

Package ‘ggQC’

December 1, 2018

Type Package

Title Quality Control Charts for 'ggplot'

Version 0.0.31

Author Kenith Grey

Maintainer Kenith Grey <kenithgrey@r-bar.net>

Description Plot single and faceted type quality control charts for 'ggplot'.

Depends R (>= 2.10)

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat, gridExtra, knitr, rmarkdown, reshape2, plyr

RoxygenNote 6.1.1

Imports ggplot2, stats, dplyr, tidyr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-12-01 05:20:27 UTC

R topics documented:

capability.summary	3
cBar_LCL	4
cBar_UCL	4
Cp	5
Cpk	6
DNS	6
LD	7
mR	7
mR_points	8

mR_UCL	8
npBar	9
npBar_LCL	10
npBar_UCL	10
pBar	11
pBar_LCL	12
pBar_UCL	13
Pp	13
Ppk	14
process_tolerance	14
QCrange	15
QC_Capability	16
QC_Lines	17
QC_Violations	20
rBar	22
rBar_LCL	23
rBar_UCL	23
rMedian	24
rMedian_LCL	25
rMedian_UCL	25
sBar	26
sBar_LCL	27
sBar_UCL	27
stat_mR	28
stat_pareto	30
stat_QC	32
stat_QC_CAPA	36
stat_QC_Capability	41
stat_QC_cap_hlabels	44
stat_QC_cap_hlines	49
stat_QC_cap_summary	54
stat_QC_cap_vlabels	59
stat_QC_cap_vlines	64
stat_QC_labels	69
stat_qc_violations	72
uBar	75
uBar_LCL	76
uBar_UCL	77
UD	78
xBar_Bar	78
xBar_one_LCL	79
xBar_one_UCL	80
xBar_rBar_LCL	80
xBar_rBar_UCL	81
xBar_rMedian_LCL	82
xBar_rMedian_UCL	83
xBar_sBar_LCL	84
xBar_sBar_UCL	85

xMedian_Bar	86
xMedian_rBar_LCL	86
xMedian_rBar_UCL	87
xMedian_rMedian_LCL	88
xMedian_rMedian_UCL	89

Index **90**

capability.summary *Calculate Summary of Quality Performance Parameters*

Description

function to report listing of quality performance parameters

- **Proc. Tolerance (sigma)**: Describes the number of your process sigma (from QC charting) that can fit in your customer’s specification window (the larger the better).
- **DNS (sigma)**: Distance to Nearest Specification (DNS) limit. Measure of how centered your process is and how close you are to the nearest process limit in sigma units.
- **Cp**: Describes how many times your 6 sigma process window (from QC charting) can fit in your customer’s specification window (the larger the better)
- **Cpk**: Describes how centered your process is relative to customer specifications. How many times can you fit a 3 sigma window (from QC charting) between your process center and the nearest customer specification limit.
- **Pp**: Describes how many times your 6 sigma process window (overall standard deviation) can fit in your customer’s specification window (the larger the better)
- **Ppk**: Describes how centered your process is relative to customer specifications. How many times can you fit a 3 sigma window (overall standard deviation) between your process center and the nearest customer specification limit.

Usage

capability.summary(LSL, USL, QC.Center, QC.Sigma, s.Sigma, digits = 2)

Arguments

LSL	number, customer’s lower specification limit.
USL	number, customer’s upper specification limit.
QC.Center	number, the mean or median value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
s.Sigma	number, the sigma value determined from overall standard deviation (i.e., sd()).
digits	integer, how many digits to report.

Value

data frame , listing of metric labels and value

`cBar_LCL`*Lower Control Limit: Count Data (c-chart)*

Description

Calculates lower control limit (LCL) for count data acquired over the same-sized area of opportunity. Negative values are reported as 0.

Usage

```
cBar_LCL(y, na.rm = FALSE, ...)
```

Arguments

<code>y</code>	Vector of count data. Each observation having the same-area of opportunity.
<code>na.rm</code>	a logical value indicating whether NA values should be stripped before the computation proceeds.
<code>...</code>	further arguments passed to or from other methods.

Value

A number; 3-sigma lower control limit (LCL). Function returns 0 for negative values.

Examples

```
set.seed(5555)
y <- rpois(30, 9)
cBar_LCL(y)
```

`cBar_UCL`*Upper Control Limit: Count Data (c-chart)*

Description

Calculates upper control limit (UCL) for count data acquired over the same-sized area of opportunity.

Usage

```
cBar_UCL(y, na.rm = FALSE, ...)
```

Arguments

y	Vector of count data. Each observation having the same-area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; 3-sigma upper control limit (UCL)

Examples

```
set.seed(5555)
y <- rpois(30, 9)
cBar_UCL(y)
```

Cp

Calculate Cp

Description

function to calculate Cp - "The elbowroom or margin your process"

Usage

```
Cp(LSL, USL, QC.Sigma)
```

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.

Value

numeric, Cp value (unitless)

 Cpk

Calculate Cpk

Description

function to calculate Cpk - "measure of process centering"

Usage

Cpk(LSL, USL, QC.Center, QC.Sigma)

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Center	number, the mean or median value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.

Value

numeric, Cpk value (unitless)

DNS

Calculate Distance to Nearest Specification Limit

Description

function to calculate a standardized distance to the nearest specification limit (sigma units)

Usage

DNS(LSL, USL, QC.Center, QC.Sigma)

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Center	number, the mean or median value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.

Value

numeric, standardized distance to the nearest specification limit (sigma units)

LD *Calculate Distance to Lower Specification Limit*

Description

function to calculate a standardized distance to the Lower specification limit (sigma units)

Usage

LD(LSL, USL, QC.Center, QC.Sigma)

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Center	number, the mean or median value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.

Value

numeric, standardized distance to the lower specification limit (sigma units)

mR *Mean One-Point Moving Range*

Description

Calculates the mean one-point moving range used when constructing a moving-range chart.

Usage

mR(y, na.rm = TRUE, ...)

Arguments

y	Vector of values
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; mean one-point moving range.

Examples

```
set.seed(5555)
values <- rnorm(n = 100, mean = 25, sd = 1)
mR(values)
```

mR_points	<i>One Point Moving Range of Vector</i>
-----------	---

Description

Calculates a one-point moving range vector given an input vector of values. Output often used to produce mR-chart.

Usage

```
mR_points(y)
```

Arguments

`y` : vector of values

Value

Vector of one-point moving range.

Examples

```
y <- seq(-5:5)
mR_points(y)
```

mR_UCL	<i>Mean One-Point Moving Range Upper Control Limit (UCL)</i>
--------	--

Description

Calculates the mean one-point moving range UCL used when constructing a moving-range chart.

