Exercise 1

Exercise introduction

This is Exercise 1 in Part 2 of the course.

The purpose of the exercise is to cover calculations of central tendency and dispersion.

Importing excel data

The first part of this exercise was to create data in Excel and then do some aggregations of those data using Excel.

This part of the exercise is to import those raw data, filter out the aggregations, then re-do some of the aggregations using `R`.

```
library('dplyr')
rawData <- openxlsx::read.xlsx('../p02_inputs/ex1.xlsx')
rawData %<>%
    filter(Sample %in% c(1:10)) %>%
    select(x, y, z)
rawData
```

```
##    x y z
## 1  2 3 A
## 2  3 4 A
## 3  4 6 A
## 4  5 5 A
## 5  4 4 B
## 6  7 7 B
## 7  9 6 B
## 8  6 5 B
## 9  7 7 B
## 10 2 2 B
```

Aggregation functions in `R`

I will use the `dplyr` and `tidyverse` syntax.

```
rawData %>%
    summarize(
        meanX = mean(x)
        , meanY = mean(y)
        , medianX = median(x)
        , medianY = median(y)
```

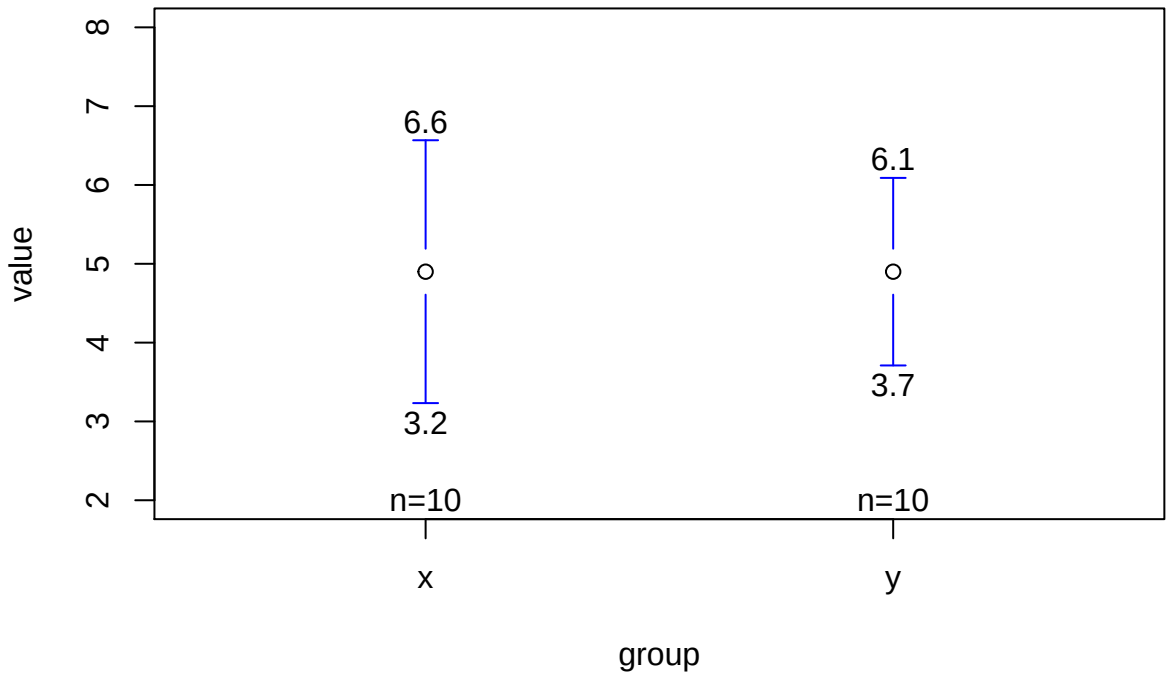
```
        , varianceX = var(x)
        , varianceY = var(y)
        , sdX = sd(x)
        , sdY = sd(y)
        , minX = min(x)
        , minY = min(y)
        , maxX = max(x)
        , maxY = max(y)
        )
```

```
##   meanX meanY medianX medianY varianceX varianceY      sdX     sdY minX minY
## 1   4.9   4.9     4.5       5  5.433333  2.766667 2.330951 1.66333    2    2
##   maxX maxY
## 1    9    7
```

Using `gplots::plotmeans`

The `gplots::plotmeans` function requires the data to be stacked differently, so I will need to pivot longer.

```
longData <-
    rawData %>%
    pivot_longer(cols = c(x,y), names_to = 'group')

longData %>%
    gplots::plotmeans(value ~ group, data = ., connect = F, xlab = 'group', ylab = 'value', ci.label = T
```

