Johnson SU distribution
MLE (Maximum Likelihood Estimate) fit to determine parameters
LR (Likelihood Ratio) approach to find tolerance limit

Pseudo-Code Used to Describe Process

Input

```
x     <- iris$Sepal.Width
sided <- 1
alpha <- 0.01  # confidence = 1 – alpha / sided
P     <- 0.99  # proportion or coverage
```

```
> x
  [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9 3.5 3.8 3.8
 [21] 3.4 3.7 3.6 3.3 3.4 3.0 3.4 3.5 3.4 3.2 3.1 3.4 4.1 4.2 3.1 3.2 3.5 3.6 3.0 3.4
 [41] 3.5 2.3 3.2 3.5 3.8 3.0 3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7
 [61] 2.0 3.0 2.2 2.9 2.9 3.1 3.0 2.7 2.2 2.5 3.2 2.8 2.5 2.8 2.9 3.0 2.8 3.0 2.9 2.6
 [81] 2.4 2.4 2.7 2.7 3.0 3.4 3.1 2.3 3.0 2.5 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8
[101] 3.3 2.7 3.0 2.9 3.0 3.0 2.5 2.9 2.5 3.6 3.2 2.7 3.0 2.5 2.8 3.2 3.0 3.8 2.6 2.2
[121] 3.2 2.8 2.8 2.7 3.3 3.2 2.8 3.0 2.8 3.0 2.8 3.8 2.8 2.8 2.6 3.0 3.4 3.1 3.0 3.1
[141] 3.1 3.1 2.7 3.2 3.3 3.0 2.5 3.0 3.4 3.0
```

Determine Johnson Su Parameters

```
## Define nll (negative log likelihood) function to fit
## parameters: gamma, delta, xi, lambda
nll <- function() {
    pdf <- delta /( lambda * sqrt(2 * pi)    ) *
        1 / sqrt(1 +                   ( (x-xi)/lambda )^2)   *
        exp( -0.5*(gamma + delta * asinh( (x-xi)/lambda ))^2 )
    nll <- -sum(log(pdf))
}

## MLE fit to minimize nnl() returns gamma, delta, xi, and lambda
```

```
                              gamma     delta        xi    lambda     quant
standard fit              -3.306484 5.319412 1.784619 1.887725        NA
```

Calculate Tolerance Limit

(1) Determine Equivalent Johnson SU Fit Using Quantile as a Parameter Instead of Gamma

```
## Define nll.q function to fit alternate parameters: quant, delta, xi, lambda
## where P     = coverage
##       quant = quantile associated with coverage
nll.q <- function() {
    gamma <- qnorm(P) - delta * asinh( (quant-xi)/lambda )
    pdf <- delta /( lambda * sqrt(2 * pi)    ) *
        1 / sqrt(1 +                   ( (x-xi)/lambda    )^2)   *
        exp( -0.5*(gamma + delta * asinh( (x-xi)/lambda ) )^2 )
    nll.q <- -sum(log(pdf))
}

## MLE fit to minimize nll.q() returns quant, delta, xi, and lambda for given P
```

```
                              gamma     delta        xi    lambda     quant
standard fit              -3.306484 5.319412 1.784619 1.887725        NA
fit on quantile at 1-P    -3.306500 5.319415 1.784613 1.887714 2.134413
fit on quantile at P      -3.307633 5.319502 1.784351 1.887396 4.178574
```

(2) Calculate Confidence Limits on Quantile Using LR (Likelihood Ratio)

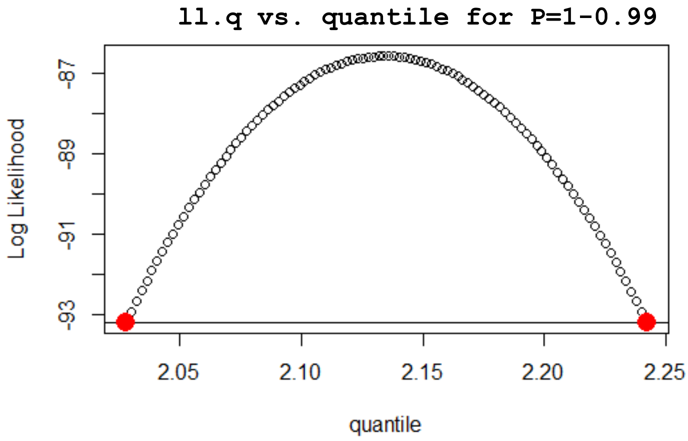
```
## Find peak (ll.max) of log likelihood function (-nll.q())
```