Usage

```
mR_UCL(y, na.rm = FALSE, ...)
```

Arguments

`y` Vector of values

`na.rm` a logical value indicating whether NA values should be stripped before the computation proceeds.

`...` further arguments passed to or from other methods.

Value

A number; mean one-point moving range UCL.

Examples

```
set.seed(5555)
values <- rnorm(n = 100, mean = 25, sd = 1)
mR_UCL(values)
```

 npBar

Mean Value: Binomial Data (np-chart)

Description

Calculates the mean value for binomial count data acquired over the same-sized area of opportunity.

Usage

```
npBar(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of binomial count data (not proportions). Each observation having the same-area of opportunity.
n	A number representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; mean value

Examples

```
set.seed(5555)
p <- rbinom(n = 100, size = 30, prob = .2)
npBar(y = p, n = 30)
```

npBar_LCL	<i>Lower Control Limit: Binomial Data (np-chart)</i>
-----------	--

Description

Calculates lower control limit (LCL) for binomial count data acquired over the same-sized area of opportunity.

Usage

```
npBar_LCL(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of binomial count data (not proportions). Each observation having the same-area of opportunity.
n	A number representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; 3-sigma upper control limit (LCL)

Examples

```
set.seed(5555)
p <- rbinom(n = 100, size = 30, prob = .2)
npBar_LCL(y = p, n = 30)
```

npBar_UCL	<i>Upper Control Limit: Binomial Data (np-chart)</i>
-----------	--

Description

Calculates upper control limit (UCL) for binomial count data acquired over the same-sized area of opportunity.

Usage

```
npBar_UCL(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of binomial count data (not proportions). Each observation having the same-area of opportunity.
n	A number representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; 3-sigma upper control limit (UCL)

Examples

```
set.seed(5555)
p <- rbinom(n = 100, size = 30, prob = .2)
npBar_UCL(y = p, n = 30)
```

pBar

Mean Proportion: Binomial Data (p-chart)

Description

Calculates overall mean proportion for binomial proportion data acquired over a variable area of opportunity.

Usage

```
pBar(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of binomial proportion data (not counts). Observations may have a different area of opportunity, n.
n	A vector representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A vector of mean proportion, length equal to length of parameter y.

Examples

```
set.seed(5555)
p <- rbinom(n = 100, size = 30, prob = .2)
n <- rpois(100, 100)
pBar(y = p/n, n = n)
```

pBar_LCL

Lower Control Limit: Binomial Data (p-chart)

Description

Calculates point-wise lower control limit (LCL) for binomial proportion data acquired over a variable area of opportunity.

Usage

```
pBar_LCL(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of binomial proportion data (not counts). Observations may have a different area of opportunity, n.
n	A vector representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A vector; point-wise 3-sigma lower control limit (LCL)

Examples

```
set.seed(5555)
p <- rbinom(n = 100, size = 30, prob = .2)
n <- rpois(100, 100)
pBar_LCL(y = p/n, n = n)
```

pBar_UCL *Upper Control Limit: Binomial Data (p-chart)*

Description

Calculates point-wise upper control limit (UCL) for binomial proportion data acquired over a variable area of opportunity.

Usage

```
pBar_UCL(y, n, na.rm = FALSE, ...)
```

Arguments

y Vector of binomial proportion data (not counts). Observations may have a different area of opportunity, n.

n A vector representing the area of opportunity.

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

... further arguments passed to or from other methods.

Value

A vector; point-wise 3-sigma upper control limit (UCL)

Examples

```
set.seed(5555)
p <- rbinom(n = 100, size = 30, prob = .2)
n <- rpois(100, 100)
pBar_UCL(y = p/n, n = n)
```

Pp *Calculate Pp*

Description

function to calculate Pp - "The elbowroom or margin your process" uses overall sigma value not the QC chart sigma values.

Usage

```
Pp(LSL, USL, s.Sigma)
```

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
s.Sigma	number, the sigma value determined from overall standard deviation (i.e., sd()).

Value

numeric, Pp value (unitless)

Ppk	<i>Calculate Cpk</i>
-----	----------------------

Description

function to calculate Cpk - "measure of process centering"

Usage

Ppk(LSL, USL, QC.Center, s.Sigma)

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Center	number, the mean or median value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
s.Sigma	number, the sigma value determined from overall standard deviation (i.e., sd()).

Value

numeric, Ppk value (unitless)

process_tolerance	<i>Calculate QC Process Tolerance</i>
-------------------	---------------------------------------

Description

function to calculate a standardized process tolerance with sigma unit

Usage

process_tolerance(LSL, USL, QC.Sigma)

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.

Value

numeric, standardized process tolerance value in sigma units

QCrange *Range: Max Min Difference*

Description

Given a set of numbers, function calculates the difference between the maximum and minimum value.

Usage

```
QCrange(y)
```

Arguments

y : vector of values

Value

a number.

Examples

```
y <- seq(-5:5)
QCrange(y)
```

QC_Capability

*Calculate Summary of Quality Performance Parameters***Description**

function to report listing of quality performance parameters

Usage

```
QC_Capability(data = NULL, value = NULL, grouping = NULL,
  formula = NULL, method = "xBar.rBar", na.rm = FALSE, LSL = NULL,
  USL = NULL, digits = 2)
```

Arguments

data	vector or dataframe, as indicated below for each chart type <ul style="list-style-type: none"> • Individuals (XmR): vector of values; • Studentized: dataframe
value	string, Studentized Charts , name of numeric vector in dataframe with values of interest.
grouping	string, Studentized Charts , name of single factor/variable to split the dataframe "values" by
formula	Studentized Charts : a formula, such as $y \sim x1 + x2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
method	string, calling one of the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit <ul style="list-style-type: none"> • Proc. Tolerance (sigma): Describes the number of your process sigma (from QC charting) that can fit in your customer's specification window (the larger the better). • DNS (sigma): Distance to Nearest Specification (DNS) limit. Measure of how centered your process is and how close you are to the nearest process limit in sigma units. • Cp: Describes how many times your 6 sigma process window (from QC charting) can fit in your customer's specification window (the larger the better) • Cpk: Describes how centered your process is relative to customer specifications. How many times can you fit a 3 sigma window (from QC charting) between your process center and the nearest customer specification limit.

- **Pp**: Describes how many times your 6 sigma process window (overall standard deviation) can fit in your customer's specification window (the larger the better)
- **Ppk**: Describes how centered your process is relative to customer specifications. How many times can you fit a 3 sigma window (overall standard deviation) between your process center and the nearest customer specification limit.

digits integer, how many digits to report.

Value

data frame , listing of metric labels and value

QC_Lines *Calculate QC Limits*

Description

Calculates QC chart lines for the following chart types and reports in a dataframe:

- **Individuals Charts**: mR, XmR,
- **Attribute Charts**: c, np, p, u,
- **Studentized Charts**: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian,
- **Dispersion Charts**: rBar, rMedian, sBar.

Usage