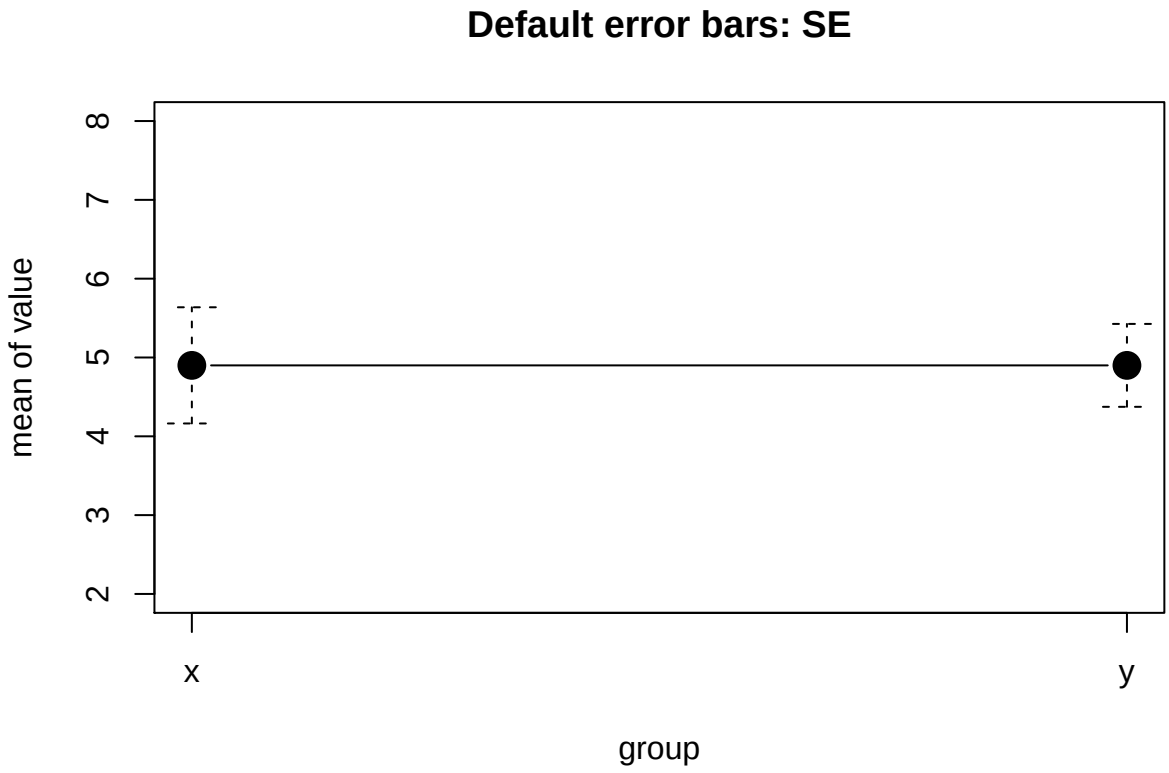


The `mean.labels` parameter adds a label for the means, but does not remove the default circle and therefore the graph looks cluttered. This is not a particularly pretty figure.

Using `RcmdrMisc::plotMeans`

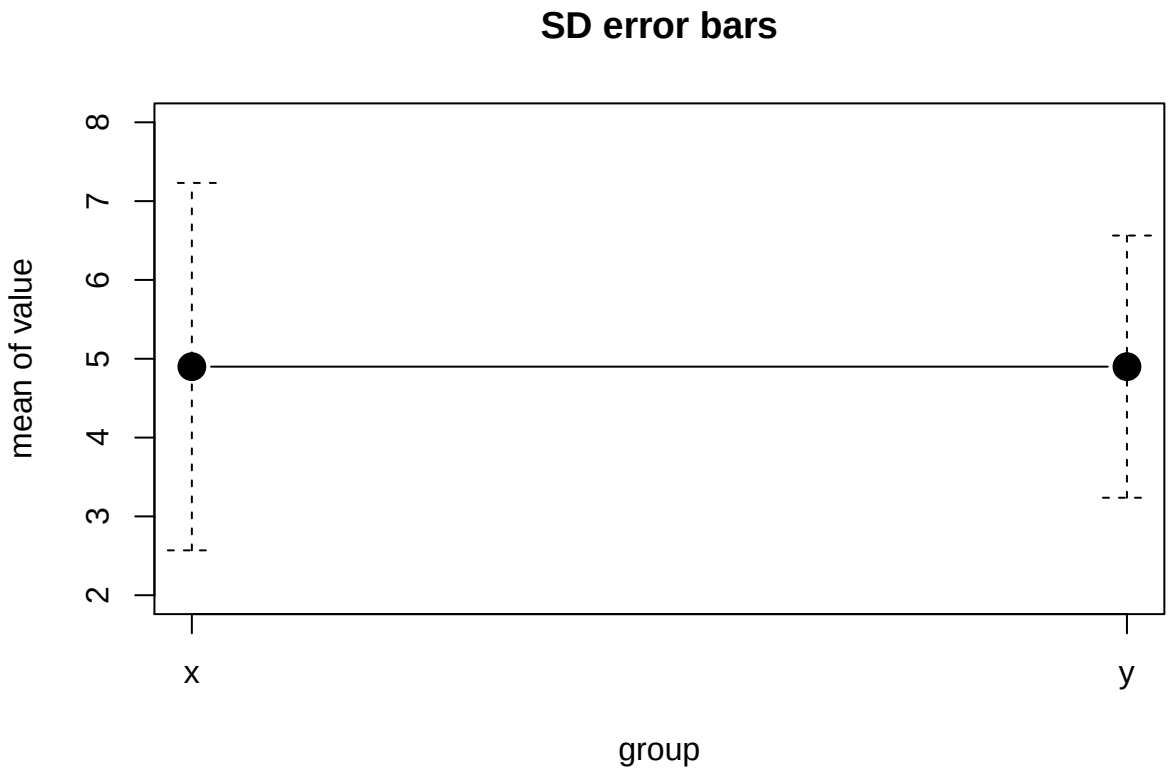
There is an alternative function called `plotMeans` from the `RcmdrMisc` package. This one comes with the `error.bars` param that allows you to modify if the error bars should represent the standard deviation or the standard error.

```
with(longData
     , RcmdrMisc::plotMeans(
         response = value
         , factor1 = group
         , ylim = c(2, 8)
         , main = 'Default error bars: SE'
         )
     )
```




```
with(longData
     , RcmdrMisc::plotMeans(
         response = value
         , factor1 = group
         , ylim = c(2, 8)
         , error.bars = 'sd'
         , main = 'SD error bars'
```

