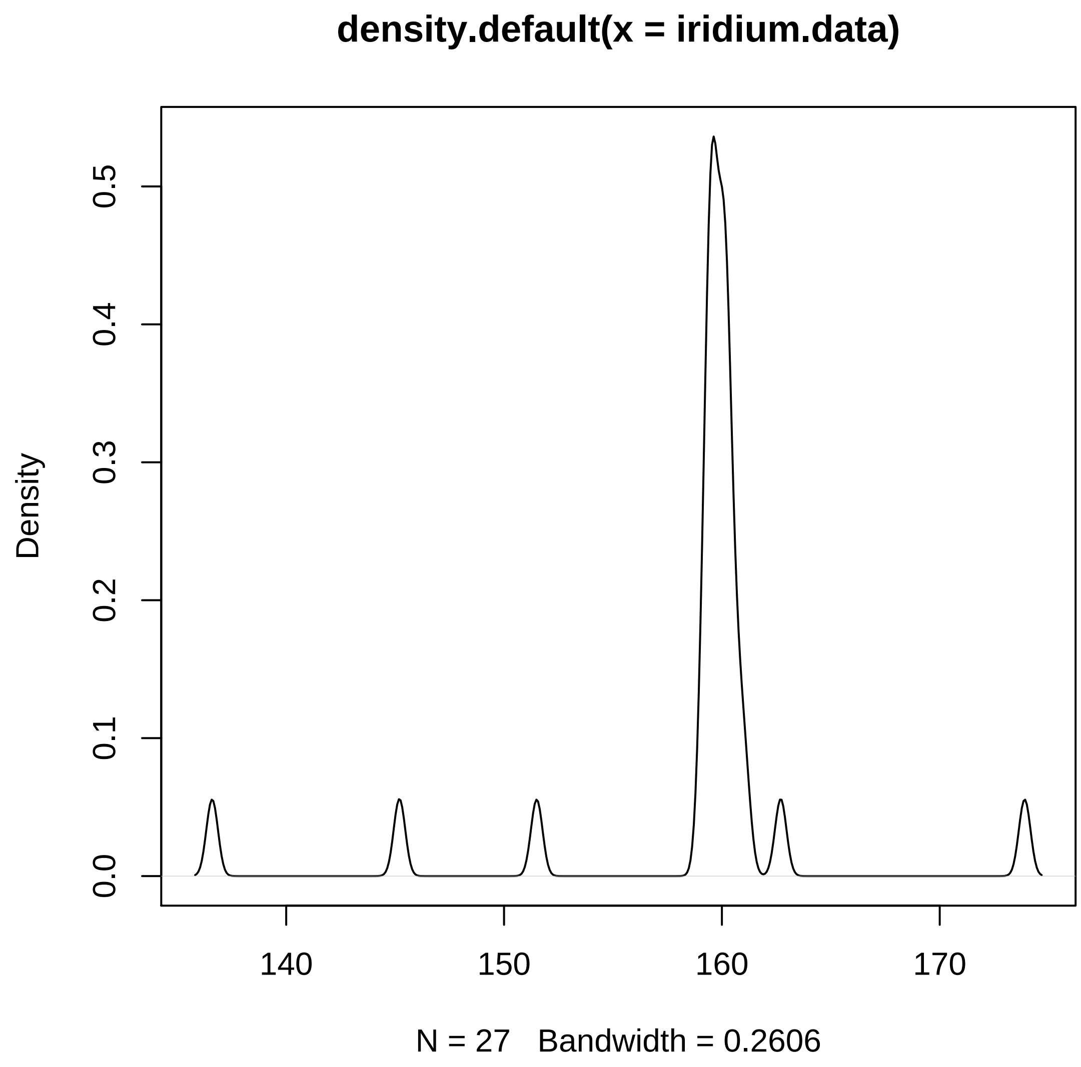


```
> ecdf.ksCI(rhodium.data)
```

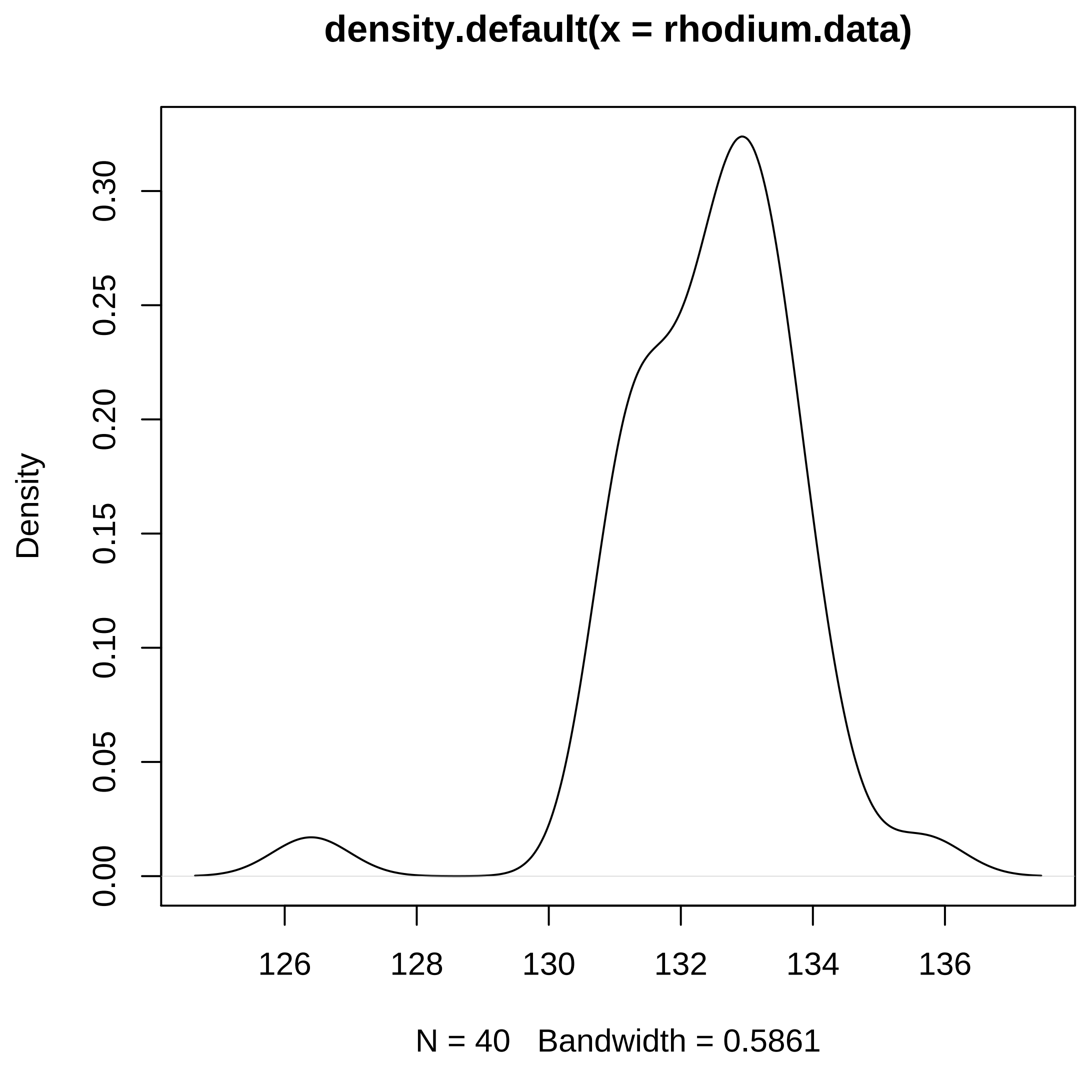


(c) **Plot the kernel density estimates.** ***Grading: 0.5 pts to make the plots***

```
> plot(density(iridium.data))
```



