

QUANTILE CALCULATIONS IN R

Objective:

Showing how quantiles (esp. quartiles) are calculated in R.
R offers different functions to calculate quartiles, which can produce different output.

Examples:

```
> data <- c(1,12,14,3,96,111)

> summary(data)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.00   5.25   13.00   39.50   75.50  111.00

> quantile(data, c(0.25, 0.5, 0.75), type = 1)
25% 50% 75%
  3  12  96
```

Sources:

- <http://stat.ethz.ch/R-manual/R-devel/library/grDevices/html/boxplot.stats.html>
- <http://stat.ethz.ch/R-manual/R-patched/library/stats/html/quantile.html>
- <http://en.wikipedia.org/wiki/Quantile>

1. Defining test sets

Q1			MEDIAN				Q3				
3			3				3				
2	3	5	7	11	13	17	19	23	29	31	37

```
> data12 <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37)

> length(data12) / 4 ## Length of each quartile
[1] 3
```

Q1			MEDIAN				Q3			
2.75			2.75				2.75			
2	3	5	7	11	13	17	19	23	29	31

```
> data11 <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31)

> length(data11) / 4 ## Length of each quartile
[1] 2.75
```

Q1			MEDIAN				Q3		
2.5			2.5				2.5		
2	3	5	7	11	13	17	19	23	29

```
> data10 <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)

> length(data10) / 4 ## Length of each quartile
[1] 2.5
```

Q1			MEDIAN				Q3		
2.25			2.25				2.25		
2	3	5	7	11	13	17	19	23	

```
> data9 <- c(2, 3, 5, 7, 11, 13, 17, 19, 23)

> length(data9) / 4 ## Length of each quartile
[1] 2.25
```

Comparison of result sets for different functions

DATA	FUNCTION	Q1	MEDIAN	Q3
data12	quantile(data12, c(0.25, 0.5, 0.75), type = 1)	5	13	23
	quantile(data12, c(0.25, 0.5, 0.75), type = 2)	6	15	26
	quantile(data12, c(0.25, 0.5, 0.75), type = 3)	5	13	23
	quantile(data12, c(0.25, 0.5, 0.75), type = 4)	5	13	23
	quantile(data12, c(0.25, 0.5, 0.75), type = 5)	6	15	26
	quantile(data12, c(0.25, 0.5, 0.75), type = 6)	5.5	15	27.5
	quantile(data12, c(0.25, 0.5, 0.75), type = 7)	6.5	15	24.5
	quantile(data12, c(0.25, 0.5, 0.75), type = 8)	5.833333	15	26.5
	quantile(data12, c(0.25, 0.5, 0.75), type = 9)	5.875	15	26.375
	summary(data12)	6.5	15	24.5
	boxplot(data12)	6	15	26
data11	quantile(data11, c(0.25, 0.5, 0.75), type = 1)	5	13	23
	quantile(data11, c(0.25, 0.5, 0.75), type = 2)	5	13	23
	quantile(data11, c(0.25, 0.5, 0.75), type = 3)	5	13	19
	quantile(data11, c(0.25, 0.5, 0.75), type = 4)	4.5	12	20
	quantile(data11, c(0.25, 0.5, 0.75), type = 5)	5.5	13	22
	quantile(data11, c(0.25, 0.5, 0.75), type = 6)	5	13	23
	quantile(data11, c(0.25, 0.5, 0.75), type = 7)	6	13	21
	quantile(data11, c(0.25, 0.5, 0.75), type = 8)	5.333333	13	22.333333
	quantile(data11, c(0.25, 0.5, 0.75), type = 9)	5.375	13	22.25
	summary(data11)	6	13	21
	boxplot(data11)	6	13	21
data10	quantile(data10, c(0.25, 0.5, 0.75), type = 1)	5	11	19
	quantile(data10, c(0.25, 0.5, 0.75), type = 2)	5	12	19
	quantile(data10, c(0.25, 0.5, 0.75), type = 3)	3	11	19
	quantile(data10, c(0.25, 0.5, 0.75), type = 4)	4	11	18
	quantile(data10, c(0.25, 0.5, 0.75), type = 5)	5	12	19
	quantile(data10, c(0.25, 0.5, 0.75), type = 6)	4.5	12	20
	quantile(data10, c(0.25, 0.5, 0.75), type = 7)	5.5	12	18.5
	quantile(data10, c(0.25, 0.5, 0.75), type = 8)	4.833333	12	19.333333
	quantile(data10, c(0.25, 0.5, 0.75), type = 9)	4.875	12	19.25
	summary(data10)	5.5	12	18.5
	boxplot(data10)	5	12	19
data9	quantile(data9, c(0.25, 0.5, 0.75), type = 1)	5	11	17
	quantile(data9, c(0.25, 0.5, 0.75), type = 2)	5	11	17
	quantile(data9, c(0.25, 0.5, 0.75), type = 3)	3	7	17
	quantile(data9, c(0.25, 0.5, 0.75), type = 4)	3.5	9	16
	quantile(data9, c(0.25, 0.5, 0.75), type = 5)	4.5	11	17.5
	quantile(data9, c(0.25, 0.5, 0.75), type = 6)	4	11	18
	quantile(data9, c(0.25, 0.5, 0.75), type = 7)	5	11	17
	quantile(data9, c(0.25, 0.5, 0.75), type = 8)	4.333333	11	17.666667
	quantile(data9, c(0.25, 0.5, 0.75), type = 9)	4.375	11	17.625
	summary(data9)	5	11	17
	boxplot(data9)	5	11	17