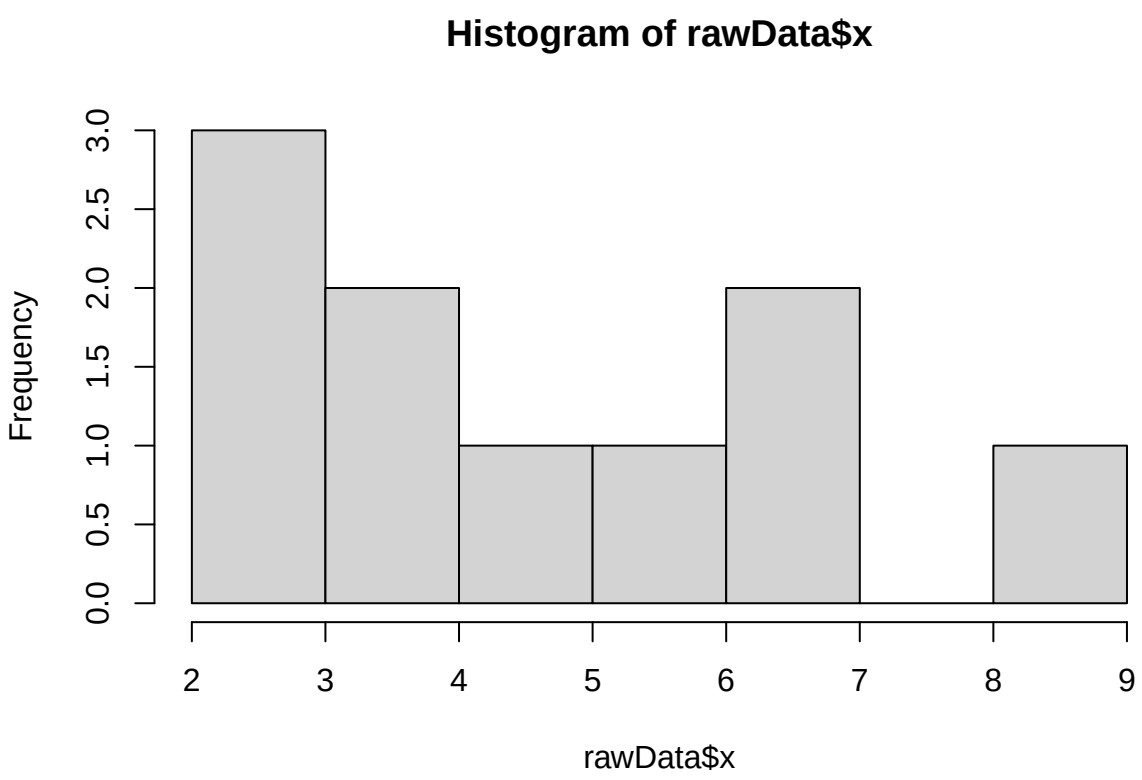
```
        )
    )
```



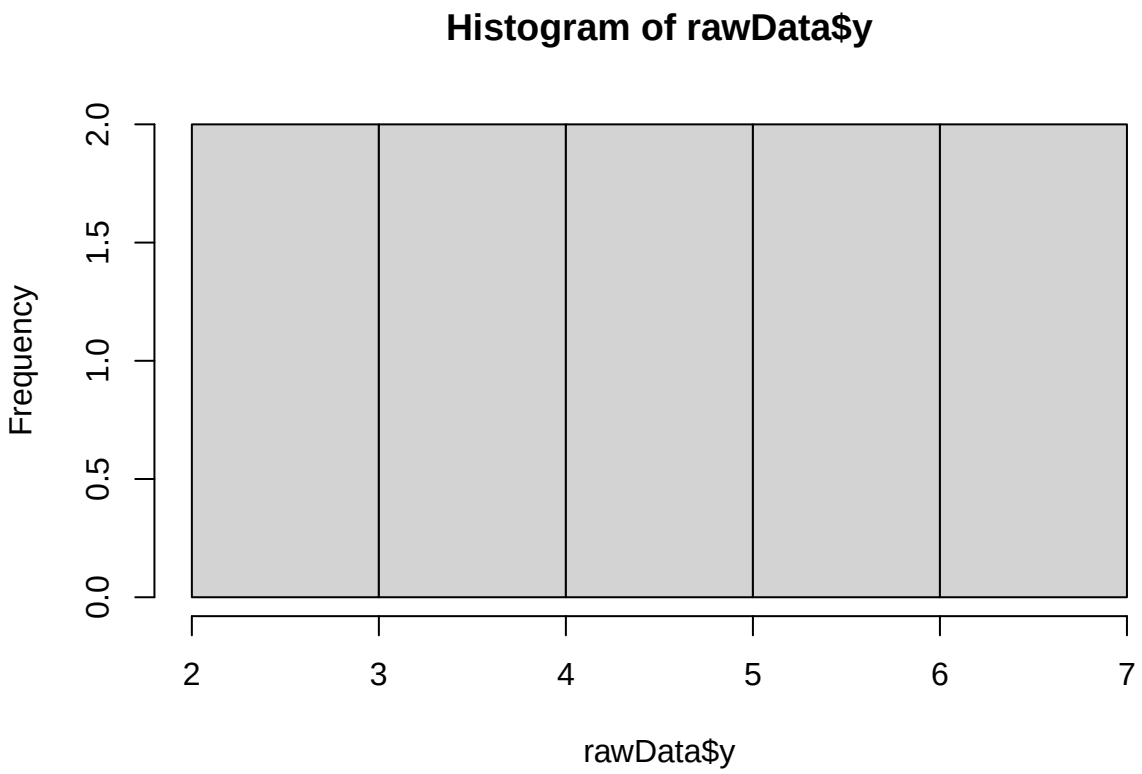
Visually assessing normality with `hist`

Although the Shapiro-Wilk test formally assesses normality, one may get a sense of normality with histograms.

```
hist(rawData$x)
```




```
hist(rawData$y)
```