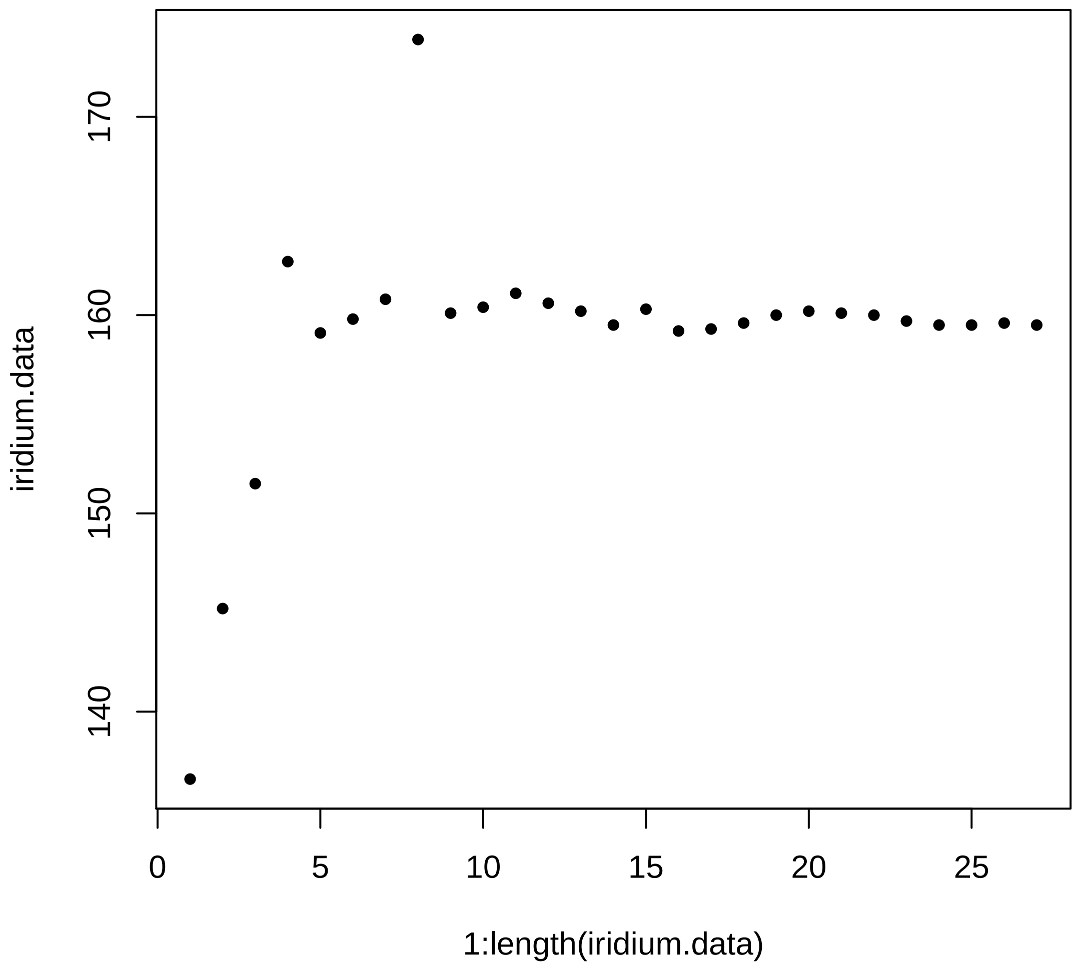
```
> plot(density(rhodium.data))
```



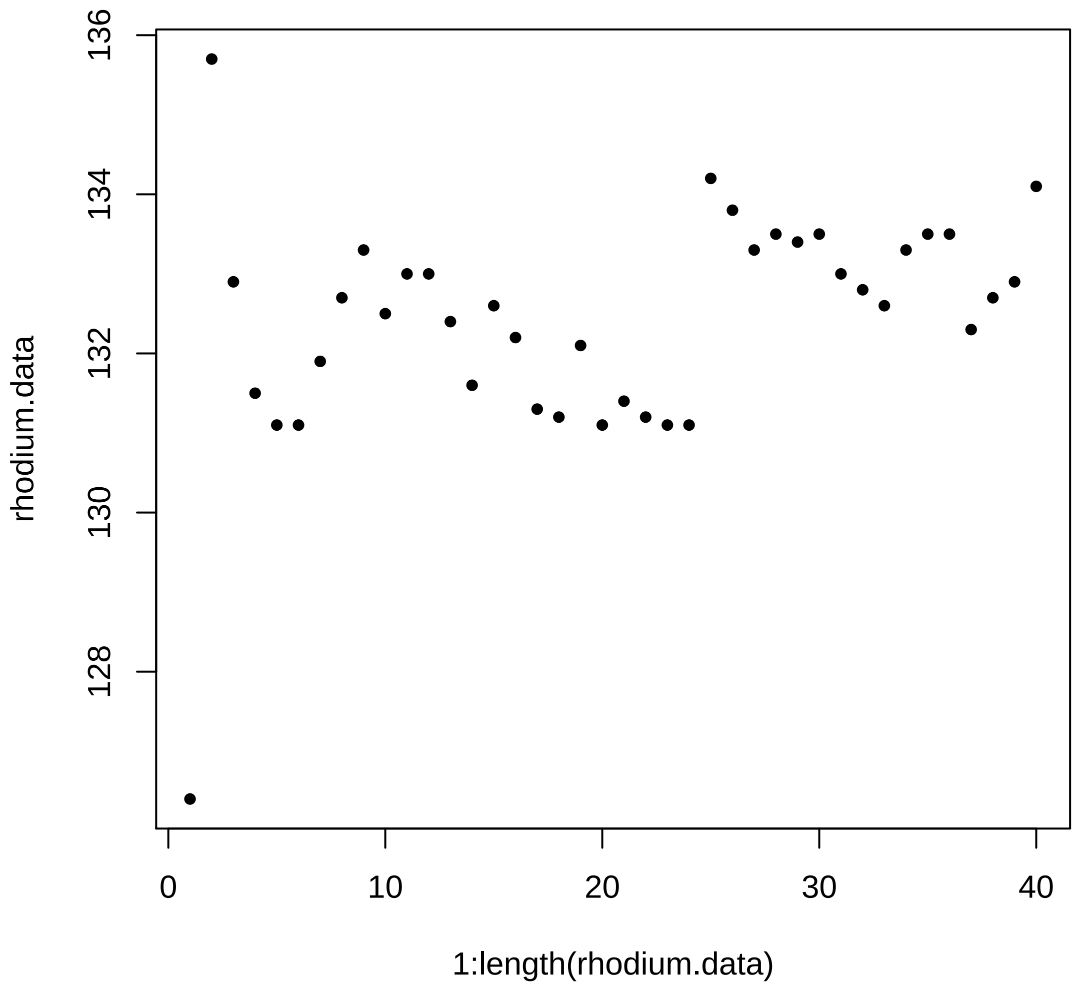
The outliers are very clear for the iridium case, however, the smoothing is not so helpful given the small number of observations. Works a bit better for rhodium.

The eCDFs make the outliers more clear (low and high for iridium and a single low for rhodium)

(d) **Plot the observations in the order of the experiment.** ***Grading: 0.5 pts to make the plots***



iridium.data
170
160
150
140
0
5
10
15
20
25
1:length(iridium.data)



Given the small number of observations, I find these plots to be the most useful for visualizing the empirical distribution.

(e) **Does that statistical model of iid measurement errors seem reasonable? Explain.**

Grading: 1 pt; full credit if they get the right answer, "no", and provide some reasonable explanation; half credit if they get the wrong answer or don't provide a justification.

No, it does not seem reasonable. When the results are plotted according to the order of the experiments, there appears to be some sequential correlation. For iridium, looks like the researchers had calibration (or other) technical issues for the first few experiments but then achieved stable results. For rhodium, there was also some initial instability as well as an upward bias that appears after the 25th experiment.

