# R statistics 2.0

Aleksandar Beatovic

4/6/2020

## 1.1 Measures of center

```r
x <- c(175, 176, 182, 165, 167, 172, 175, 196, 158, 172)

#mean
xmean <- sum(x)/length(x)
print(c(mean(x), xmean))
```

```
## [1] 173.8 173.8
```

```r
#median
xmed <- (sort(x)[length(x)/2] + sort(x)[length(x)/2 +1])/2
print(c(median(x), xmed))
```

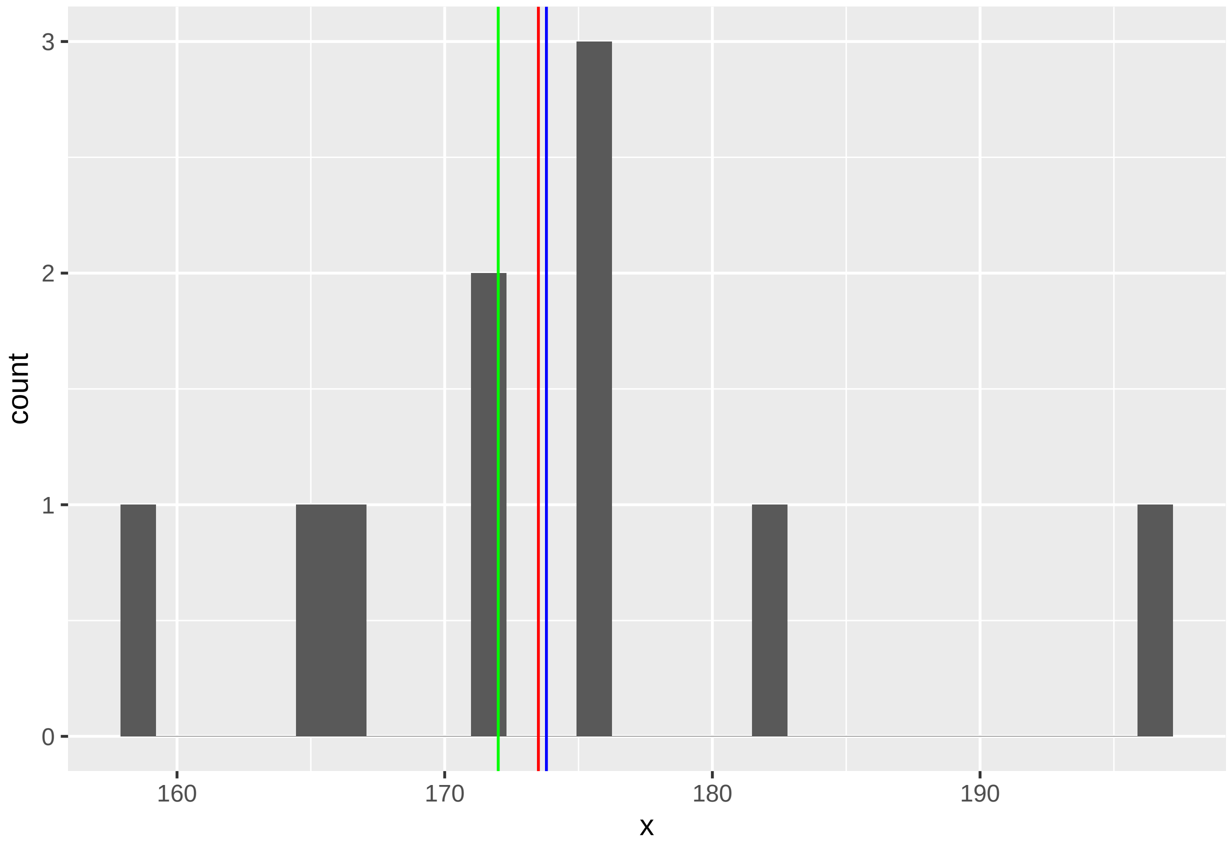
```
## [1] 173.5 173.5
```

```r
#mode
xmod <- as.numeric(names(sort(table(round(x)), decreasing = TRUE))[1])
print(xmod)
```

```
## [1] 172
```

```r
ggplot() + aes(x) + geom_histogram() + geom_vline(xintercept = xmed,color = "red")+ geom_vline(xintercep
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




## 1.2 Measures of center - spoiled

```r
x_s <- c(175, 176, 182, 165, 167, 172, 175, 196, 158, 172, 195, 198)