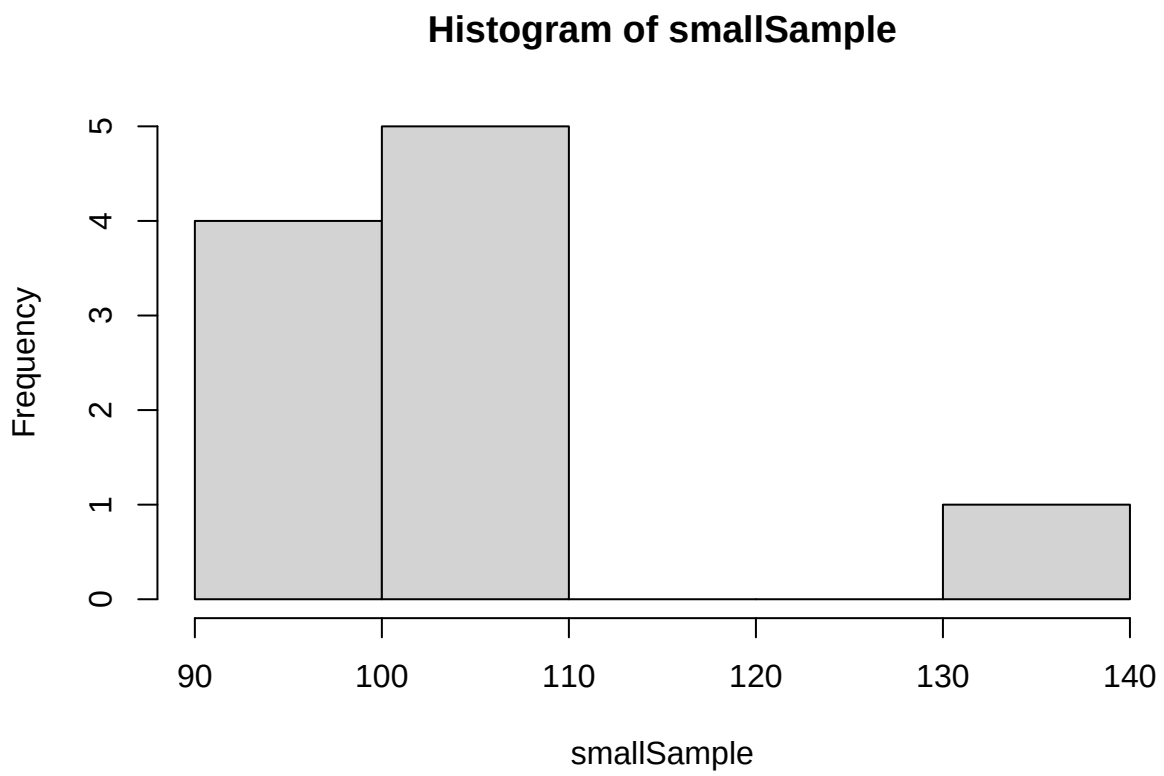
The data do not appear to be normally distributed, but with a sample size of only 10, it is hard to observe normal distribution anyway.

How does SD and SEM change as the sample size increases?

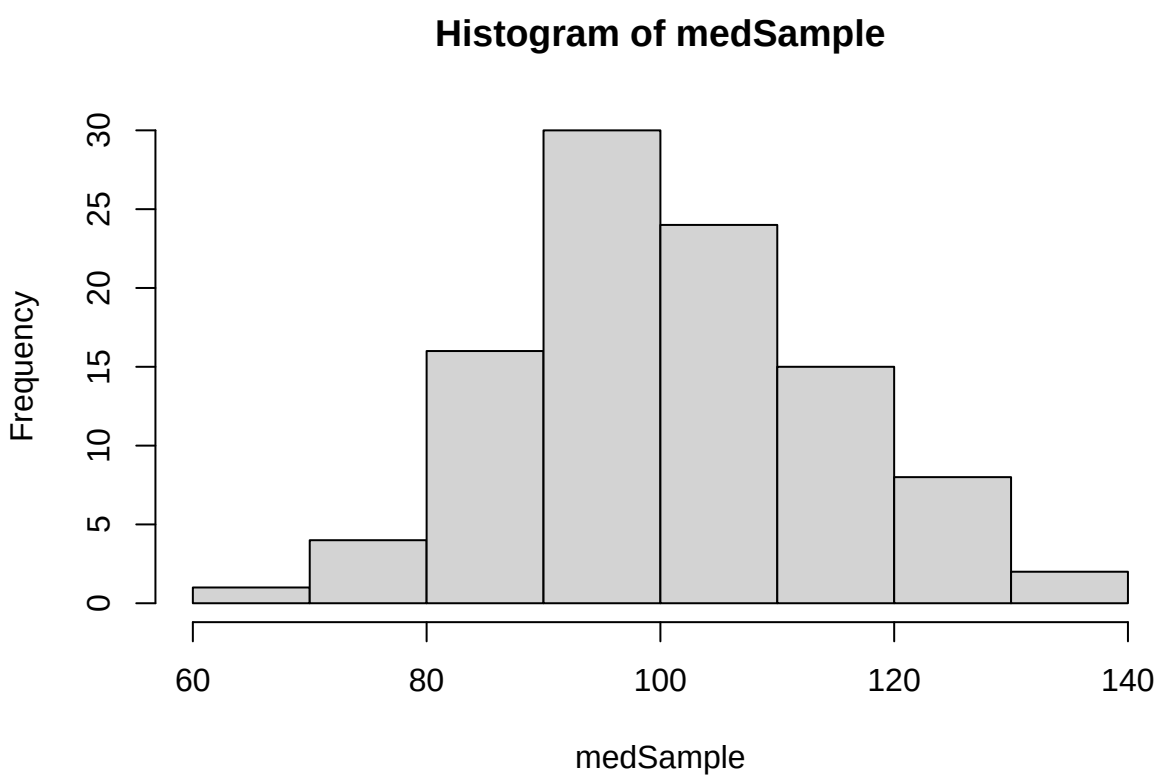
Now I will generate random normally-distributed data with different sample sizes and observe how the histograms better approximate normal distributions.