(f) **Find the mean, 10% and 20% trimmed means, and median and compare them.** ***Grading: 1 pt; full credit if everything is generated correctly; half credit if there are errors***

```
> results = matrix(nrow=4, ncol=2,
+         dimnames=list(measure=c("mean", "10% trimmed mean", "20% trimmed mean", "med
+                 data=c("Iridium", "Rhodium")))
> results[,1] = c(mean(iridium.data),
+                 mean(iridium.data, trim=0.1),
+                 mean(iridium.data, trim=0.2),
+                 median(iridium.data))
> results[,2] = c(mean(rhodium.data),
+                 mean(rhodium.data, trim=0.1),
+                 mean(rhodium.data, trim=0.2),
+                 median(rhodium.data))
> results
```

```
                  data
measure             Iridium  Rhodium
  mean             158.8148 132.4200
  10% trimmed mean 159.5478 132.4781
  20% trimmed mean 159.8412 132.5292
  median           159.8000 132.6500
```

Despite the appearance of outliers, all of these measures appear quite similar. The iridium mean is a bit lower than the robust measures. The is likely due to the fact that the number of outliers is small and there are balancing small and large outliers.

(g) **Find the standard error of the sample mean and a corresponding 90% confidence interval. Overlay this CI on a density plot.** ***Grading: 2 pts; get full credit if they generate and plot the 90% CIs; 1 pt if they generate something but items are missing or look invalid***

We know that the sample mean is asymptotically normal with mean μ and variance σ^2/n:

$$P(\Phi^{-1}(0.05) \leq \frac{\bar{x}-\mu}{\sigma/\sqrt{n}} \leq \Phi^{-1}(0.95)) = 0.9$$

$$P(-\Phi^{-1}(0.95) \leq \frac{\bar{x}-\mu}{\sigma/\sqrt{n}} \leq \Phi^{-1}(0.95)) = 0.9$$

$$P(-\frac{\sigma\Phi^{-1}(0.95)}{\sqrt{n}} - \bar{x} \leq -\mu \leq \frac{\sigma^2\Phi^{-1}(0.95)}{n} - \bar{x}) = 0.9$$

$$P(\bar{x} - \frac{\sigma\Phi^{-1}(0.95)}{\sqrt{n}} \leq \mu \leq \bar{x} + \frac{\sigma\Phi^{-1}(0.95)}{\sqrt{n}}) = 0.9$$

Plugging in the sample variance for σ^2 we get 90% CIs for μ based on the sample mean of:

$$(\bar{x} - \frac{s\Phi^{-1}(0.95)}{\sqrt{n}}, \bar{x} + \frac{s\Phi^{-1}(0.95)}{\sqrt{n}})$$

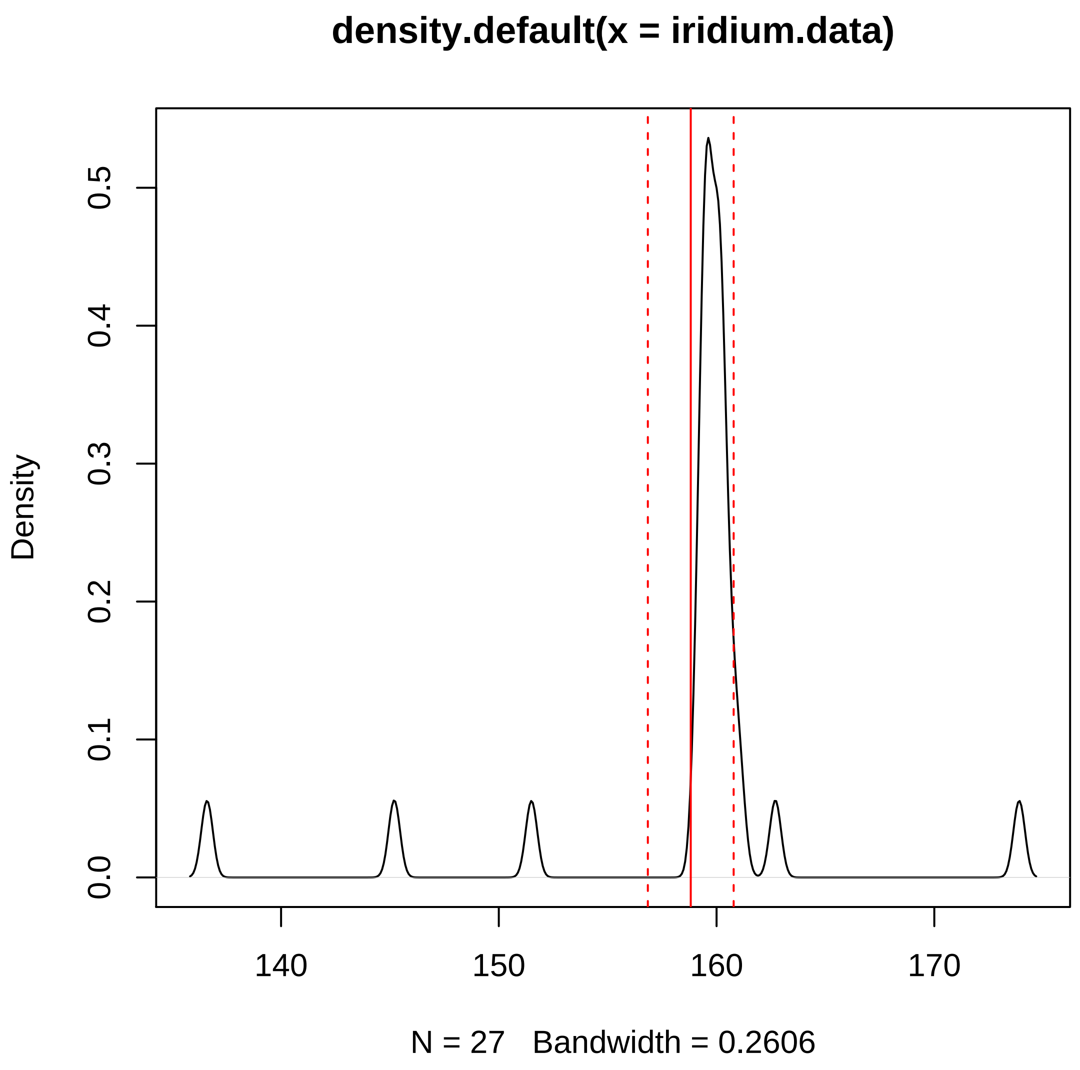
For the iridium data this is:

```
> n=length(iridium.data)
> (mean.ci.low.iridium =mean(iridium.data) - sd(iridium.data) * qnorm(0.95)/sqrt(n))
[1] 156.8444
```

```
> (mean.ci.high.iridium =mean(iridium.data) + sd(iridium.data) * qnorm(0.95)/sqrt(n))
[1] 160.7852
```

We can visualize this CI on the density plot:

```
> plot(density(iridium.data))
> abline(v=mean(iridium.data), col="red")
> abline(v=mean.ci.low.iridium, col="red", lty="dashed")
> abline(v=mean.ci.high.iridium, col="red", lty="dashed")
```