#mean
xmean_s <- sum(x_s)/length(x_s)
print(c(mean(x_s), xmean_s))
```

```
## [1] 177.5833 177.5833
```

```r
#median
xmed_s <- (sort(x_s)[length(x_s)/2] + sort(x_s)[length(x_s)/2 +1])/2
print(c(median(x_s), xmed_s))
```

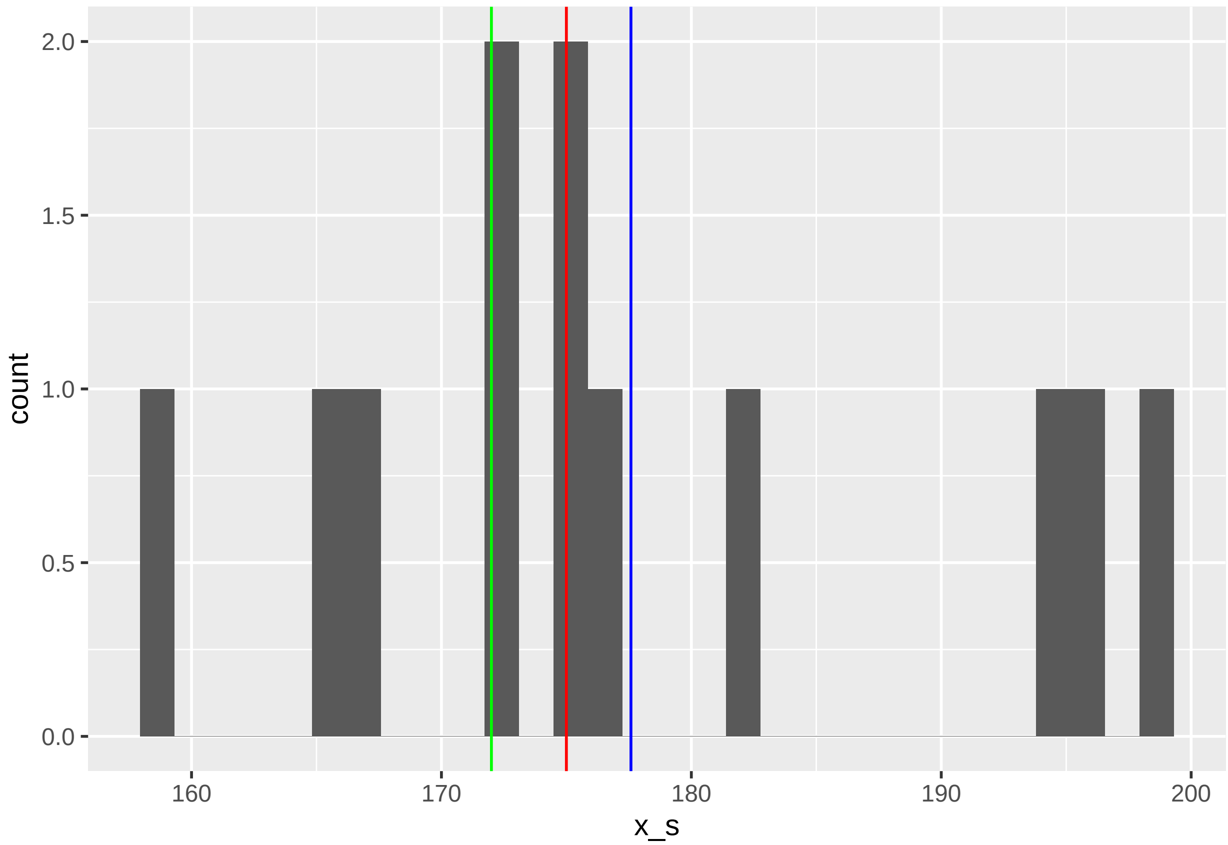
```
## [1] 175 175
```

```r
#mode
xmod_s <- as.numeric(names(sort(table(round(x_s)), decreasing = TRUE))[1])
print(xmod_s)
```

```
## [1] 172
```

```r
ggplot() + aes(x_s) + geom_histogram() + geom_vline(xintercept = xmed_s,color = "red")+ geom_vline(xinte
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