Then I will examine how the SD and SEM change with increasing sample sizes although the underlying data were the same.

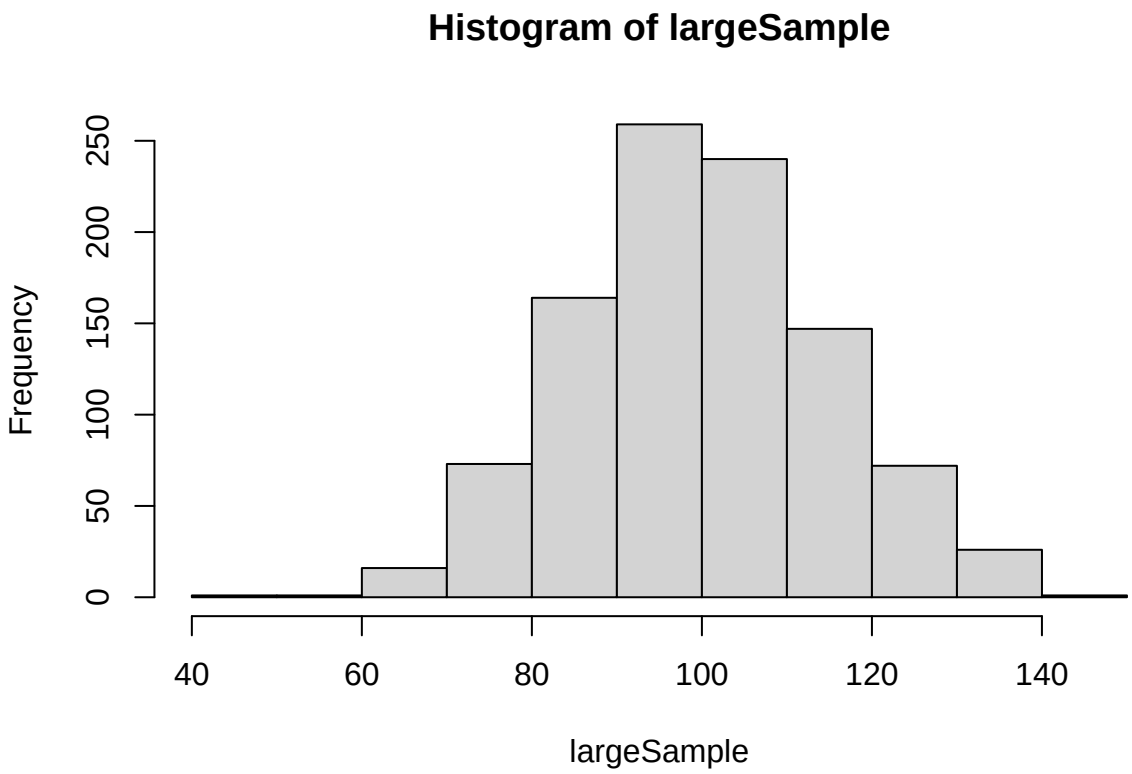
```
set.seed(4123)
largeSample <- rnorm(1000, mean = 100, sd = 15)
medSample <- sample(largeSample, 100)
smallSample <- sample(medSample, 10)

hist(smallSample)
```