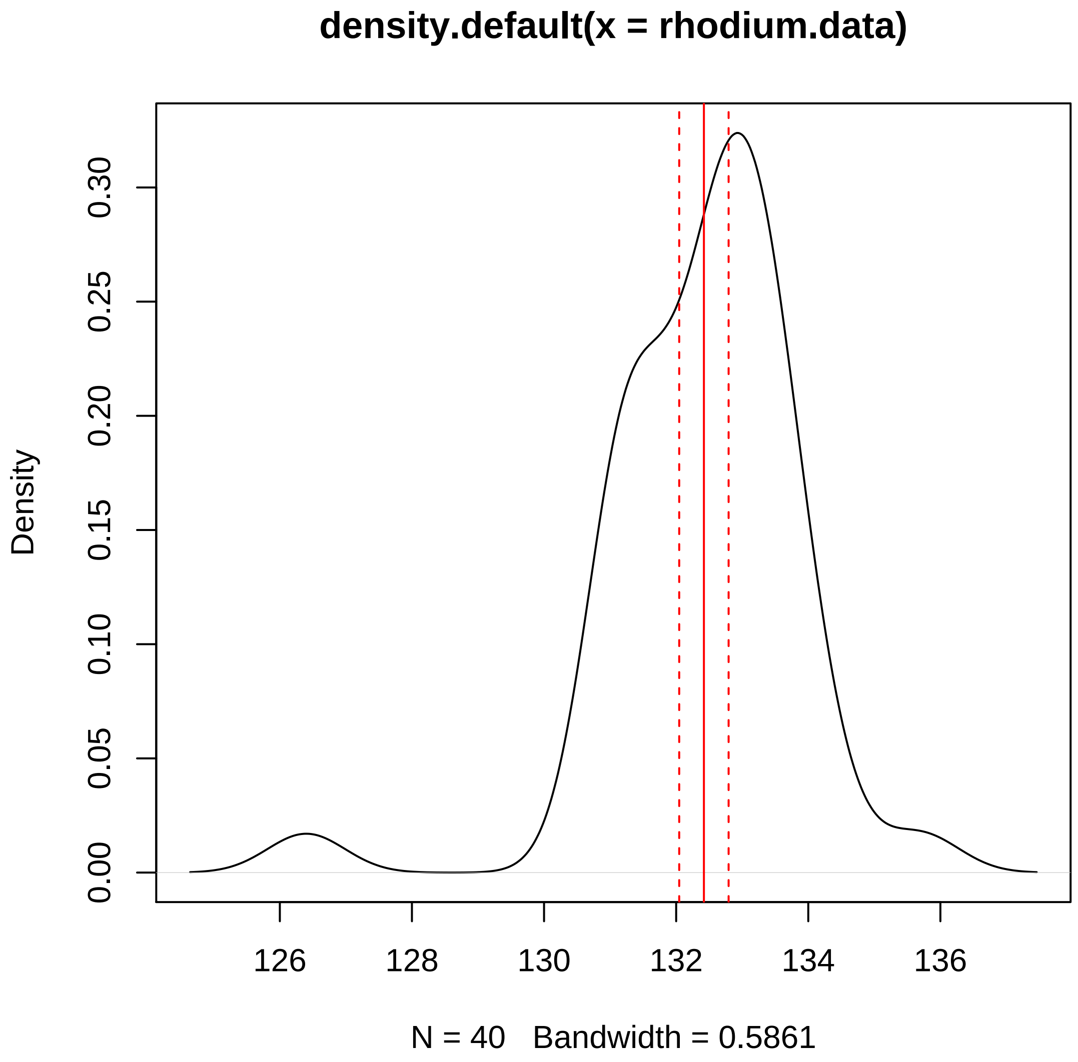


For the rhodium data this is:

```
> n=length(rhodium.data)
> (mean.ci.low.rhodium =mean(rhodium.data) - sd(rhodium.data) * qnorm(0.95)/sqrt(n))
[1] 132.0461
> (mean.ci.high.rhodium =mean(rhodium.data) + sd(rhodium.data) * qnorm(0.95)/sqrt(n))
[1] 132.7939
```

We can visualize this CI on the density plot:

```
> plot(density(rhodium.data))
> abline(v=mean(rhodium.data), col="red")
> abline(v=mean.ci.low.rhodium, col="red", lty="dashed")
> abline(v=mean.ci.high.rhodium, col="red", lty="dashed")
```



(h) **Use the bootstrap to approximate the sampling distribution of the 10% and 20% trimmed means and median. Plot the kernel density estimates of these bootstrap distributions in a single plot. Compute the standard errors and compare.** ***Grading: 2 pts; get full credit if they display bootstrap distributions and SEs that look valid; 1 pt if they generate something but items are missing or look invalid***

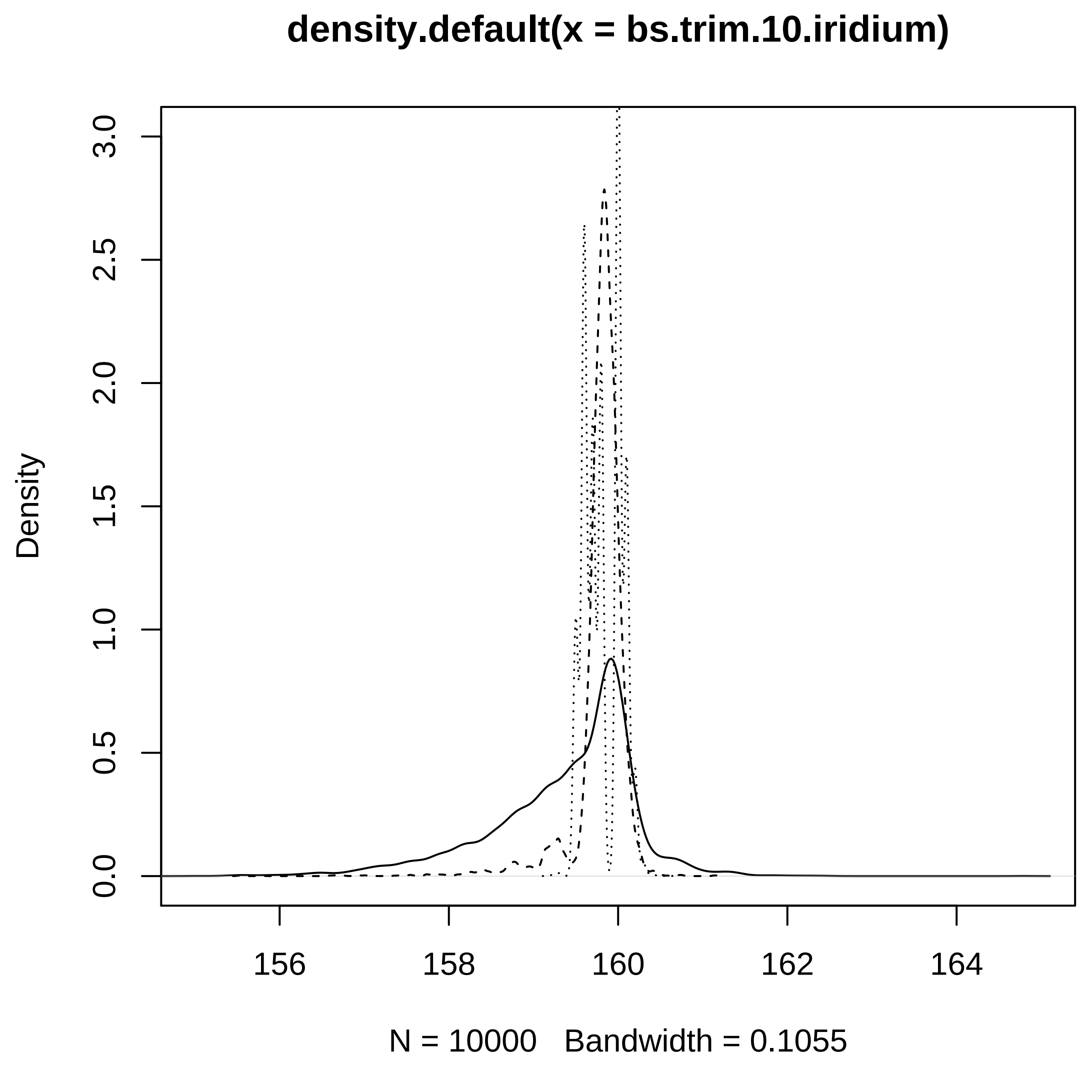
Compute for iridium:

```
> computeBootstrapDist = function(data, estimator, B=10000) {
+         n=length(data)
+         bootstrap.data = matrix(sample(data, B*n, replace=T), nrow=B, ncol=n)
+         bs.estimates = apply(bootstrap.data, 1, function(x) {
+                 estimator(x)
+         })
+         return (bs.estimates)
+ }
> bs.trim.10.iridium =computeBootstrapDist(iridium.data, function(x){mean(x, trim=0.1
> se.trim.10 = sd(bs.trim.10.iridium)
> bs.trim.20.iridium = computeBootstrapDist(iridium.data, function(x){mean(x, trim=0.
> se.trim.20 = sd(bs.trim.20.iridium)
> bs.median.iridium =computeBootstrapDist(iridium.data, function(x){median(x)})
> se.median = sd(bs.median.iridium)
> (ses = list(se.trim.10=se.trim.10,
+ se.trim.20=se.trim.20,
+ se.median=se.median))

$se.trim.10
[1] 0.8542939

$se.trim.20
[1] 0.3135588

$se.median
[1] 0.2107316
```

For the iridium data, the 20% trimmed mean is able to more consistently remove the outliers so has a smaller SE and narrower CI than the 10% trimmed mean. In this case, the median has a lower SE than either trimmed mean which implies that neither the 10% nor 20% trimmed means are effectively removing all outliers.