## 2.1 Measures of spread

```
variance <- function(heights, mean) {
  tmp <- vector()
  for (variable in heights) {
    tmp <-c(tmp,((variable - mean)^2))
    }

  return(sum(tmp) / (length(heights) -1))
}

print(c(variance(x,xmean),var(x)))
```

```
## [1] 105.2889 105.2889
```

```
print(c(sqrt(variance(x,xmean)),sd(x)))
```

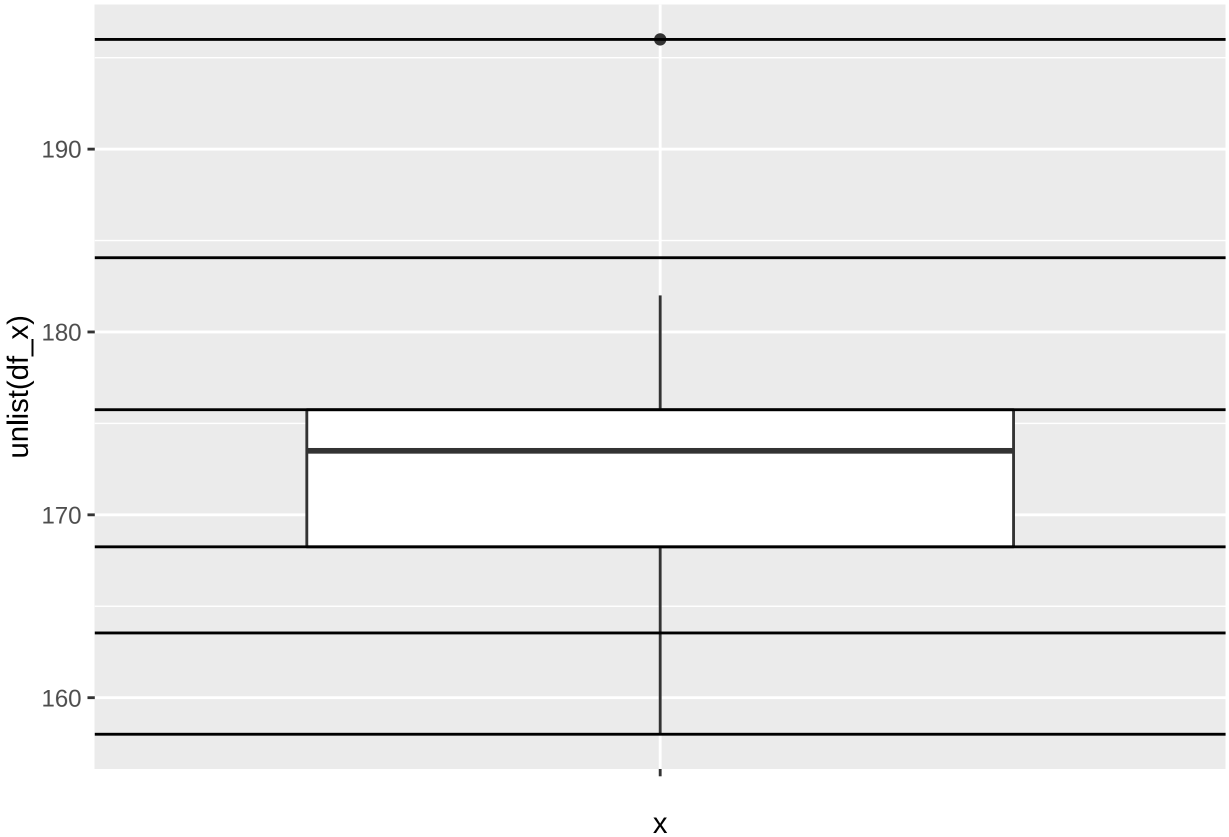
```
## [1] 10.26104 10.26104
```

```
quantile(x)
```

```
##     0%    25%    50%    75%   100%
## 158.00 168.25 173.50 175.75 196.00
```

```
df_x <- data.frame(x)
ggplot(data = df_x, aes(x = "", y = unlist(df_x))) + geom_boxplot() + geom_hline(yintercept=min(x)) + g
```

```
geom_hline(yintercept=mean(x) + sd(x)) +
geom_hline(yintercept=mean(x) - sd(x))
```




"'