```
QC_Lines(data = NULL, value = NULL, grouping = NULL,
          formula = NULL, n = NULL, method = "xBar.rBar", na.rm = FALSE)
```

Arguments

data	vector or dataframe, as indicated below for each chart type <ul style="list-style-type: none"> • Individuals & Attribute Charts: vector of values; • Studentized & Dispersion Charts: dataframe
value	string, Studentized Charts and Dispersion Charts , numeric vector in dataframe with values of interest
grouping	string, Studentized Charts and Dispersion Charts : single factor/variable to split the dataframe "values" by
formula	Studentized Charts and Dispersion Charts : a formula, such as $y \sim x1 + x2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
n	number or vector as indicated below for each chart type. <ul style="list-style-type: none"> • Individuals Charts: No effect

	<ul style="list-style-type: none"> • Attribute Charts: (p and u) vector, indicating sample area of opportunity. • Attribute Charts: (np) number, indicating constant sampling area of opportunity. • Studentized Charts: number, user specified subgroup size. • Dispersion Charts: No effect
method	<p>string, calling the following methods:</p> <ul style="list-style-type: none"> • Individuals Charts: mR, XmR, • Attribute Charts: c, np, p, u, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian • Dispersion Charts: rBar, rMedian, sBar.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

Value

a dataframe,

- **Attribute Data:** (p and u) Center Line, Upper Control Limit and Lower Control limit for each point.
- **Other Data:** single line dataframe, with relevant control limits noted in column headings.

Note

If using the **formula** argument do not use **value** and **group** arguments.

References

Wheeler, DJ, and DS Chambers. Understanding Statistical Process Control, 2nd Ed. Knoxville, TN: SPC, 1992. Print.

Examples

```
#####
# Example 1: Charts other than "p" or "u" #
#####

# Load Libraries -----
require(ggQC)
require(plyr)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(processID = as.factor(rep(1,100)),
                      metric_value = rnorm(100,0,1),
                      subgroup_sample=rep(1:20, each=5),
                      Process_run_id = 1:100)

set.seed(5555)
```

```

Process2 <- data.frame(processID = as.factor(rep(2,100)),
                      metric_value = rnorm(100,5, 1),
                      subgroup_sample=rep(1:10, each=10),
                      Process_run_id = 101:200)

Both_Processes <- rbind(Process1, Process2)

# QC Values For Individuals -----
# All Together
QC_Lines(data = Both_Processes$metric_value, method = "XmR")

# For Each Process
ddply(Both_Processes, .variables = "processID",
      .fun =function(df){
        QC_Lines(data = df$metric_value, method = "XmR")
      }
    )

# QC Values For Studentized Runs-----
# All Together
QC_Lines(data = Both_Processes,
          formula = metric_value ~ subgroup_sample)

# For Each Process
ddply(Both_Processes, .variables = "processID",
      .fun =function(df){
        QC_Lines(data = df, formula = metric_value ~ subgroup_sample)
      }
    )

#####
# Example 2 "p" data #
#####

# Setup p Data -----
set.seed(5555)
bin_data <- data.frame(
  trial = 1:30,
  Num_Incomplete_Items = rpois(n = 30, lambda = 30),
  Num_Items_in_Set = runif(n = 30, min = 50, max = 100))

bin_data$Proportion_Incomplete <- bin_data$Num_Incomplete_Items/bin_data$Num_Items_in_Set

# QC_Lines for "p" data -----
QC_Lines(data = bin_data$Proportion_Incomplete,
          n = bin_data$Num_Items_in_Set, method="p")

#####
# Example 3 "u" data #

```

```
#####

# Setup u Data -----
set.seed(5555)
bin_data <- data.frame(
  trial=1:30,
  Num_of_Blemishes = rpois(n = 30, lambda = 30),
  Num_Items_Inspected = runif(n = 30, min = 50, max = 100))

bin_data$Blemish_Rate <- bin_data$Num_of_Blemishes/bin_data$Num_Items_Inspected

# QC Lines for "u" data -----
QC_Lines(data = bin_data$Blemish_Rate,
  n = bin_data$Num_Items_Inspected, method="u")
```

QC_Violations

Calculate QC Violations

Description

function that calculates QC violations on sequentially ordered data based on the following 4 rules:

- **Violation Same Side:** 8 or more consecutive, same-side points
- **Violation 1 Sigma:** 4 or more consecutive, same-side points exceeding 1 sigma
- **Violation 2 Sigma:** 2 or more consecutive, same-side points exceeding 2 sigma
- **Violation 3 Sigma:** any points exceeding 3 sigma

Usage

```
QC_Violations(data, value = NULL, grouping = NULL, formula = NULL,
  method = NULL, ...)
```

Arguments

data	vector or dataframe, as indicated below for each chart type <ul style="list-style-type: none"> • Individuals: vector of values; • Studentized Charts: dataframe
value	Studentized Charts: numeric vector in dataframe with values of interest
grouping	Studentized Charts: single factor/variable to split the dataframe "values" by
formula	Studentized Charts: a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
...	further arguments passed to or from other methods.


```

data.frame(
  sub_group = rep(1:42),                #Define Subgroups
  sub_class = letters[X],
  c(
    c(runif(n = 5, min = 2.0,3.2)),      #Outlier Data
    sample(c(rnorm(30),5,-4), 32, replace = FALSE), #Normal Data
    c(runif(n = 5, min = -3.2, max = -2.0)) #Outlier Data
  )
)
}
)
)

colnames(QC_xBar.rBar) <- c("sub_group","sub_class", "value")
QC_Vs <- QC_Violations(data = QC_xBar.rBar,
  formula = value~sub_group,
  method = "xBar.rBar")

```

rBar

Mean Subgroup Range

Description

Calculates the mean subgroup range used when constructing a XbarR chart.

Usage

```
rBar(data, value, grouping, formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean subgroup range.

Examples

```

set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
rBar(data = df, formula = v~g)

```

rBar_LCL	<i>Mean Subgroup Range Lower Control Limit (LCL)</i>
----------	--

Description

Calculates the mean subgroup range Lower control limit (UCL) used when constructing a XbarR chart.