Custom quantile functions per type

```
QuantileType1 <- function (v, p) {  
  v = sort(v)  
  m = 0  
  n = length(v)  
  j = floor((n * p) + m)  
  g = (n * p) + m - j  
  y = ifelse (g == 0, 0, 1)  
  ((1 - y) * v[j]) + (y * v[j+1])  
}
```

```
QuantileType2 <- function (v, p) {  
  v = sort(v)  
  m = 0  
  n = length(v)  
  j = floor((n * p) + m)  
  g = (n * p) + m - j  
  y = ifelse (g == 0, 0.5, 1)  
  ((1 - y) * v[j]) + (y * v[j+1])  
}
```

```
QuantileType3 <- function (v, p) {  
  v = sort(v)  
  m = -0.5  
  n = length(v)  
  j = floor((n * p) + m)  
  g = (n * p) + m - j  
  y = ifelse(trunc(j/2)*2==j, ifelse(g==0, 0, 1), 1)  
  ((1 - y) * v[j]) + (y * v[j+1])  
}
```

```
QuantileType7 <- function (v, p) {  
  v = sort(v)  
  h = ((length(v)-1)*p)+1  
  v[floor(h)]+((h-floor(h))*(v[floor(h)+1]- v[floor(h)]))  
}
```

Example:

```
> data12 <- c(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37)  
> QuantileType1(data12, 0.25)  
[1] 5
```

Boxplots

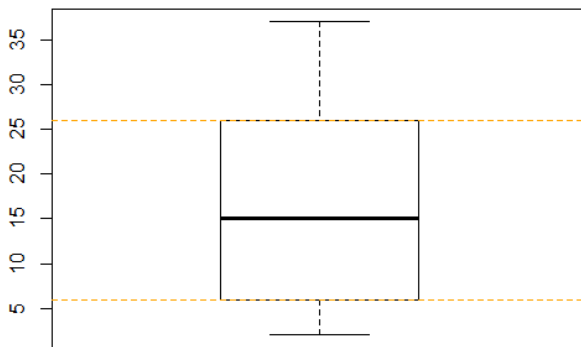
```
boxplot(data12, main="Boxplot (data12)")  
abline(h= 6, col="orange", lty=2, )  
abline(h=26, col="orange", lty=2)
```

```
boxplot(data11, main="Boxplot (data11)")  
abline(h= 6, col="orange", lty=2, )  
abline(h=21, col="orange", lty=2)
```

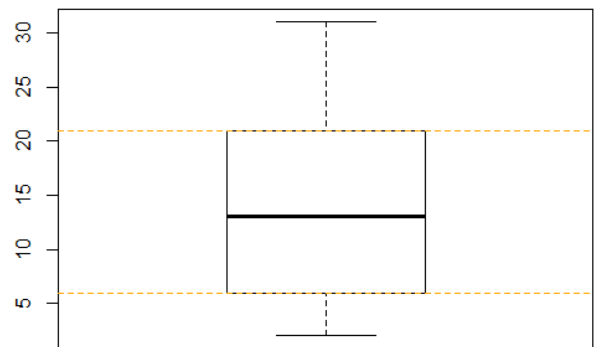
```
boxplot(data10, main="Boxplot (data10)")  
abline(h= 5, col="orange", lty=2, )  
abline(h=19, col="orange", lty=2)
```

```
boxplot(data9, main="Boxplot (data9)")  
abline(h= 5, col="orange", lty=2, )  
abline(h=17, col="orange", lty=2)
```