Plot the bootstrap estimate of the sampling distributions:

```
> plot(density(bs.trim.10.iridium), ylim=c(0,3), xlim=c(155,165))
> lines(density(bs.trim.20.iridium), lty="dashed")
> lines(density(bs.median.iridium), lty="dotted")
```



Compute for rhodium:

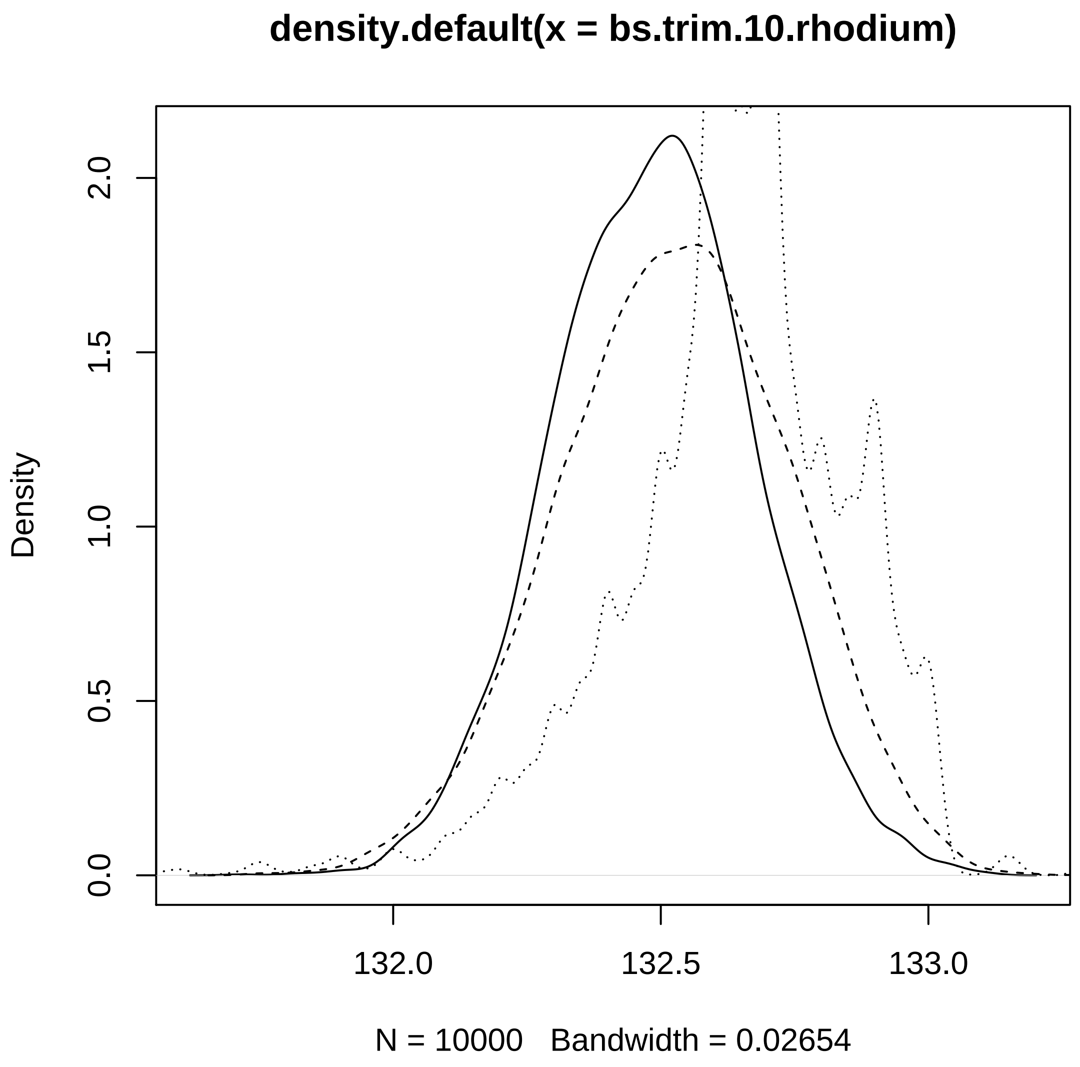
```
> bs.trim.10.rhodium =computeBootstrapDist(rhodium.data, function(x){mean(x, trim=0.1)
> se.trim.10 = sd(bs.trim.10.rhodium)
> bs.trim.20.rhodium =computeBootstrapDist(rhodium.data, function(x){mean(x, trim=0.2)
> se.trim.20 = sd(bs.trim.20.rhodium)
> bs.median.rhodium =computeBootstrapDist(rhodium.data, function(x){median(x)})
> se.median = sd(bs.median.rhodium)
> (ses = list(se.trim.10=se.trim.10,
+ se.trim.20=se.trim.20,
+ se.median=se.median))
```

```
$se.trim.10
[1] 0.1860299

$se.trim.20
[1] 0.2126874
```

```
$se.median
[1] 0.2160088
```

Plot the bootstrap estimate of the sampling distributions:

```
> plot(density(bs.trim.10.rhodium))
> lines(density(bs.trim.20.rhodium), lty="dashed")
> lines(density(bs.median.rhodium), lty="dotted")
```



For the rhodium data, the smaller number of outliers (and larger data set size) mean that both the 10% and 20% trimmed means are able to remove the outliers consistently so the 20% ends up with a slightly larger SE and CI since it is estimated on fewer observations than the 10% trimmed mean. As expected, the median has a higher SE than either of the trimmed means though the increase is small.

Overall, this example demonstrates that it is worthwhile to trade robustness for smaller sample size in the presence of outliers.

(i) **Find approximate 90% CIs based on the trimmed means and compare to the intervals for the mean and median.** ***Grading: 2 pts; get full credit if they generate and display CIs that look valid; 1 pt if they generate something but items are missing or look invalid***

```
> # create data structures to hold all of the values
> iridium.results = matrix(nrow=4, ncol=3,
+         dimnames=list(measure=c("mean", "10% trimmed mean", "20% trimmed mean", "mec
+                 data=c("estimate", "90% CI low", "90% CI high")))
> iridium.results[,1] = c(mean(iridium.data),
+                 mean(iridium.data, trim=0.1),
+                 mean(iridium.data, trim=0.2),
+                 median(iridium.data))
> iridium.results[1,2:3] = c(mean.ci.low.iridium, mean.ci.high.iridium)
> rhodium.results = matrix(nrow=4, ncol=3,
+         dimnames=list(measure=c("mean", "10% trimmed mean", "20% trimmed mean", "mec
+                 data=c("estimate", "90% CI low", "90% CI high")))
> rhodium.results[,1] = c(mean(rhodium.data),
+                 mean(rhodium.data, trim=0.1),
+                 mean(rhodium.data, trim=0.2),
+                 median(rhodium.data))
> rhodium.results[1,2:3] = c(mean.ci.low.rhodium, mean.ci.high.rhodium)
> computeBootstrapCI = function(bs.values, estimate, ci.percent=.9, B=10000) {
+         bs.values = sort(bs.values)
+         (quant.low = bs.values[B*(1-ci.percent)/2])
+         (quant.high = bs.values[B*(ci.percent + (1-ci.percent)/2)])
+         bs.ci.low = 2*estimate - quant.high
+         bs.ci.high = 2*estimate - quant.low
+         return (c(bs.ci.low,bs.ci.high))
+ }
> iridium.results[2,2:3] = computeBootstrapCI(bs.trim.10.iridium, mean(iridium.data, t
> iridium.results[3,2:3] = computeBootstrapCI(bs.trim.20.iridium, mean(iridium.data, t
> iridium.results[4,2:3] = computeBootstrapCI(bs.median.iridium, median(iridium.data)
> rhodium.results[2,2:3] = computeBootstrapCI(bs.trim.10.rhodium, mean(rhodium.data, t
> rhodium.results[3,2:3] = computeBootstrapCI(bs.trim.20.rhodium, mean(rhodium.data, t
> rhodium.results[4,2:3] = computeBootstrapCI(bs.median.rhodium, median(rhodium.data)
>
```