## 2.2 Measures of spread - spoiled

```
print(c(variance(x_s,xmean_s),var(x_s)))
```

```
## [1] 164.6288 164.6288
```

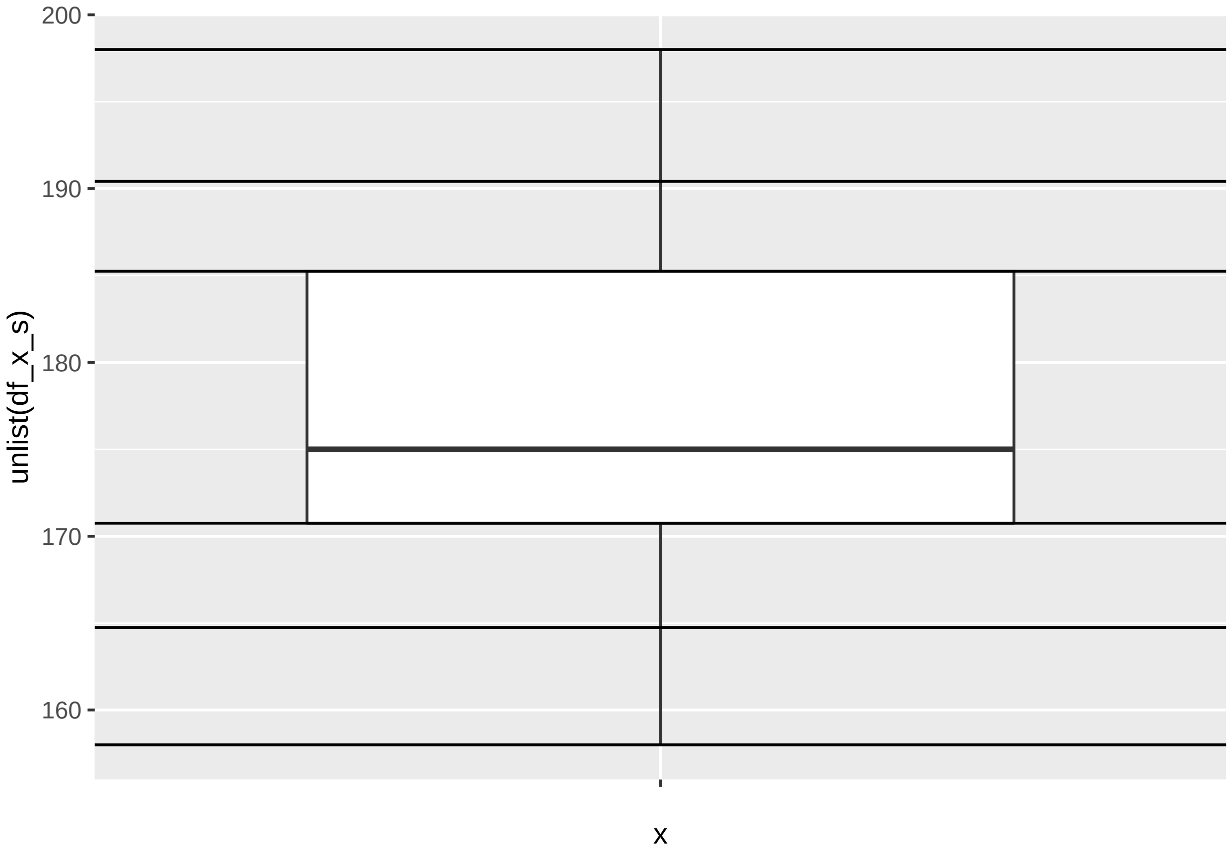
```
print(c(sqrt(variance(x_s,xmean_s)),sd(x_s)))
```

```
## [1] 12.83078 12.83078
```

```
quantile(x_s)
```

```
##     0%    25%    50%    75%   100%
## 158.00 170.75 175.00 185.25 198.00
```

```
df_x_s <- data.frame(x_s)
ggplot(data = df_x_s, aes(x = "", y = unlist(df_x_s))) + geom_boxplot() + geom_hline(yintercept=min(x_s)
geom_hline(yintercept=mean(x_s) + sd(x_s)) +
geom_hline(yintercept=mean(x_s) - sd(x_s))
```




## 3. Properties

So despite these properties holding true (as seen by table), the first two lines output false due to differing by a very small amount due to the computer's representation of numbers, also known as floating point error, which is confirmed by subtracting them.

```
print(mean(x-100) == mean(x) - 100)
```

```
## [1] FALSE
```

```
print(mean(x / 100) == mean(x) / 100)
```

```
## [1] FALSE
```

```
print(abs(sum(x - mean(x)) - 0) < 0.000000001)
```

```
## [1] TRUE
```

```
#float point error
print(mean(x-100) -  (mean(x) - 100))
```

```
## [1] -1.421085e-14
```