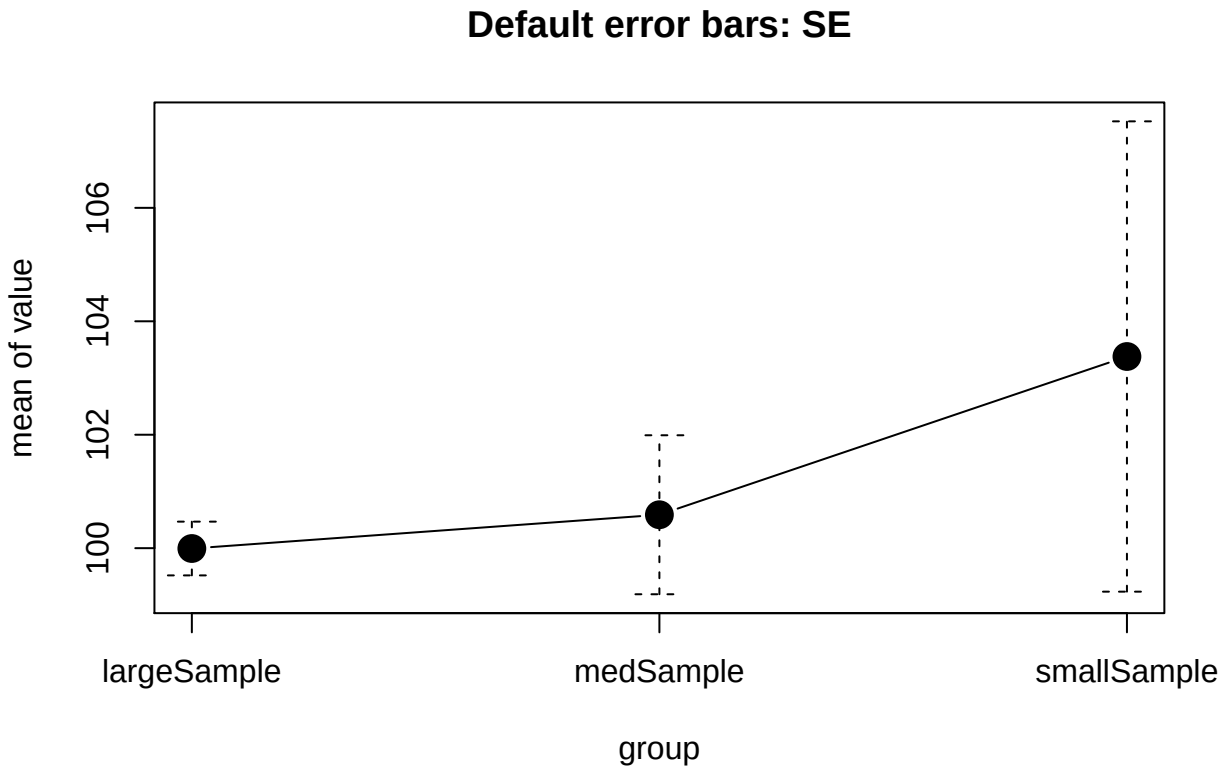

```
hist(medSample)
```



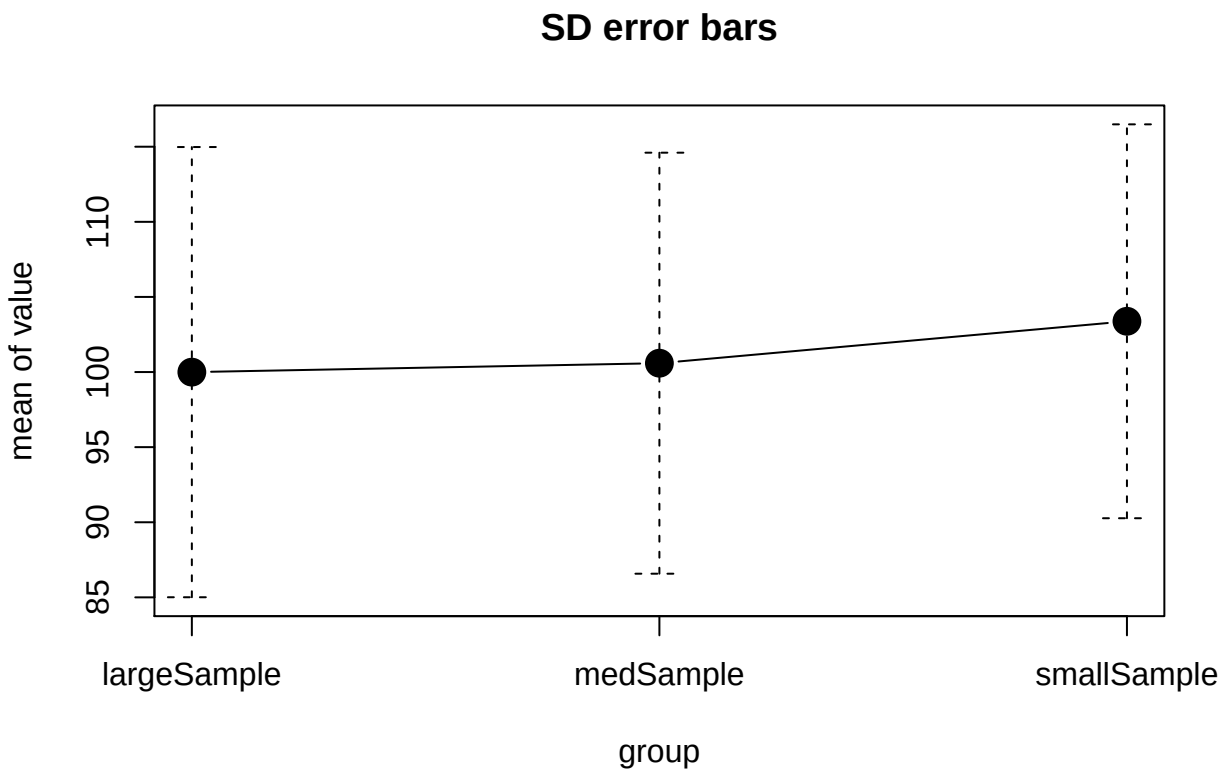

```
hist(largeSample)
```




```
longSamples <-
    data.frame(
        group = c(
            rep('largeSample', 1000)
            , rep('medSample', 1000)
            , rep('smallSample', 1000)),
        value = c(
            largeSample
            , medSample, rep(NA, 900)
            , smallSample, rep(NA, 990)
            )
    )

with(longSamples
 , RcmdrMisc::plotMeans(
     response = value
     , factor1 = group
     , main = 'Default error bars: SE'
     )
 )
```




```
with(longSamples
 , RcmdrMisc::plotMeans(
     response = value
     , factor1 = group
     , error.bars = 'sd'
     , main = 'SD error bars'
     )
 )
```