For the iridium data, the various location measures and 90% CIs are:

```
> iridium.results

                  data
measure            estimate 90% CI low 90% CI high
  mean             158.8148   156.8444    160.7852
  10% trimmed mean 159.5478   158.7435    161.3957
  20% trimmed mean 159.8412   159.5824    160.4353
  median           159.8000   159.5000    160.1000
```

In this case, the large number of outliers give all of the robust measures narrower CIs than the mean. As the % trim increases, the sampling distribution variance decreases (i.e., the outliers are move effectively handled with higher trim levels).

For the rhodium data, the various location measures and 90% CIs are:

```
> rhodium.results
                  data
measure            estimate 90% CI low 90% CI high
  mean             132.4200   132.0461    132.7939
  10% trimmed mean 132.4781   132.1688    132.7781
  20% trimmed mean 132.5292   132.1917    132.8917
  median           132.6500   132.3500    133.0500
```

In this case, the smaller number of outliers and larger sample size result in the robust measures being more similar to the mean (though still lower variance given the presence the small number of outliers). As the % trim increases, the sampling distribution variance increases slightly since all trim levels effectively handle the outliers so the sampling variance is driven by the number of observations used in the computation.

2. **Rice, Chapter 11, Problem 21: A study was done to compare the performances of engine bearings made of different compounds. Ten bearings of each type were tested. The following table gives the times until failure (in millions of cycles):**

```
> type.I.failure.times = c(3.03, 5.53, 5.6, 9.3, 9.92, 12.51, 12.95, 15.21, 16.04, 16.84)
> type.II.failure.times = c(3.19, 4.26, 4.47, 4.53, 4.67, 4.69, 12.78, 6.79, 9.37, 12.75)
```

   (a) **Use normal theory to test the hypothesis that there is no difference between the type types of bearings (you can use pt() but not t.test()).**

   Since we cannot use t.test(), need to compute our T statistic. We will assume equal variances and use the pooled variance estimate. Our test statistic therefore takes the form:

$$T = \frac{(\bar{x}_1 - \bar{x}_2)}{s_p\sqrt{1/n + 1/m}} \sim t_{n+m-2}$$

   Because we are not using the

   For the observed data, this is:

```
> (n = length(type.I.failure.times))
[1] 10
> (m = length(type.II.failure.times))
[1] 10
> var.p.hat = ((n-1) * var(type.I.failure.times) +
+            (m-1)*var(type.II.failure.times))/(n+m-2)
> (T = (mean(type.I.failure.times) - mean(type.II.failure.times))/
+          (sqrt(var.p.hat)*sqrt(1/n + 1/m)))
[1] 2.072309
```

   The p-value for a two-sided test is:

```
> (p.value = 2*(1-pt(T, df=(n+m-2))))
[1] 0.05287565
```

   Let's just confirm that this matches the output from t.test():

```
> t.test(type.I.failure.times, type.II.failure.times, var.equal=T)
        Two Sample t-test

data:  type.I.failure.times and type.II.failure.times
t = 2.0723, df = 18, p-value = 0.05288
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.05444249  7.94044249
sample estimates:
mean of x mean of y
   10.693     6.750
```

(b) **Test the same hypothesis using a nonparametric method (use just pnorm() to evaluate using the normal approximation for the rank sum and compare that result to the exact distribution using wilcox.test()).**

First compute the ranks and rank sums:

```
> computeRankSum = function(x.1, x.2) {
+         merged = c(x.1, x.2)
+         ranks = rank(merged, ties.method="average")
+         x.1.ranks = ranks[1:n]
+         R.sum.1 = sum(x.1.ranks)
+         return (R.sum.1)
+ }
> (R.type.I = computeRankSum(type.I.failure.times, type.II.failure.times))
[1] 130
```

Now form the approximately standard normal test statistic for the type I rank sum:

```
> (E.R.type.I =(n*(n+m+1))/2)
[1] 105
> (sd.R.type.I = sqrt( n*m*(n+m+1)/12))
[1] 13.22876
> (Z = (R.type.I - E.R.type.I)/sd.R.type.I)
[1] 1.889822
> (p.val = 2*(1-pnorm(Z)))
[1] 0.05878172
```

Confirm using wilcox.test():

```
> wilcox.test(type.I.failure.times, type.II.failure.times, exact=F, correct=F)
        Wilcoxon rank sum test

data:  type.I.failure.times and type.II.failure.times
W = 75, p-value = 0.05878
alternative hypothesis: true location shift is not equal to 0
```

(c) **Which of the methods, parametric or nonparametric, do you think is better in this case?**

Unless we are confident that the data are normally distributed, the nonparametric method is to be preferred. In this case, the results are quite similar.

(d) **Estimate π, the probability that a type I bearing will outlast a type II bearning?**

To estimate π, we compute the proportion of all pairs of type I and type II failure times for which the type I time is larger. From Rice, we know that sum of all such pairs can be computed from the rank sum:

$$U_1 = R_1 - \frac{n(n+1)}{2}$$

For this example, that can be computed as:

```
> (num.larger = R.type.I - (n*(n+1))/2)
```

```
[1] 75
```

Our estimate $\hat{\pi}$ is sum divided by the total number of pairs:

```
> (pi.hat = num.larger/(n*m))
```

```
[1] 0.75
```

(e) **Use the bootstrap to estimate the sampling distribution of $\hat{\pi}$ and its SE (visualize the bootstrap distribution using both a kernel density plot probability plot relative to normal distribution and comment on the bootstrap distribution.)**

We will estimate the sampling distribution of $\hat{\pi}$ by generated a large number of bootstrap resampled data sets, computing $\hat{\pi}$ on each and using the empirical distribution of those values as our sampling distribution estimate:

```
> B = 10000
> boot.data.type.I = matrix(sample(type.I.failure.times, n*B,replace=T), nrow=B)
> boot.data.type.II = matrix(sample(type.II.failure.times, m*B,replace=T), nrow=B)
> pi.hats = rep(0, B)
> for (i in 1:B) {
+   rank.sum = computeRankSum(boot.data.type.I[i,],
+           boot.data.type.II[i,])
+   num.larger = rank.sum - (n*(n+1))/2
+   pi.hats[i] = num.larger/(n*m)
+ }
> mean(pi.hats)
```