Usage

```
rBar_LCL(data = data, value = value, grouping = grouping,
         formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x1 + x2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean subgroup range lower control limit (LCL).

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
rBar_LCL(data = df, formula = v~g)
```

rBar_UCL	<i>Mean Subgroup Range Upper Control Limit (UCL)</i>
----------	--

Description

Calculates the mean subgroup range upper control limit (UCL) used when constructing a XbarR chart.

Usage

```
rBar_UCL(data = data, value = value, grouping = grouping,
         formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean subgroup range upper control limit (UCL).

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
rBar_UCL(data = df, formula = v~g)
```

rMedian

Median of Subgroup Ranges

Description

Calculates the median of subgroup ranges, used when constructing xBar_rMedian charts.

Usage

```
rMedian(data, value, grouping, formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; median subgroup range.

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
rMedian(data = df, formula = v~g)
```

rMedian_LCL	<i>Median of Subgroup Ranges Lower Control Limit (LCL)</i>
-------------	--

Description

Calculates the median of subgroup range Lower control limit (LCL) used when constructing a xBar_rMedian chart.

Usage

```
rMedian_LCL(data = data, value = value, grouping = grouping,
            formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x1 + x2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; median of subgroup range lower control limit (LCL).

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
rMedian_LCL(data = df, formula = v~g)
```

rMedian_UCL	<i>Median of Subgroup Ranges Upper Control Limit (UCL)</i>
-------------	--

Description

Calculates the median of subgroup range upper control limit (UCL) used when constructing a xBar_rMedian chart.

Usage

```
rMedian_UCL(data = data, value = value, grouping = grouping,
            formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; median of subgroup range upper control limit (UCL).

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
rMedian_UCL(data = df, formula = v~g)
```

sBar

Mean Subgroup Standard Deviation

Description

Calculates the mean subgroup standard deviation used when constructing a XbarS chart.

Usage

```
sBar(data, value, grouping, formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean subgroup standard deviation.

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
sBar(data = df, formula = v~g)
```

sBar_LCL	<i>Mean Subgroup Standard Deviation Lower Control Limit (LCL)</i>
----------	---

Description

Calculates the mean subgroup standard deviation Lower control limit (UCL) used when constructing a XbarS chart.

Usage

```
sBar_LCL(data = data, value = value, grouping = grouping,
         formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x1 + x2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean subgroup standard deviation lower control limit (LCL).

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
sBar_LCL(data = df, formula = v~g)
```

sBar_UCL	<i>Mean Subgroup Standard Deviation Upper Control Limit (UCL)</i>
----------	---

Description

Calculates the mean subgroup standard deviation upper control limit (UCL) used when constructing a XbarS chart.

Usage

```
sBar_UCL(data = data, value = value, grouping = grouping,
         formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean subgroup standard deviation upper control limit (UCL).

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
sBar_UCL(data = df, formula = v~g)
```

stat_mR

Generate mR chart in ggplot

Description

ggplot stat used to create a mR chart in ggplot

Usage

```
stat_mR(mapping = NULL, data = NULL, geom = "point",
        position = "identity", show.legend = NA, inherit.aes = TRUE,
        na.rm = FALSE, color.mr_point = "black", color.mr_line = "black",
        color.qc_limits = "red", color.qc_center = "blue", ...)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to ggplot() . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.

geom	The geometric object to use display the data
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
color.mr_point	color, to be used for the mR points.
color.mr_line	color, to be used for line connecting points.
color.qc_limits	color, used to colorize the plot's upper and lower mR control limits.
color.qc_center	color, used to colorize the plot's center line.
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.

Value

data need to produce the mR plot in ggplot.

Examples

```
#####
# Example 1: mR Chart #
#####

# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(processID = as.factor(rep(1,100)),
                      metric_value = rnorm(100,0,1),
                      subgroup_sample=rep(1:20, each=5),
                      Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(processID = as.factor(rep(2,100)),
                      metric_value = rnorm(100,5, 1),
                      subgroup_sample=rep(1:10, each=10),
                      Process_run_id = 101:200)

Both_Processes <- rbind(Process1, Process2)
```

```
# One Plot Both Processes -----
ggplot(Both_Processes, aes(x=Process_run_id, y = metric_value)) +
  stat_mR() + ylab("Moving Range")

# Facet Plot - Both Processes -----
ggplot(Both_Processes, aes(x=Process_run_id, y = metric_value)) +
  stat_mR() + ylab("Moving Range") +
  facet_grid(.~processID, scales = "free_x")
```

stat_pareto

Generate a Pareto Plot with ggplot

Description

stat function to create ggplot Pareto chart

Usage

```
stat_pareto(mapping = NULL, data = NULL, geom = "point",
  position = "identity", show.legend = NA, inherit.aes = TRUE,
  group = 1, na.rm = FALSE, point.color = "black", point.size = 2,
  line.color = "black", line.size = 0.5, bars.fill = c("red",
  "white"), ...)
```

Arguments

mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
geom	The geometric object to use display the data
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .

group	defines grouping for variable for pareto plot, default and suggested is 1.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
point.color	color, used to define point color of cumulative percentage line
point.size	number, used to define point size of cumulative percentage line
line.color	color, used to define line color of cumulative percentage line
line.size	color, used to define line weight of cumulative percentage line
bars.fill	character vector length 2, start and end colors for pareto bars.
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .

Value

Pareto plot.

Examples

```
#####
# Example 1: Pareto Plot #
#####

# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
df <- data.frame(
  x = letters[1:10],
  y = as.integer(runif(n = 10, min = 0, max=100))
)

# Render Pareto Plot -----

ggplot(df, aes(x=x, y=y)) +
  stat_pareto(point.color = "red",
             point.size = 3,
             line.color = "black",
             #size.line = 1,
             bars.fill = c("blue", "orange"),
             )
```

stat_QC

*Produce QC Charts with ggplot Framework.***Description**

Produce QC charts with ggplot framework. Support for faceting and layering of multiple QC chart lines on a single plot. Charts supported (see method argument for call):

- **Individuals Charts:** mR, XmR,
- **Attribute Charts:** c, np, p, u,
- **Studentized Charts:** xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian,
- **Dispersion Charts:** rBar, rMedian, sBar.

To label chart lines see [stat_QC_labels](#)

Usage