ggplot2 for histograms

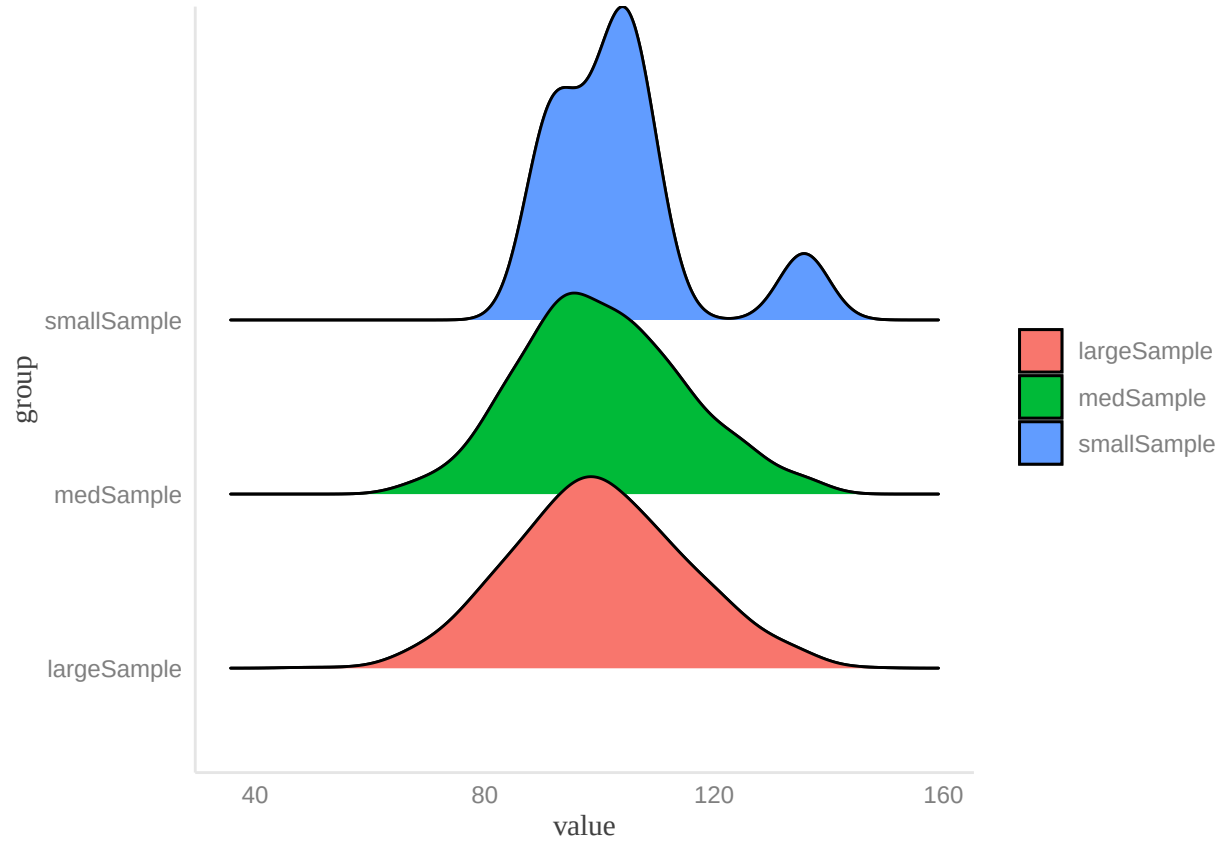
```
activatePkgs('ggridges')
```

```
## Loading required package: ggridges
```

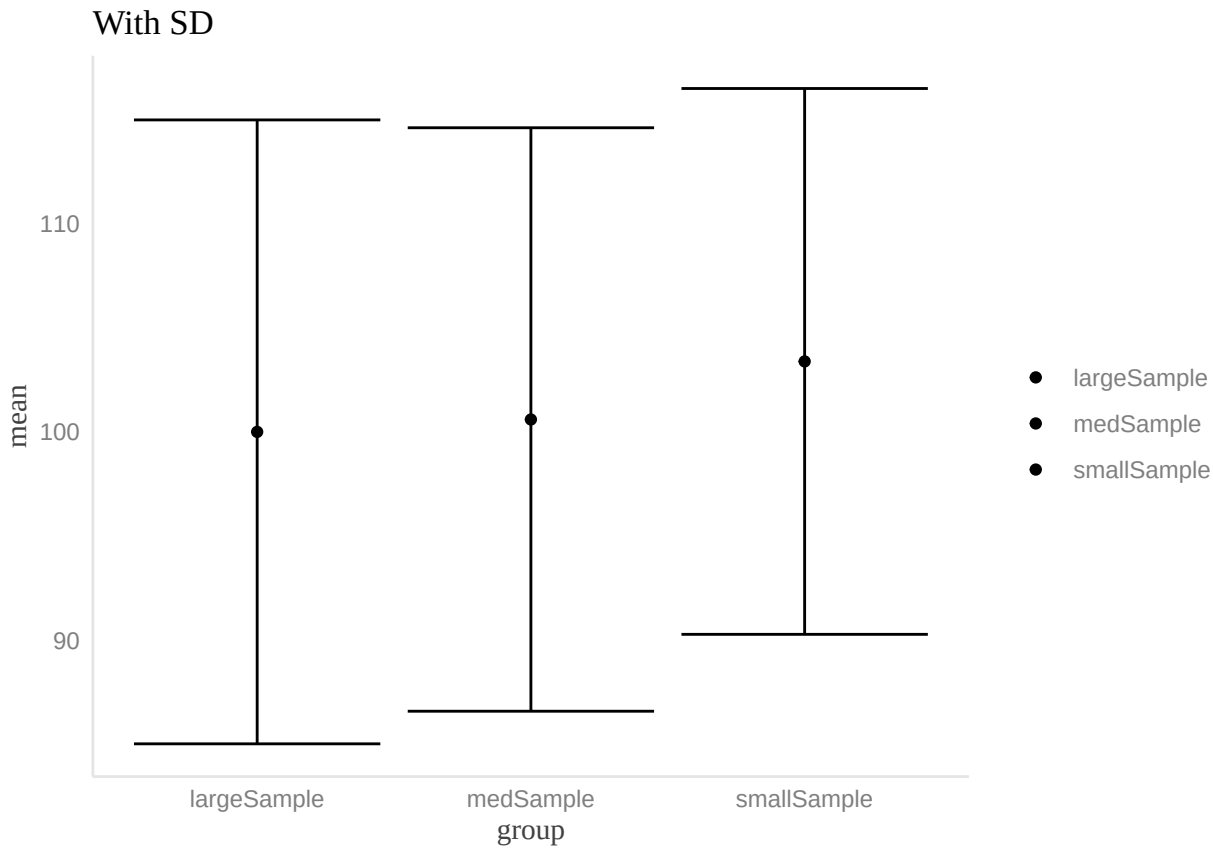
```
longSamples %>%
    ggplot(aes(x = value, y = group, fill = group)) +
    geom_density_ridges() +
    stat_density_ridges()
```

```
## Picking joint bandwidth of 4.38
```