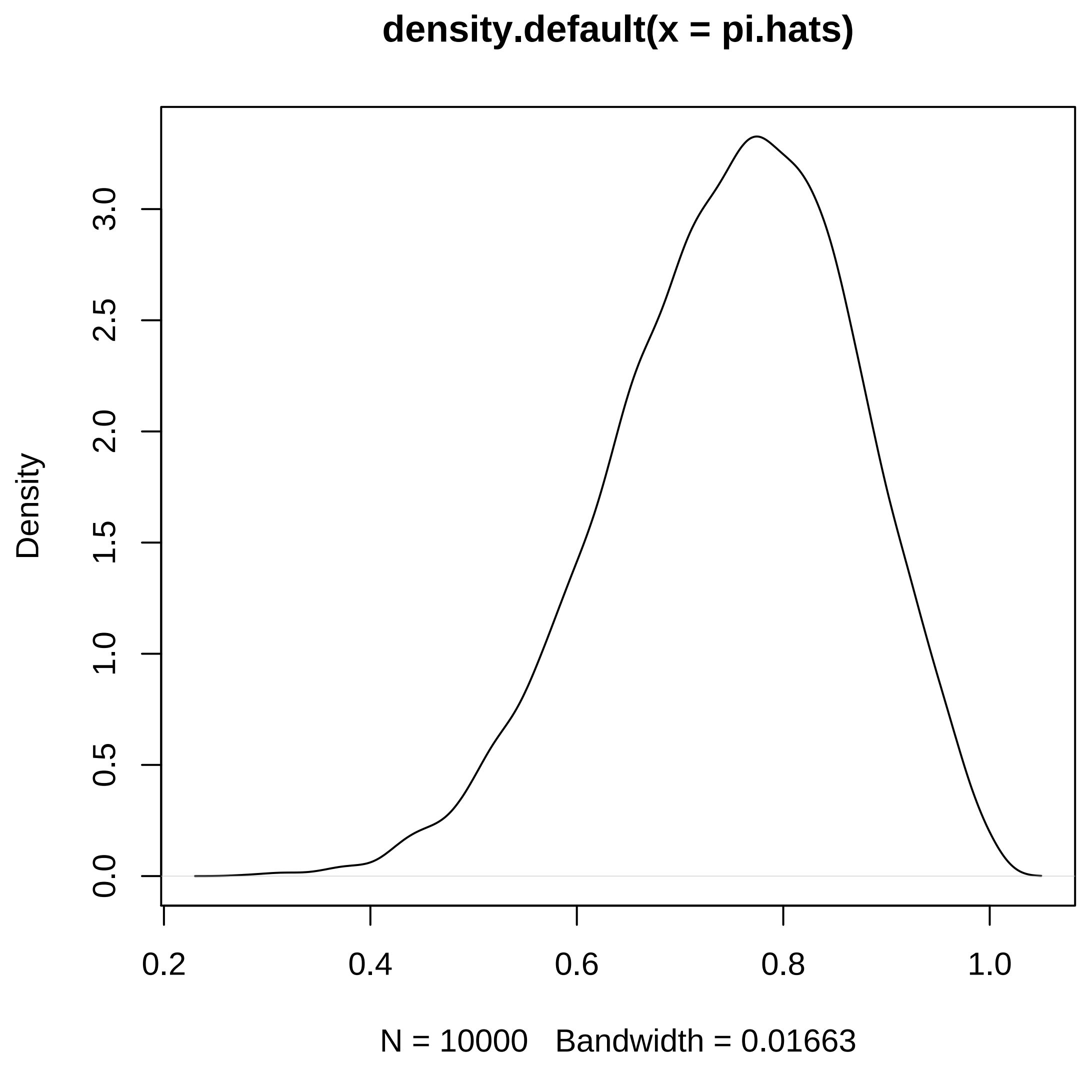
```
[1] 0.74939
```

```
> sd(pi.hats)
```

```
[1] 0.1166046
```

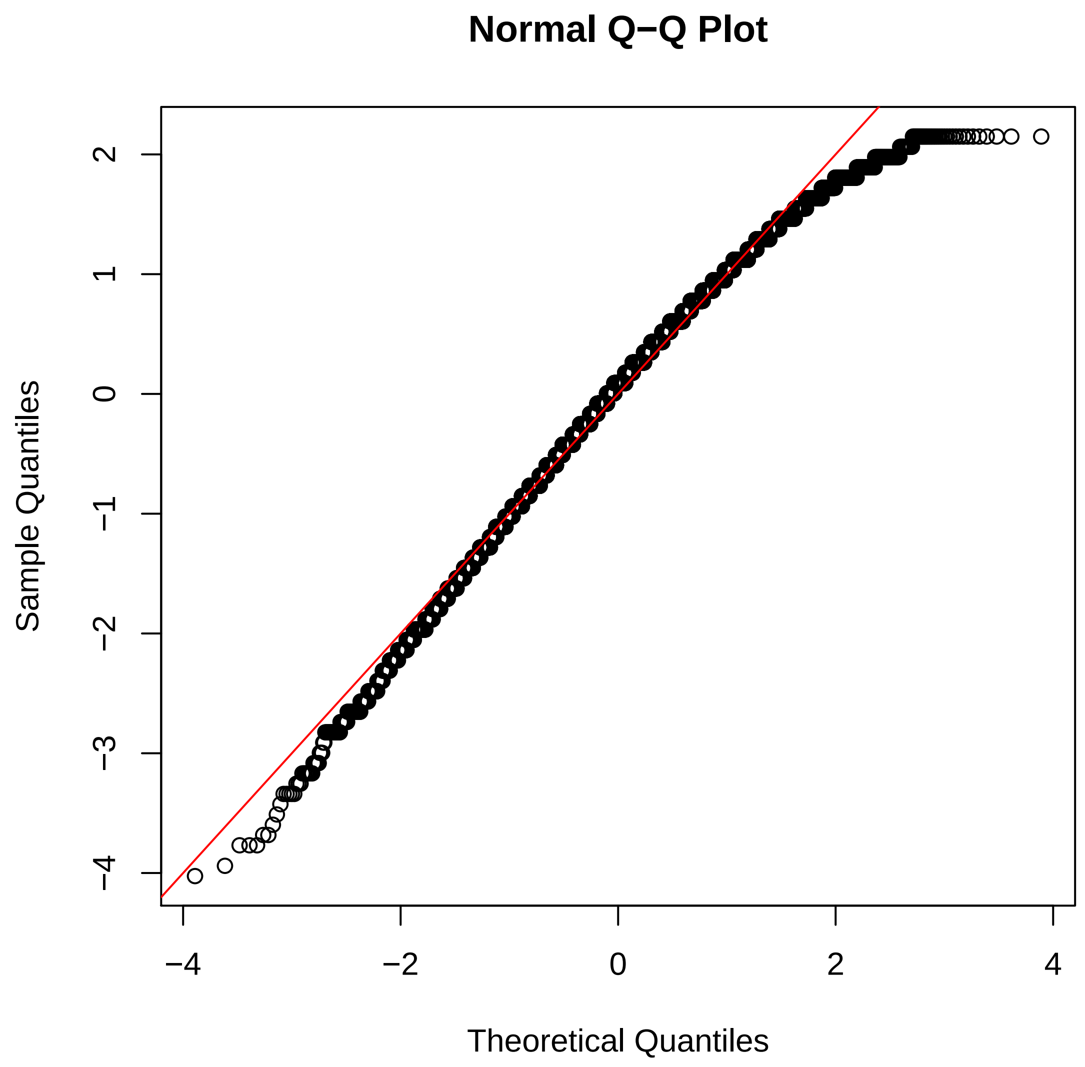
Visualize using kernel density:

```
> plot(density(pi.hats))
```



Visualize using probability plot. We'll first standardize the bootstrap estimates so they can be compared to N(0,1):

```
> stand.pi.hats = (pi.hats - mean(pi.hats))/sd(pi.hats)
> qqnorm(stand.pi.hats)
> abline(0,1, col="red")
```



Seems to match a normal fairly well expect in the tails (though that is the most uncertain region of a probability plot).

(f) **Use the bootstrap to find an approximate 90% CI for π (compute using both the basic and percentile bootstrap CI methods).**

First, need to rank the bootstrap estimates and find the 0.05 and 0.95 quantiles:

```
> ranked.pi.hats = sort(pi.hats)
> (quant.05 = ranked.pi.hats[500])
[1] 0.54
> (quant.95 = ranked.pi.hats[9500])
[1] 0.93
```

The percentile 90% bootstrap CI is defined directly by the 0.05 and 0.95 quantiles:

```
> (percentile.CI = c(quant.05, quant.95))
[1] 0.54 0.93
```

The basic bootstrap CI is $(2\hat{pi} - \pi^*_{0.95}, 2\hat{pi} - \pi^*_{0.05})$:

```
> (basic.CI = c(2*pi.hat - quant.95, 2*pi.hat - quant.05))
[1] 0.57 0.96
```

3. **Rice, Chapter 11, Problem 25: Referring to Example A in Section 11.2.1:**

   (a) **If the smallest observation for method B is made arbitrarily small, will the t test still reject?**

   It depends. Making the smallest observation in B smaller would just increase the observed difference in sample averages resulting in an even smaller p-value, however, there is a trade-off between decreasing the mean difference and increasing the sample variance. At a certain point the increase in the sample variance would dominate the increase in the mean difference and the test would no longer reject. On could plot the generated t statistic vs. the smallest method B value to determine the exact transition point between rejecting the null and accepting the null.