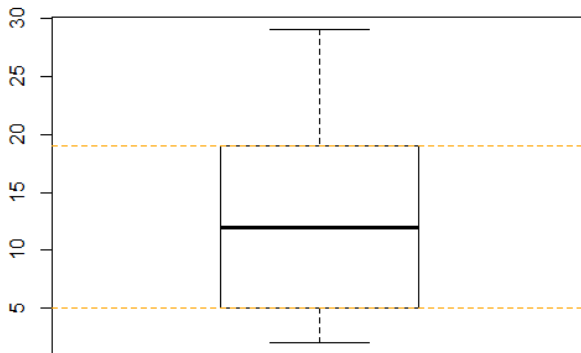
Boxplot (data12)



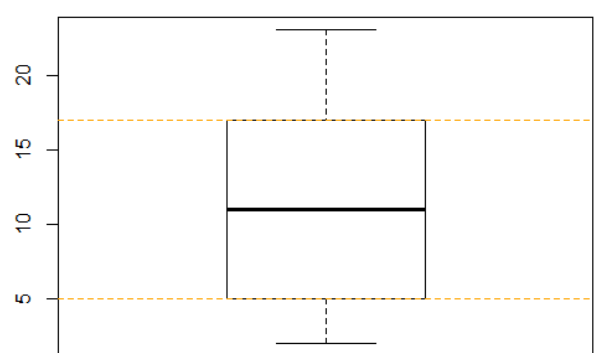
Boxplot (data11)



Boxplot (data10)



Boxplot (data9)



Conclusions so far...

The function `summary()` seems to use the same algorithm for calculating Q1, median and Q3 as does the function `quantiles()` with `type` set to 7.

Sometimes, the function `quantiles()` generates the same results with different `type` set.

The function `boxplot()` does not seem to use one of the 9 types that the function `quantiles()` uses to calculate Q1, median and Q3.

Box Plot Statistics

```
> ?boxplot.stats
```

The two 'hinges' are versions of the first and third quartile, i.e., close to $\text{quantile}(x, c(1,3)/4)$. The hinges equal the quartiles for odd n (where $n \leftarrow \text{length}(x)$) and differ for even n . Whereas the quartiles only equal observations for $n \% 4 == 1$ ($n = 1 \bmod 4$), the hinges do so additionally for $n \% 4 == 2$ ($n = 2 \bmod 4$), and are in the middle of two observations otherwise.

Based on this information, the two 'hinges' in `boxplot(data9)` should be equal to the calculation of the first and third quartile using function `quantile(data9, c(1,3)/4)`. The same should apply to `boxplot(data11)` and `quantile(data11, c(1,3)/4)`. This is confirmed by the data in the table on page 2.

In addition, quartiles only equal observations for $n \% 4 == 1$ and $n \% 4 == 2$, and are in the middle of two observations otherwise.

Based on this insights and the defined data sets `data9`, `data10`, `data11` and `data12` a custom function can be defined to calculate the first and third quartile according to 'boxplot logic'.

```
BoxplotQuartiles <- function (v) {
  v <- sort(v)
  n <- length(v)
  ql <- n/4 ## Length of a quartile
  p1 <- ql*1 ## Position Q1 in vector
  p3 <- ql*3 ## Position Q3 in vector
  f1 <- p1%%1 ## Fractional part of p1
  f3 <- p3%%1 ## Fractional part of p3
  if (f1 == 0.25)
    Q1 <- v[p1 + 0.75]
  else if (f1 == 0.50)
    Q1 <- v[p1 + 0.50]
  else if (f1 == 0.75)
    Q1 <- (v[p1 + 0.25] + v[p1 + 0.25 + 1.00]) / 2
  else if (f1 == 0.00)
    Q1 <- (v[p1 + 0.00] + v[p1 + 0.00 + 1.00]) / 2
  else
    Q1 <- 'Error in calculation Q1'
  if (f3 == 0.25)
    Q3 <- (v[p3 - 0.25] + v[p3 - 0.25 + 1.00]) / 2
  else if (f3 == 0.50)
    Q3 <- v[p3 + 0.50]
  else if (f3 == 0.75)
    Q3 <- v[p3 + 0.25]
  else if (f3 == 0.00)
    Q3 <- (v[p3 - 0.00] + v[p3 - 0.00 + 1.00]) / 2
  else
    Q3 <- 'Error in calculation Q3'
  return(c(Q1, Q3))
}
```