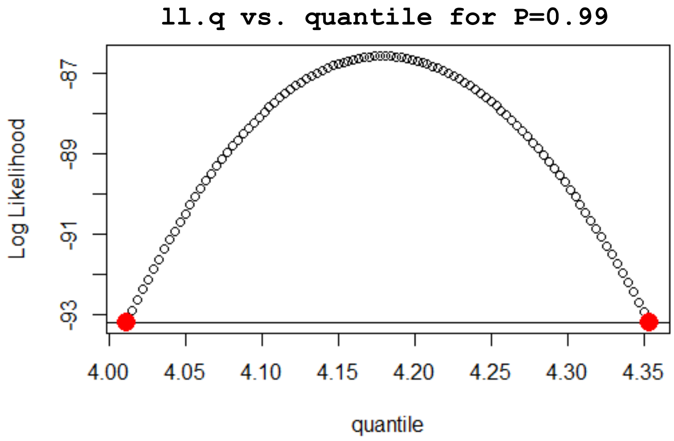
```
ll.max = -86.55915
```

```
## Reduce peak by chi-squared
ll.tol <-  ll.max - qchisq(1 - alpha/sided, 1)  # qchisq(1-0.01/1, 1) = 6.634897
```

```
ll.tol = -93.19405
```

```
## Confidence limits are the intersection of ll.tol and the log likelihood function
## for given level of coverage, P
ll.q <- -nll.q(x, P, quantile, delta, xi, lambda)
```





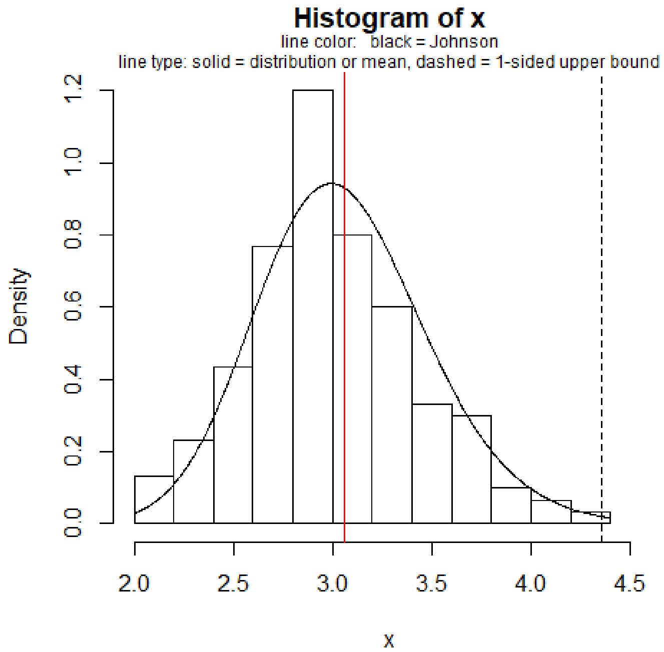

```
Final confidence interval for P= 0.01
2.027558 2.242365
```

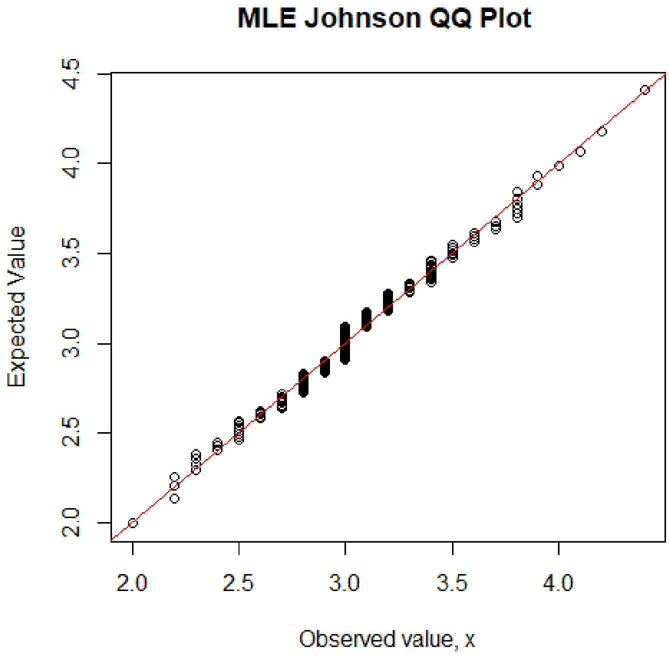
```
Final confidence interval for P= 0.99
4.011839 4.352955
```

```
## Based on the above:
```

```
  alpha    P sided tol.lower tol.upper
1  0.01 0.99     1  2.027558  4.352955
```

Histogram with Upper, 1-Sided, 99/99 Tolerance Limit and QQ Plot





Actual Coding Used to Recreate Above Output

```
## put common packages and functions defined in folder "modules" into namespace
## "modules" folder contains the two functions used below:
##    mle.johnsonsu() and hist_nwj()
source("setup.r")

## define input
x     <- iris$Sepal.Width

## use MLE to determine fit and LR to determine confidence limits
## using function defaults: alpha=0.01, P=0.99, and sided=1
out.fit <- mle.johnsonsu(x, plots=TRUE)

## create histogram with Johnson SU fit (type='j') and show upper tolerance limit
## using function defaults: alpha=0.01, P=0.99, and sided=1
out.hist <- hist_nwj(x, type='j')

## create QQ plot for Johnson SU fit
out.qq   <- qqplot_nwj(x, type='j')
```


Acknowledgement

The approach described in the pseudo-code section is based on the approach found here:
https://personal.psu.edu/abs12/stat504/Lecture/lec3_4up.pdf