   (b) **If the largest observation for method B is made arbitrarily large, will the t test still reject?**

   No, it would not reject. Changing just one observation in B can increase the sample average by an arbitrary amount so it would be possible to give both samples the same average and a consequent larger p-value. The increase in sample variance in this case would also tend to increase the p-value.

   (c) **Answer the same questions for the Mann-Whitney test.**

   For the Mann-Whitney test, lowering the smallest value for method B would have no impact on the ranks since the smallest value in B is the smallest overall. Increasing the largest value in B would alter the ranks but the potential change in the rank sum is finite (once it is the largest value overall the rank is capped at 23) so it would not have a large overall impact but might change the rejection decision depending on the desired significance level.

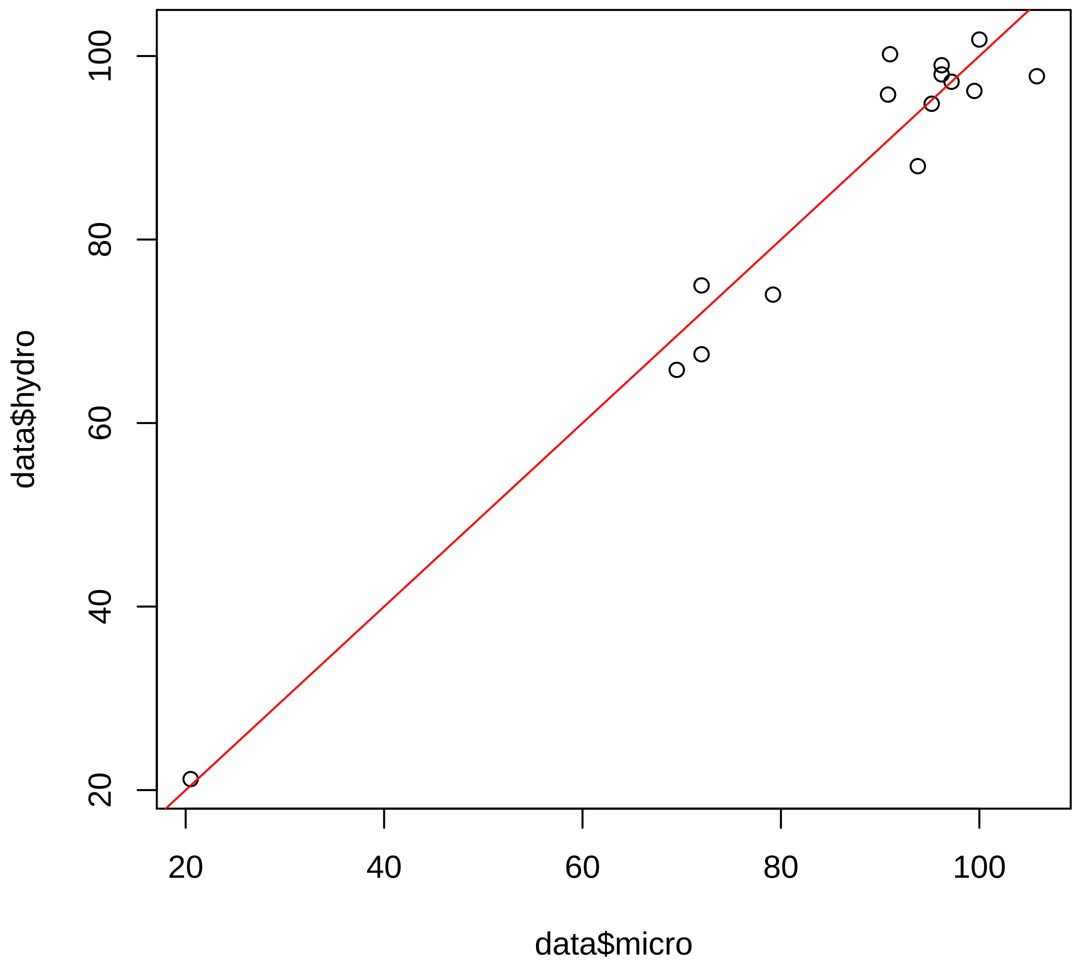
4. **Rice, Chapter 11, Problem 36: Lin, Sutton and Qurashi compared microbiological and hydroxylamine methods for the analysis of ampicillin dosages. In one series of experiments, pairs of tablets were analyzed by the two methods. The data in the following table give the percentages of the claimed amount of ampicillin found by the two methods in several pairs of tablets.**

```
> data = data.frame(
+         micro=c(97.2, 105.8, 99.5, 100, 93.8, 79.2, 72,
+                 72, 69.5, 20.5, 95.2, 90.8, 96.2, 96.2, 91),
+         hydro=c(97.2, 97.8, 96.2, 101.8, 88, 74, 75, 67.5, 65.8,
+                 21.2, 94.8, 95.8, 98, 99, 100.2))
> data

   micro hydro
1   97.2  97.2
2  105.8  97.8
3   99.5  96.2
4  100.0 101.8
```

```
5   93.8  88.0
6   79.2  74.0
7   72.0  75.0
8   72.0  67.5
9   69.5  65.8
10  20.5  21.2
11  95.2  94.8
12  90.8  95.8
13  96.2  98.0
14  96.2  99.0
15  91.0 100.2
```

Let's first plot the paired data to see if the pairing is justified:

```
> plot(data$micro, data$hydro)
> abline(0,1, col="red")
```



So, the values are correlated to the pairing appears justified. Does not look like there is a true mean difference.

(a) **What are $\bar{X} - \bar{Y}$ and $s_{\bar{X}-\bar{Y}}$?**

```
> diffs = data$micro - data$hydro
> (diffs.bar = mean(diffs))
[1] 0.44
> (diffs.bar.se = sd(diffs))
[1] 4.630767
```

(b) **If the pairing had been erroneously ignored and it had been assumed that the two samples were independent, what would have been the estimate of the SD of $\bar{X} - \bar{Y}$?**

Here we need to compute pooled SD assuming equal variance:

$$s_{\bar{X}-\bar{Y}} = \sqrt{\frac{(n-1)s_X^2 + (m-1)s_Y^2}{n+m-2}\frac{1}{n} + \frac{1}{m}}$$

```
> n=m=nrow(data)
> micro.var = var(data$micro)
> hydro.var = var(data$hydro)
> (var.pooled = ((n-1)*micro.var + (m-1)*hydro.var)/(n+m-2))
[1] 456.9207
> (indep.diffs.bar.se = sqrt(var.pooled*(1/n + 1/m)))
[1] 7.805303
```

So, a much larger standard error if we ignore the pairings.

(c) **Analyze the data to determine if there is a systematic difference between the two methods.**

We'll use both paired and unpaired nonparametric tests:

```
> wilcox.test(data$micro, data$hydro, paired=T)

        Wilcoxon signed rank test with continuity correction

data:  data$micro and data$hydro
V = 61, p-value = 0.6154
alternative hypothesis: true location shift is not equal to 0

> wilcox.test(data$micro, data$hydro, paired=F)

        Wilcoxon rank sum test with continuity correction

data:  data$micro and data$hydro
W = 106.5, p-value = 0.8194
alternative hypothesis: true location shift is not equal to 0
```

As suspected from the initial plot of the data, both methods accept the H_0 that $\mu_{micro} = \mu_{hydro}$.