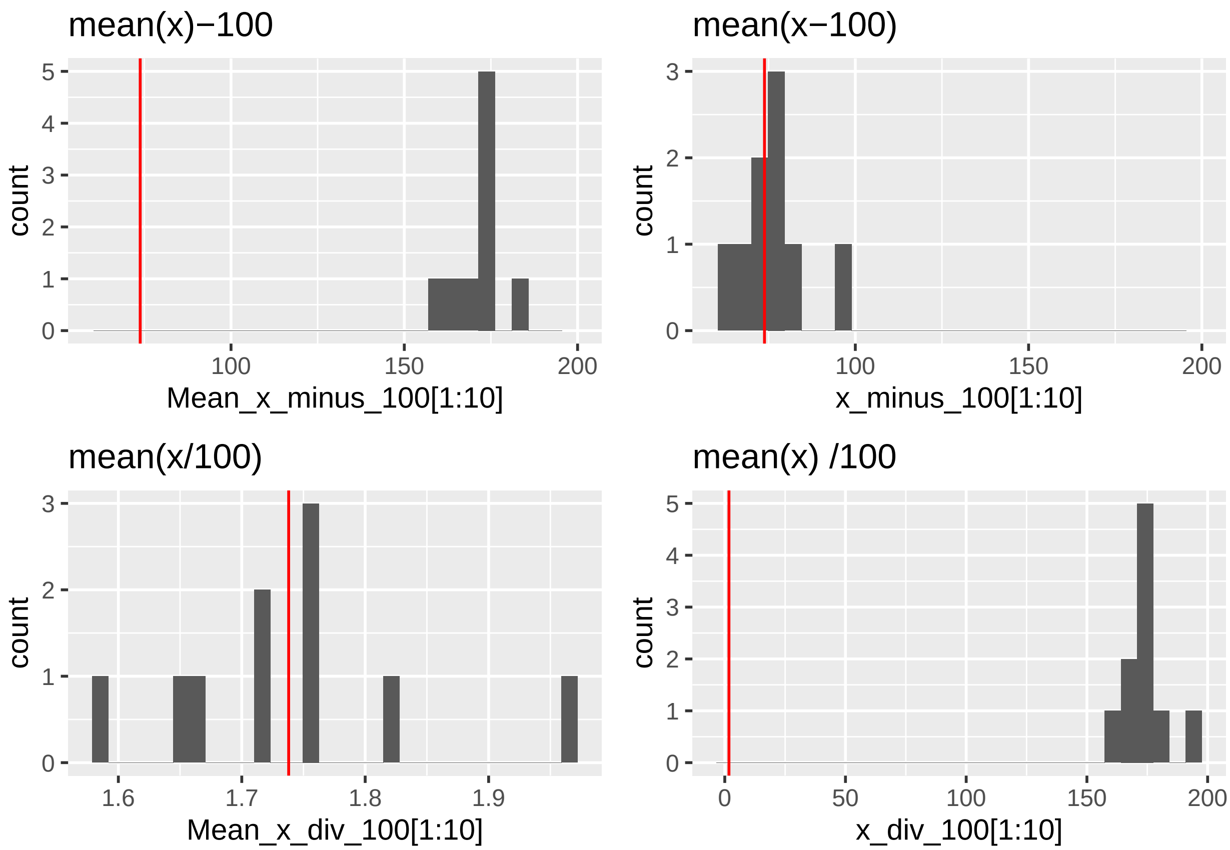
```
Mean_x_minus_100 <- c(x,mean(x)- 100)
x_minus_100 <- c((x-100), mean(x-100))
Mean_x_div_100 <- c(x/100, mean(x/100))
x_div_100 <- c(x, mean(x)/100)
```

```r
df_properties <- data.frame(Mean_x_minus_100, x_minus_100, Mean_x_div_100, x_div_100)
rownames(df_properties)[rownames(df_properties) == "11"] <- "Mean"
df_properties
```

```
##      Mean_x_minus_100 x_minus_100 Mean_x_div_100 x_div_100
## 1               175.0        75.0          1.750   175.000
## 2               176.0        76.0          1.760   176.000
## 3               182.0        82.0          1.820   182.000
## 4               165.0        65.0          1.650   165.000
## 5               167.0        67.0          1.670   167.000
## 6               172.0        72.0          1.720   172.000
## 7               175.0        75.0          1.750   175.000
## 8               196.0        96.0          1.960   196.000
## 9               158.0        58.0          1.580   158.000
## 10              172.0        72.0          1.720   172.000
## Mean             73.8        73.8          1.738     1.738
```

```r
a <- ggplot() + aes(Mean_x_minus_100[1:10]) + geom_histogram() + geom_vline(xintercept = Mean_x_minus_10

b <- ggplot() + aes(x_minus_100[1:10]) + geom_histogram() + geom_vline(xintercept = x_minus_100[11],col

c <- ggplot() + aes(Mean_x_div_100[1:10]) + geom_histogram() + geom_vline(xintercept = Mean_x_div_100[1

d <- ggplot() + aes(x_div_100[1:10]) + geom_histogram() + geom_vline(xintercept = x_div_100[11],color =

print(ggarrange(a,b,c,d ))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom_bar).

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 1 rows containing non-finite values (stat_bin).

## Warning: Removed 2 rows containing missing values (geom_bar).

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
var(x - 100) == var(x)
```

```
## [1] TRUE
```

```r
var(x / 100) == var(x) / 10000
```

```
## [1] FALSE
```