```
stat_QC(mapping = NULL, data = NULL, geom = "hline",
        position = "identity", na.rm = FALSE, show.legend = NA,
        inherit.aes = TRUE, n = NULL, method = "xBar.rBar",
        color.qc_limits = "red", color.qc_center = "blue",
        color.point = "black", color.line = "black",
        physical.limits = c(NA, NA), auto.label = FALSE,
        limit.txt.label = c("LCL", "UCL"), label.digits = 1,
        show.ln2.sigma = FALSE, ...)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to ggplot() . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
geom	The geometric object to use display the data
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
n	number, for <ul style="list-style-type: none"> • Studentized Charts, used for custom or hypothetical subgroup size. • np Charts, used to specify a fixed area of opportunity.
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: mR, XmR, • Attribute Charts: c, np, p, u, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian • Dispersion Charts: rBar, rMedian, sBar.
color.qc_limits	color, used to colorize the plot's upper and lower control limits.
color.qc_center	color, used to colorize the plot's center line.
color.point	color, used to colorize points in studentized plots. You will need <code>geom_point()</code> for C, P, U, NP, and XmR charts.
color.line	color, used to colorize lines connecting points in studentized plots. You will need <code>geom_line()</code> for C, P, U, NP, and XmR charts.
physical.limits	vector, specify lower physical boundary and upper physical boundary
auto.label	boolean setting, if T labels graph with control limits.
limit.txt.label	vector, provides option for naming or not showing the limit text labels (e.g., UCL, LCL) <ul style="list-style-type: none"> • limit.txt.label = c("LCL", "UCL"): default • limit.txt.label = c("Low", "High"): changes the label text to low and high • limit.txt.label = NA: does not show label text.
label.digits	integer, number of decimal places to display.
show.1n2.sigma	boolean setting, if T labels graph 1 and 2 sigma lines. Line color is set by <code>color.qc_limits</code>
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .

Value

ggplot control charts.

Examples

```

# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(processID = as.factor(rep(1,100)),
                       metric_value = rnorm(100,0,1),
                       subgroup_sample = rep(1:20, each=5),
                       Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(processID = as.factor(rep(2,100)),
                       metric_value = rnorm(100,5, 1),
                       subgroup_sample = rep(1:10, each=10),
                       Process_run_id = 101:200)

Both_Processes <- rbind(Process1, Process2)

#####
# Example 1: XmR Chart #
#####

EX1.1 <- ggplot(Both_Processes, aes(x=Process_run_id, y = metric_value)) +
  geom_point() + geom_line() + stat_QC(method="XmR") +
  stat_QC_labels(method="XmR", digits = 2) +
  facet_grid(.~processID, scales = "free_x")
#EX1.1

EX1.2 <- ggplot(Both_Processes, aes(x=Process_run_id, y = metric_value)) +
  stat_mR() + ylab("Moving Range") +
  stat_QC_labels(method="mR", digits = 2) +
  facet_grid(.~processID, scales = "free_x")
#EX1.2

#####
# Example 2: XbarR Chart #
#####

EX2.1 <- ggplot(Both_Processes, aes(x = subgroup_sample,
                                   y = metric_value,
                                   group = processID)) +
  stat_summary(fun.y = "mean", color = "blue", geom = c("point")) +
  stat_summary(fun.y = "mean", color = "blue", geom = c("line")) +
  stat_QC(method = "xBar.rBar") + facet_grid(.~processID, scales = "free_x")
#EX2.1

EX2.2 <- ggplot(Both_Processes, aes(x = subgroup_sample,
                                   y = metric_value,
                                   group = processID)) +
  stat_summary(fun.y = "QCrage", color = "blue", geom = "point") +

```

```

stat_summary(fun.y = "QCrange", color = "blue", geom = "line") +
stat_QC(method = "rBar") +
ylab("Range") +
facet_grid(.~processID, scales = "free_x")
#EX2.2

#####
# Example 3: p Chart      #
#####
# p chart Setup -----
set.seed(5556)
bin_data <- data.frame(
  trial=1:30,
  Num_Incomplete_Items = rpois(30, lambda = 30),
  Num_Items_in_Set = runif(n = 30, min = 50, max = 100))
bin_data$Proportion_Incomplete <- bin_data$Num_Incomplete_Items/bin_data$Num_Items_in_Set

# Plot p chart -----
EX3.1 <- ggplot(data = bin_data, aes(x=trial,
                                     y=Proportion_Incomplete,
                                     n=Num_Items_in_Set)) +
  geom_point() + geom_line() +
  stat_QC(method = "p")
#EX3.1

#####
# Example 4: u Chart      #
#####
# u chart Setup -----
set.seed(5555)
bin_data <- data.frame(
  trial=1:30,
  Num_of_Blemishes = rpois(30, lambda = 30),
  Num_Items_Inspected = runif(n = 30, min = 50, max = 100)
)
bin_data$Blemish_Rate <- bin_data$Num_of_Blemishes/bin_data$Num_Items_Inspected

# Plot u chart -----
EX4.1 <- ggplot(data = bin_data, aes(x=trial,
                                     y=Blemish_Rate,
                                     n=Num_Items_Inspected)) +
  geom_point() + geom_line() +
  stat_QC(method = "u")
#EX4.1

#####
# Example 5: np Chart     #
#####
# np chart Setup -----
set.seed(5555)
bin_data <- data.frame(
  trial=1:30,
  NumNonConforming = rbinom(30, 30, prob = .50))

```

```

Units_Tested_Per_Batch <- 60

# Plot np chart -----
EX5.1 <- ggplot(data = bin_data, aes(trial, NumNonConforming)) +
  geom_point() +
  stat_QC(method = "np", n = Units_Tested_Per_Batch)
#EX5.1

#####
# Example 6: c Chart #
#####
# c chart Setup -----
set.seed(5555)
Process1 <- data.frame(Process_run_id = 1:30,
                       Counts=rpois(n = 30, lambda = 25),
                       Group = "A")
Process2 <- data.frame(Process_run_id = 1:30,
                       Counts = rpois(n = 30, lambda = 5),
                       Group = "B")

all_processes <- rbind(Process1, Process2)
# Plot C Chart -----

EX6.1 <- ggplot(all_processes, aes(x=Process_run_id, y = Counts)) +
  geom_point() + geom_line() +
  stat_QC(method = "c", auto.label = TRUE, label.digits = 2) +
  scale_x_continuous(expand = expand_scale(mult = .25)) +
  facet_grid(.~Group)
# EX6.1

```

stat_QC_CAPA

Generic Function for drawing QC capability information on plots

Description

Generic Function for drawing QC capability information on plots

Usage

```

stat_QC_CAPA(LSL, USL, method = "xBar.rBar", digits = 1,
             mapping = NULL, data = NULL, geom = "vline",
             position = "identity", na.rm = FALSE, show.legend = NA,
             inherit.aes = TRUE, show = c("LSL", "USL"), direction = "v",
             type = NA, ...)