Test function with the defined data sets

```
> BoxplotQuartiles(data12)
[1] 6 26

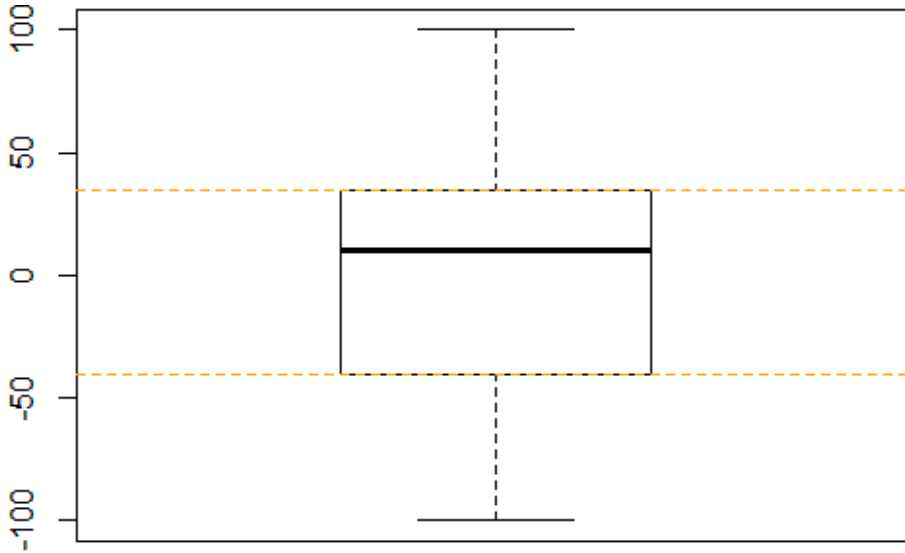
> BoxplotQuartiles(data11)
[1] 6 21

> BoxplotQuartiles(data10)
[1] 5 19

> BoxplotQuartiles(data9)
[1] 5 17
```

Test function with another odd data set

```
> data25 <- c(-100,99,99,100,96,10,-80,85,4,30,30,30,-40,-50,0,0,6,17,19,35,10,71,-99,-98,-97)
> BoxplotQuartiles(data25)
[1] -40 35
> boxplot(data25)
> abline(h=-40, col="orange", lty=2)
> abline(h= 35, col="orange", lty=2)
```



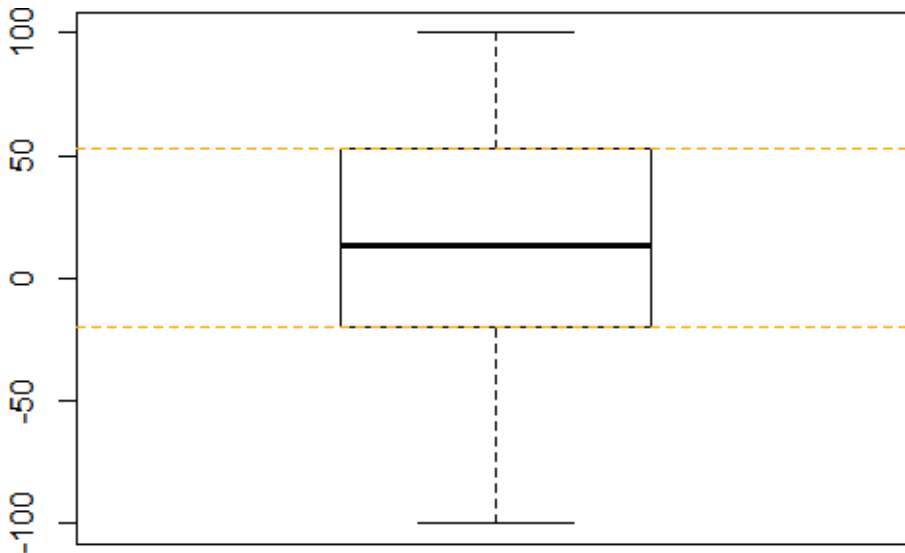
Expectation:

Quantile and summary functions give equal results for the first and third quartile compared to the custom function BoxplotQuartiles() because of the odd data set.

```
> quantile(data25, c(1,3)/4, type=7)
25% 75%
-40 35
> summary(data25)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-100.00 -40.00   10.00    7.08  35.00  100.00
```

Test function with another even data set

```
> data24 <- c(-100,99,99,100,96,10,-80,85,4,30,30,30,-40,-50,0,0,6,17,19,35,10,71,-99,-98)
> BoxplotQuartiles(data24)
[1] -20 53
> boxplot(data24)
> abline(h=-20, col="orange", lty=2)
> abline(h=53, col="orange", lty=2)
```



Expectation:

Quantile and summary functions give other results for the first and third quartile compared to the custom function BoxplotQuartiles() because of the even data set.

```
> quantile(data24, c(1,3)/4, type=7)
25% 75%
-10 44
> summary(data24)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-100.00 -10.00   13.50   11.42  44.00  100.00
```