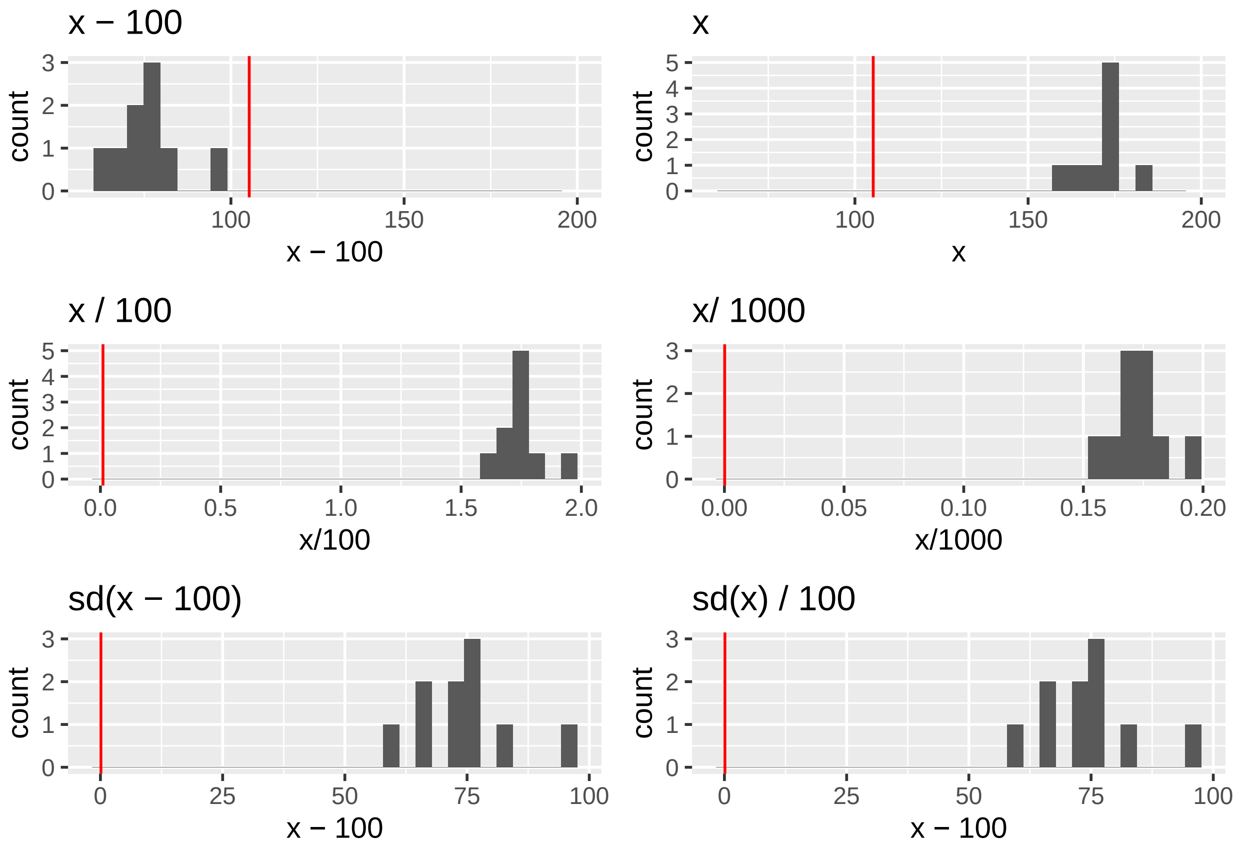
```r
sd(x / 100) == sd(x) / 100
```

```
## [1] FALSE
```

```r
e <- ggplot() + aes(x-100) + geom_histogram() + geom_vline(xintercept = var(x - 100),color = "red") +gg


f <- ggplot() + aes(x) + geom_histogram() + geom_vline(xintercept = var(x),color = "red") +ggtitle(label

g <- ggplot() + aes(x / 100) + geom_histogram() + geom_vline(xintercept = var(x / 100),color = "red") +

h <- ggplot() + aes(x / 1000) + geom_histogram() + geom_vline(xintercept = var(x / 1000),color = "red")

i <- ggplot() + aes(x-100) + geom_histogram() + geom_vline(xintercept = sd(x / 100),color = "red") +ggt

j <- ggplot() + aes(x-100) + geom_histogram() + geom_vline(xintercept = sd(x) / 100 ,color = "red") +gg

print(ggarrange(e,f,g,h,i,j, ncol = 2, nrow = 3))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom_bar).

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




4.0 Normal distribution