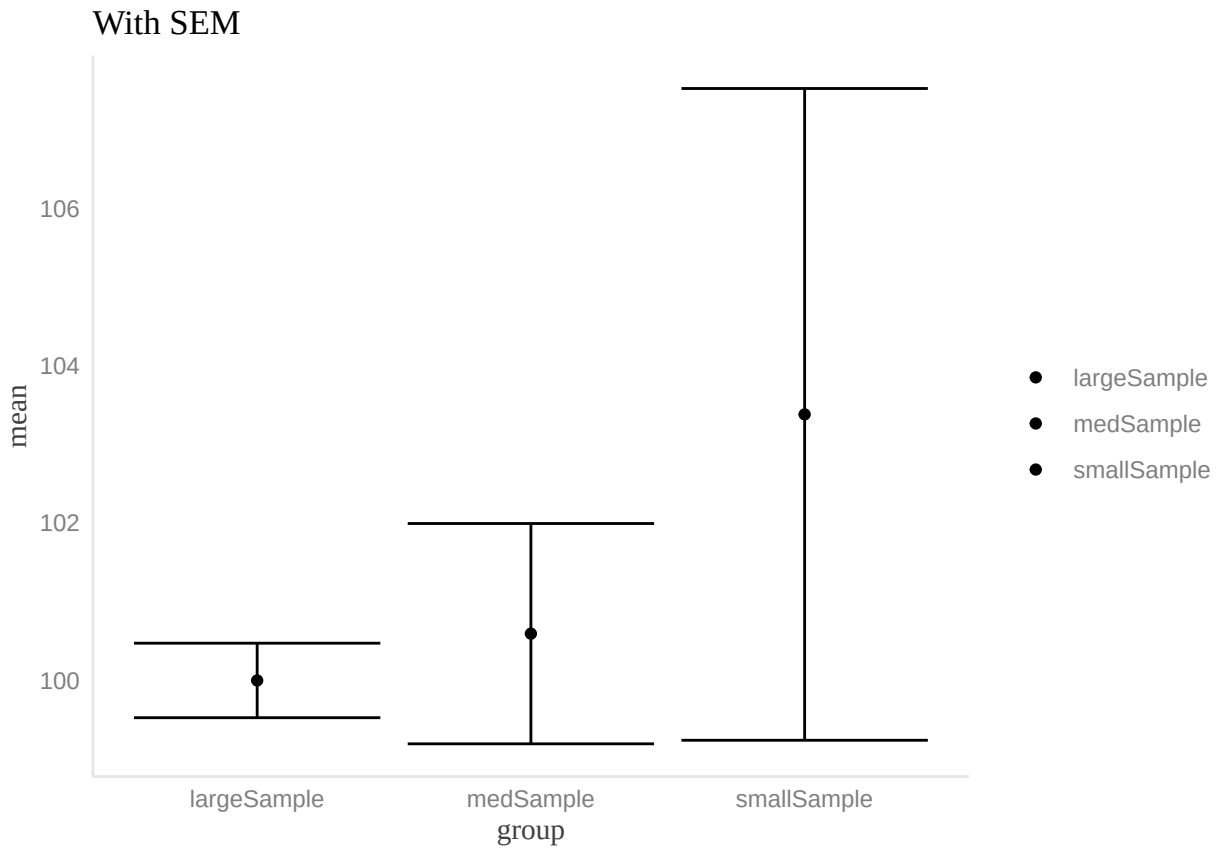
```
## Picking joint bandwidth of 4.38
```




```
longSamples %>%
    group_by(group) %>%
    summarize(mean = mean(value, na.rm = T)
              , sd = sd(value, na.rm =T)
              , sem = sd(value, na.rm = T) / sqrt(length(na_remove(value)))) %>%
    ggplot(aes(x = group, y = mean, fill = group)) +
    geom_point() +
    geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd)) +
    labs(title = 'With SD')
```




```
longSamples %>%
    group_by(group) %>%
    summarize(mean = mean(value, na.rm = T)
              , sd = sd(value, na.rm =T)
              , sem = sd(value, na.rm = T) / sqrt(length(na_remove(value)))) %>%
    ggplot(aes(x = group, y = mean, fill = group)) +
    geom_point() +
    geom_errorbar(aes(ymin = mean - sem, ymax = mean + sem)) +
    labs(title = 'With SEM')
```



Key learnings

- If using an RProfile to load libraries, the standard R libraries other than `base` are loaded last, including `stats::filter`. This makes using `dplyr::filter` verbose. The workaround is using the `conflicted` package and `conflict_prefer('filter', 'dplyr')` in the RProfile after loading the libraries.
- With increased sample size, the standard deviation will remain approximately the same, but the standard error of the mean will decrease. Since the confidence interval is related to the SEM, this means that with greater sample sizes you have greater confidence about your point estimate, even if the variation around that point estimate is large.
- The `RcmdrMisc::plotMeans` function allows you to plot standard error or standard deviation around a mean in a relatively straight-forward manner

Unresolved questions

- Is there a way to use the `conflicted` and `conflict_prefer` function to load `renv` in a way that it does not interfere with R Markdown's use of the `load` function? Both `conflict_prefer("load", "renv")` and `conflict_prefer("load", "base")` fail...
- Why does this `for` loop plot every data point instead of every sample?:

```
for (sample in c(smallSample, medSample, largeSample)) {
    hist(sample)
```

```
}
```