```r
pnorm(156,175,10)
```

```
## [1] 0.02871656
```

```r
pnorm(156,175,10, lower.tail = FALSE)
```

```
## [1] 0.9712834
```

```r
pnorm(168,175,10, lower.tail = FALSE) - pnorm(172,175,10)
```

```
## [1] 0.3759478
```

4.1 check the properties of 1-2-3-sd's for standard normal distribution using pnorm()

```r
pnorm(1) - pnorm(-1)
```

```
## [1] 0.6826895
```

```r
pnorm(2) - pnorm(-2)
```

```
## [1] 0.9544997
```

```r
pnorm(3) - pnorm(-3)
```

```
## [1] 0.9973002
```

4.2 generate sample using rnorm() from N(175, 10), find mean ans sd;

```r
set.seed(42)
sample <- rnorm(1000,175,10)
mean(sample)
```

```
## [1] 174.7418
```

```r
sd(sample)
```

```
## [1] 10.02521
```

```r
normalized_sample <- (x - mean(sample))/sd(sample)
mean(normalized_sample)
```

```
## [1] -0.09393873
```

```r
sd(normalized_sample)
```

```
## [1] 1.023523
```

### 5.0 Generate large population (n ~ 100 000 - 1 000 000) distributed as N(0, 1)

### Sample from population k observations for 30 times - you will have set of 30 samples.

### For each sample calculate mean. For the set calculate means of means, sd of means, SE.

### Create table with k, mean of means, sd of means, SE.

### Visualize distribution of means with histogram and lines for mean of means and SE.

### 5.1 k = 10

### 5.2 k = 50

### 5.3 k = 100

### 5.4 k = 500

### 5.5 Compare results

```
data <- function(k){
data_k <- length(k[,1])
data_mean <- mean(unlist(lapply(k, mean)))
data_sd <- sd(unlist(lapply(k, mean)))
data_se <- sd(unlist(lapply(k, mean)))/(sqrt(length(k)))
return(c( data_k, data_mean, data_sd, data_se ))
}
```

```
population <- rnorm(200000, 0 , 1)
k_10 <- replicate (30, sample(population, 10))
k_50 <- replicate (30, sample(population, 50))
k_100 <- replicate (30, sample(population, 100))
k_500 <- replicate (30, sample(population, 500))

k_10_data <- data(k_10)
k_50_data <- data(k_50)
k_100_data <- data(k_100)
k_500_data <- data(k_500)
```

Table

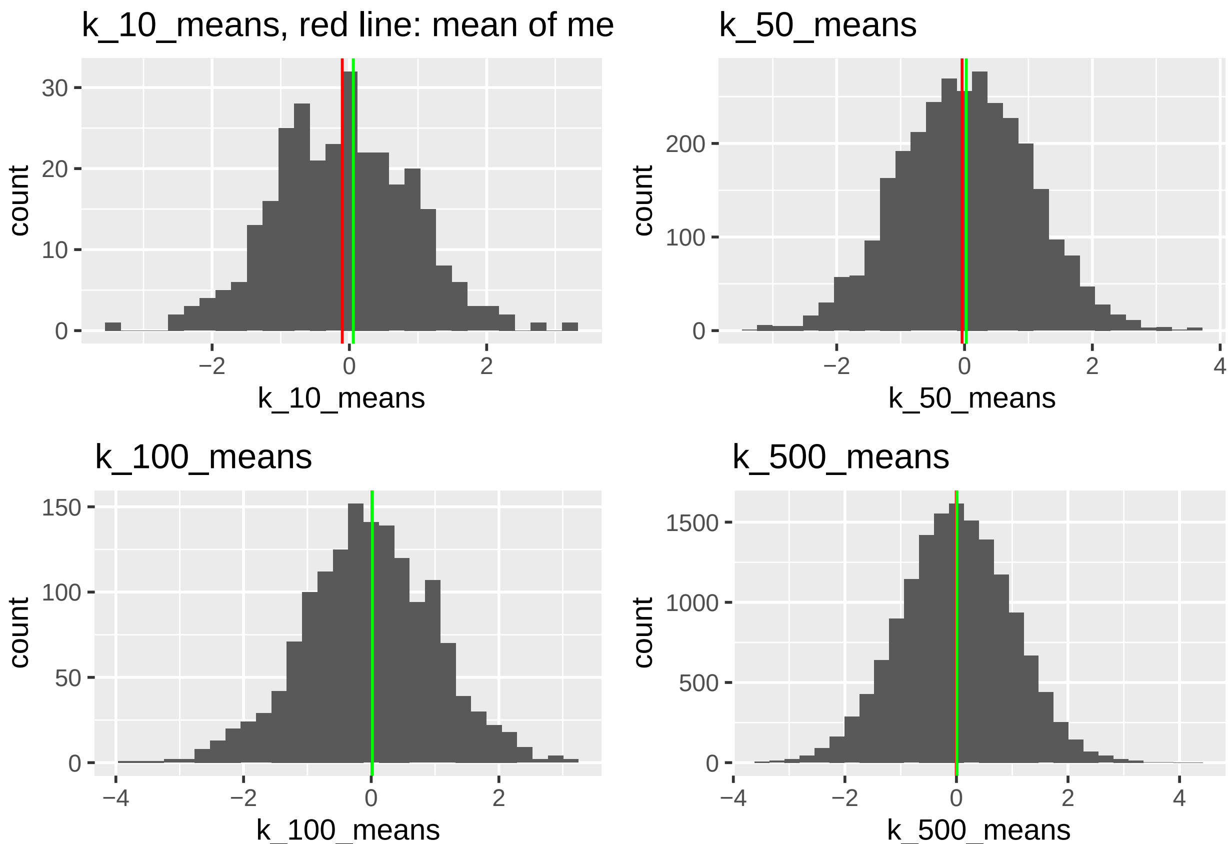
```r
k_sample_table <- data.frame(k_10_data, k_50_data, k_100_data, k_500_data)
k_sample_table
```

```
##    k_10_data   k_50_data   k_100_data    k_500_data
## 1 10.0000000 50.00000000 100.00000000 500.000000000
## 2 -0.1040083 -0.03852698   0.01361227  -0.004044936
## 3  1.0092384  1.02530994   1.02475080   0.995991151
## 4  0.0582684  0.02647339   0.01870930   0.008132234
```

Graph

```r
k_10_means <- unlist(lapply(k_10, mean))
k_100_means <- unlist(lapply(k_50, mean))
k_50_means <- unlist(lapply(k_100, mean))
k_500_means <- unlist(lapply(k_500, mean))

graph_k_10 <- ggplot() + aes(k_10_means) + geom_histogram() + geom_vline(xintercept = k_10_data[2], col

graph_k50 <-  ggplot() + aes(k_50_means) + geom_histogram() + geom_vline(xintercept = k_50_data[2], col

graph_k_100 <- ggplot() + aes(k_100_means) + geom_histogram() + geom_vline(xintercept = k_100_data[2], 

graph_k500 <-  ggplot() + aes(k_500_means) + geom_histogram() + geom_vline(xintercept = k_500_data[2], 

ggarrange(graph_k_10, graph_k50, graph_k_100, graph_k500 )
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



k_10_means, red line: mean of me
k_50_means
k_100_means
k_500_means
count
k_10_means
k_50_means
k_100_means
k_500_means