```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit

method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
digits	-
mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
geom	The geometric object to use display the data
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	-
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
show	-
direction	-
type	-
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.

Value

ggplot control charts.

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
```

```

set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
  Value = rnorm(100,10,1),
  Subgroup = rep(1:20, each=5),
  Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
  Value = rnorm(100,20, 1),
  Subgroup = rep(1:10, each=10),
  Process_run_id = 101:200)

df <- rbind(Process1, Process2)

#####
## Example 1 XmR  ##
#####
##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"

method <- "XmR"

# Normal Histogram XmR -----

EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +
  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) +
  ylim(0,45)
#Ex1.1

# Facet Histogram XmR -----

EX1.2 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),#show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +

```

```

    scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
    facet_grid(.~ProcessID) + ylim(0,45)
#EX1.2

# Facet Density Plot XmR -----

EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +

    scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) + ylim(0,.5)
#EX1.3

# Facet Density Plot XmR -----

EX1.4 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_density(bw = .4, fill="grey", trim=TRUE) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  # geom_hline(yintercept=0, color="black") +
  facet_grid(.~ProcessID) + ylim(0,.5)
#EX1.4

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####

method <- "xBar.rBar" #Alternatively Use "xBar.sBar" if desired

# Single Histogram xBar.rBar -----

EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +

```

```

stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) #+
#EX2.1

```

```

# Faceted Histogram xBar.rBar -----

```

```

EX2.2 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))+
  facet_grid(.~ProcessID, scales="free_x")
#EX2.2

```

```

# Single Density xBar.rBar -----

```

```

EX2.3 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) #+
#EX2.3

```

```

# Faceted Density xBar.rBar -----

```

```

EX2.4 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,

```



```

#show="ALL",
#show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
#       "LCL", "X", "UCL", "Sig"),
#show=c("Sig","TOL", "DNS"),
show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))+
  facet_grid(.~ProcessID, scales="free_x")
#EX2.4

#####
## Example 3: xBar.rMedian ##
#####

## Plots involving medians should give warning: "median based QC methods represent
## at best *potential* process capability"

##These plot work the same as in examples 2.X; below is an example.

method <- "xBar.rMedian"
EX3.1 <- ggplot(df[order(df$Process_run_id),], aes(x=Value, QC.Subgroup=Run)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
#show="ALL",
#show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
#       "LCL", "X", "UCL", "Sig"),
#show=c("Sig","TOL", "DNS"),
show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))
#EX3.1

```

stat_QC_Capability *Auto QC Capability Stat Function*

Description

Draws lines, labels and summary statistics. Works best with histogram and density plots.

Usage

```

stat_QC_Capability(LSL, USL, method = "xBar.rBar",
  show.lines = c("LSL", "USL"), line.direction = "v",
  show.line.labels = TRUE, line.label.size = 3,
  show.cap.summary = c("Cp", "Cpk", "Pp", "Ppk"), cap.summary.size = 4,
  px = Inf, py = -Inf, digits = 3)

```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
show.lines	vector, indicating which lines to draw ie., c("LCL", "LSL", "X", "USL", "UCL") <ul style="list-style-type: none"> • LCL: Lower Control Limit • LSL: Lower Specification Limit • X: Process Center • USL: Upper Specification Limit • UCL: Upper Control Limit
line.direction	string "v" or "h", specifies which direction to draw lines.
show.line.labels	boolean, if TRUE then draw.
line.label.size	numeric, control the size of the line labels.
show.cap.summary	vector, indicating which lines to draw ie., c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk", "LCL", "X", "UCL", "Sig"). The order given in the vector is the order presented in the graph. <ul style="list-style-type: none"> • TOL: Tolerance in Sigma Units (USL-LSL)/sigma • DNS: Distance to Nearest Specification Limit in Simga Units • Cp: Cp (Within) • Cpk: Cpk (Within) • Pp: Pp (Between) • Ppk: Ppk (Between) • LCL: Lower Control Limit • X: Process Center • UCL: Upper Control Limit • Sig: Sigma from control charts
cap.summary.size	numeric, control the size/scale of the summary text box.
px	numeric, x position for summary text box. Use Inf to force label to x-limit.
py	numeric, y position for summary text box. Use Inf to force label to y-limits. May also need vjust parameter.
digits	integer, how many digits to report.

Value

capability layer for histogram and density plots.

See Also

for more control over lines, labels, and capability data see the following functions:

- [stat_QC_cap_vlabels](#)
- [stat_QC_cap_hlabels](#)
- [stat_QC_cap_vlines](#)
- [stat_QC_cap_hlines](#)
- [stat_QC_cap_summary](#)

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)
# Setup Data -----
set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
                       Value = rnorm(100,10,1),
                       Subgroup = rep(1:20, each=5),
                       Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
                       Value = rnorm(100,20, 1),
                       Subgroup = rep(1:10, each=10),
                       Process_run_id = 101:200)

df <- rbind(Process1, Process2)

#####
## Example 1 XmR ##
#####

##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"

# Normal Histogram XmR -----
EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_Capability(LSL=5, USL=15, show.cap.summary = "all", method="XmR") +
  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) +
  ylim(0,45)
#Ex1.1

# Facet Histogram XmR -----
EX1.2 <- ggplot(df[order(df$Process_run_id),],
               aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_Capability(LSL=5, USL=15, show.cap.summary = "all", method="XmR") +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  facet_grid(.~ProcessID, scales = "free_x") + ylim(0,45)
```

```

#EX1.2

# Normal Density XmR -----
EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_Capability(LSL=5, USL=15, show.cap.summary = "all", method="XmR") +
  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) + ylim(0,.5)
#EX1.3

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####
# Single Histogram xBar.rBar -----
EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_Capability(LSL=5, USL=15, method="xBar.rBar") +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.1

```

stat_QC_cap_hlabels *horizontal Label Capability Stat*

Description

Draws horizontal Lables on horizontal Capability lines

Usage

```
stat_QC_cap_hlabels(LSL, USL, method = "xBar.rBar", show = c("LSL",
  "USL"), mapping = NULL, data = NULL, inherit.aes = TRUE, ...)
```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
show	vector, indicating which lines to draw ie., c("LCL", "LSL", "X", "USL", "UCL") <ul style="list-style-type: none"> • LCL: Lower Control Limit • LSL: Lower Specification Limit • X: Process Center • USL: Upper Specification Limit • UCL: Upper Control Limit

mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .

Value

horizontal lines for histogram and density plots.

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
                       Value = rnorm(100,10,1),
                       Subgroup = rep(1:20, each=5),
                       Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
                       Value = rnorm(100,20, 1),
                       Subgroup = rep(1:10, each=10),
                       Process_run_id = 101:200)

df <- rbind(Process1, Process2)

#####
## Example 1 XmR  ##
#####
##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"

method <- "XmR"
```

```

# Normal Histogram XmR -----
EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +
  scale_x_continuous(expand = expand_scale(mult = c(0.15, .8))) +
  ylim(0,45)
#Ex1.1

# Facet Histogram XmR -----
EX1.2 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"), #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) +
  facet_grid(.~ProcessID) + ylim(0,45)
#EX1.2

# Facet Density Plot XmR -----
EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +

  scale_x_continuous(expand = expand_scale(mult = c(0.15, .8))) + ylim(0, .5)

```

```

#EX1.3

# Facet Density Plot XmR -----

EX1.4 <- ggplot(df[order(df$Process_run_id),],
               aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_density(bw = .4, fill="grey", trim=TRUE ) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                    #      "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) +
  # geom_hline(yintercept=0, color="black") +
  facet_grid(.~ProcessID) + ylim(0, .5)
#EX1.4

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####

method <- "xBar.rBar" #Alternatively Use "xBar.sBar" if desired

# Single Histogram xBar.rBar -----

EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                    #      "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) #+
#EX2.1

# Faceted Histogram xBar.rBar -----

EX2.2 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,

```

```

        #show="ALL",
        #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
        #      "LCL", "X", "UCL", "Sig"),
        #show=c("Sig","TOL", "DNS"),
        show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
        color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))+
  facet_grid(.~ProcessID, scales="free_x")
#EX2.2

# Single Density xBar.rBar -----

EX2.3 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.3

# Faceted Density xBar.rBar -----

EX2.4 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))+
  facet_grid(.~ProcessID, scales="free_x")
#EX2.4

#####
## Example 3: xBar.rMedian ##
#####

## Plots involving medians should give warning: "median based QC methods represent
## at best *potential* process capability"

##These plot work the same as in examples 2.X; below is an example.

```



```

method <- "xBar.rMedian"
EX3.1 <- ggplot(df[order(df$Process_run_id),], aes(x=Value, QC.Subgroup=Run)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))
#EX3.1

```

stat_QC_cap_hlines *horizontal Line Capability Stat*

Description

Draws horizontal Capability Lines

Usage

```
stat_QC_cap_hlines(LSL, USL, method = "xBar.rBar", show = c("LSL",
  "USL"), mapping = NULL, data = NULL, inherit.aes = TRUE, ...)
```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
show	vector, indicating which lines to draw ie., c("LCL", "LSL", "X", "USL", "UCL") <ul style="list-style-type: none"> • LCL: Lower Control Limit • LSL: Lower Specification Limit • X: Process Center • USL: Upper Specification Limit • UCL: Upper Control Limit
mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data	<p>The data to be displayed in this layer. There are three options:</p> <p>If NULL, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data.</p>
inherit.aes	<p>If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code>.</p>
...	<p>Other arguments passed on to <code>layer()</code>. These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code>. They may also be parameters to the paired <code>geom/stat</code>.</p>

Value

horizontal lines for histogram and density plots.

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
                      Value = rnorm(100,10,1),
                      Subgroup = rep(1:20, each=5),
                      Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
                      Value = rnorm(100,20, 1),
                      Subgroup = rep(1:10, each=10),
                      Process_run_id = 101:200)

df <- rbind(Process1, Process2)

#####
## Example 1 XmR  ##
#####
##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"

method <- "XmR"

# Normal Histogram XmR -----
```

```

EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +
  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) +
  ylim(0,45)
#Ex1.1

```

```

# Facet Histogram XmR -----

```

```

EX1.2 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),#show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  facet_grid(.~ProcessID) + ylim(0,45)
#EX1.2

```

```

# Facet Density Plot XmR -----

```

```

EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +

  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) + ylim(0,.5)
#EX1.3

```

```

# Facet Density Plot XmR -----

```

```

EX1.4 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_density(bw = .4, fill="grey", trim=TRUE ) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  # geom_hline(yintercept=0, color="black") +
  facet_grid(.~ProcessID) + ylim(0,.5)
#EX1.4

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####

method <- "xBar.rBar" #Alternatively Use "xBar.sBar" if desired

# Single Histogram xBar.rBar -----

EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.1

# Faceted Histogram xBar.rBar -----

EX2.2 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),

```

```

        #show=c("Sig","TOL", "DNS"),
        show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
        color="black", digits=4, size=4) +
    scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))+
    facet_grid(.~ProcessID, scales="free_x")
#EX2.2

# Single Density xBar.rBar -----

EX2.3 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #       "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.3

# Faceted Density xBar.rBar -----

EX2.4 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #       "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8)))+
  facet_grid(.~ProcessID, scales="free_x")
#EX2.4

#####
## Example 3: xBar.rMedian ##
#####

## Plots involving medians should give warning: "median based QC methods represent
## at best *potential* process capability"

##These plot work the same as in examples 2.X; below is an example.

method <- "xBar.rMedian"
EX3.1 <- ggplot(df[order(df$Process_run_id),], aes(x=Value, QC.Subgroup=Run)) +
  geom_histogram(binwidth = 1) +

```

```

stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                    #      "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))
#EX3.1

```

stat_QC_cap_summary *horizontal Label Capability Stat*

Description

Draws horizontal Lables on horizontal Capability lines

Usage

```

stat_QC_cap_summary(LSL, USL, method = "xBar.rBar", px = Inf,
  py = -Inf, show = c("Cp", "Cpk", "Pp", "Ppk"), digits = 8,
  mapping = NULL, data = NULL, inherit.aes = TRUE, ...)

```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
px	numeric, x position for table. Use Inf to force label to x-limit.
py	numeric, y position for table. Use Inf to force label to y-limits. May also need vjust parameter.
show	vector, indicating which lines to draw ie., c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk", "LCL", "X", "UCL", "Sig"). The order given in the vector is the order presented in the graph. <ul style="list-style-type: none"> • TOL: Tolerance in Sigma Units (USL-LSL)/sigma • DNS: Distance to Nearest Specification Limit in Simga Units • Cp: Cp (Within) • Cpk: Cpk (Within) • Pp: Pp (Between) • Ppk: Ppk (Between)

	<ul style="list-style-type: none"> • LCL: Lower Control Limit • X: Process Center • UCL: Upper Control Limit • Sig: Sigma from control charts
digits	integer, how many digits to report.
mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data.</p>
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .

Value

horizontal lines for histogram and density plots.

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
  Value = rnorm(100,10,1),
  Subgroup = rep(1:20, each=5),
  Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
  Value = rnorm(100,20, 1),
  Subgroup = rep(1:10, each=10),
  Process_run_id = 101:200)

df <- rbind(Process1, Process2)
```

```
#####
## Example 1 XmR ##
#####
##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"

method <- "XmR"

# Normal Histogram XmR -----

EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    # "LCL", "X", "UCL", "Sig"),
    #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +
  scale_x_continuous(expand = expand_scale(mult = c(0.15, .8))) +
  ylim(0,45)
#Ex1.1

# Facet Histogram XmR -----

EX1.2 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    # "LCL", "X", "UCL", "Sig"), #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) +
  facet_grid(.~ProcessID) + ylim(0,45)
#EX1.2

# Facet Density Plot XmR -----

EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
```



```

#show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
#       "LCL", "X", "UCL", "Sig"),
#show=c("Sig","TOL", "DNS"),
show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
color="black", digits=2, size=4) +

  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) + ylim(0,.5)
#EX1.3

# Facet Density Plot XmR -----

EX1.4 <- ggplot(df[order(df$Process_run_id),],
               aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_density(bw = .4, fill="grey", trim=TRUE) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                     #show="ALL",
                     #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
                     #       "LCL", "X", "UCL", "Sig"),
                     #show=c("Sig","TOL", "DNS"),
                     show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
                     color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  # geom_hline(yintercept=0, color="black") +
  facet_grid(.~ProcessID) + ylim(0,.5)
#EX1.4

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####

method <- "xBar.rBar" #Alternatively Use "xBar.sBar" if desired

# Single Histogram xBar.rBar -----

EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                     #show="ALL",
                     #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
                     #       "LCL", "X", "UCL", "Sig"),
                     #show=c("Sig","TOL", "DNS"),
                     show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
                     color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.1

```

```

# Faceted Histogram xBar.rBar -----
EX2.2 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  facet_grid(.~ProcessID, scales="free_x")
#EX2.2

# Single Density xBar.rBar -----
EX2.3 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
#EX2.3

# Faceted Density xBar.rBar -----
EX2.4 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  facet_grid(.~ProcessID, scales="free_x")
#EX2.4

#####
## Example 3: xBar.rMedian ##

```

```
#####

## Plots involving medians should give warning: "median based QC methods represent
## at best *potential* process capability"

##These plot work the same as in examples 2.X; below is an example.

method <- "xBar.rMedian"
EX3.1 <- ggplot(df[order(df$Process_run_id),], aes(x=Value, QC.Subgroup=Run)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                     #show="ALL",
                     #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                     #show=c("Sig", "TOL", "DNS"),
                     show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                     color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))
#EX3.1
```

stat_QC_cap_vlabels *Vertical Label Capability Stat*

Description

Draws Vertical Lables on Vertical Capability lines

Usage

```
stat_QC_cap_vlabels(LSL, USL, method = "xBar.rBar", show = c("LSL",
  "USL"), mapping = NULL, data = NULL, inherit.aes = TRUE, ...)
```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
show	vector, indicating which lines to draw ie., c("LCL", "LSL", "X", "USL", "UCL") <ul style="list-style-type: none"> • LCL: Lower Control Limit • LSL: Lower Specification Limit • X: Process Center • USL: Upper Specification Limit

- **UCL:** Upper Control Limit

mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .

Value

vertical lines for histogram and density plots.

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
                       Value = rnorm(100,10,1),
                       Subgroup = rep(1:20, each=5),
                       Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
                       Value = rnorm(100,20, 1),
                       Subgroup = rep(1:10, each=10),
                       Process_run_id = 101:200)

df <- rbind(Process1, Process2)

#####
## Example 1 XmR  ##
#####
##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"
```

```

method <- "XmR"

# Normal Histogram XmR -----

EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +
  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) +
  ylim(0,45)
#Ex1.1

# Facet Histogram XmR -----

EX1.2 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),#show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  facet_grid(.~ProcessID) + ylim(0,45)
#EX1.2

# Facet Density Plot XmR -----

EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +

```

```

    scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) + ylim(0,.5)
#EX1.3

# Facet Density Plot XmR -----

EX1.4 <- ggplot(df[order(df$Process_run_id),],
               aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_density(bw = .4, fill="grey", trim=TRUE ) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
                    #      "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig","TOL", "DNS"),
                    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  # geom_hline(yintercept=0, color="black") +
  facet_grid(.~ProcessID) + ylim(0,.5)
#EX1.4

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####

method <- "xBar.rBar" #Alternatively Use "xBar.sBar" if desired

# Single Histogram xBar.rBar -----

EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
                    #      "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig","TOL", "DNS"),
                    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.1

# Faceted Histogram xBar.rBar -----

EX2.2 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +

```

```

stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))+
facet_grid(.~ProcessID, scales="free_x")
#EX2.2

# Single Density xBar.rBar -----

EX2.3 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
geom_density(bw = .4, fill="grey", alpha=.4) +
stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) #+
#EX2.3

# Faceted Density xBar.rBar -----

EX2.4 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
geom_density(bw = .4, fill="grey", alpha=.4) +
stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                              # "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))+
facet_grid(.~ProcessID, scales="free_x")
#EX2.4

#####
## Example 3: xBar.rMedian ##
#####

## Plots involving medians should give warning: "median based QC methods represent
## at best *potential* process capability"

```

```
##These plot work the same as in examples 2.X; below is an example.

method <- "xBar.rMedian"
EX3.1 <- ggplot(df[order(df$Process_run_id),], aes(x=Value, QC.Subgroup=Run)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))
#EX3.1
```

stat_QC_cap_vlines *Vertical Line Capability Stat*

Description

Draws Vertical Capability Stats

Usage

```
stat_QC_cap_vlines(LSL, USL, method = "xBar.rBar", show = c("LSL",
  "USL"), mapping = NULL, data = NULL, inherit.aes = TRUE, ...)
```

Arguments

LSL	numeric, Customer's lower specification limit
USL	numeric, Customer's Upper specification limit
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
show	vector, indicating which lines to draw ie., c("LCL", "LSL", "X", "USL", "UCL") <ul style="list-style-type: none"> • LCL: Lower Control Limit • LSL: Lower Specification Limit • X: Process Center • USL: Upper Specification Limit • UCL: Upper Control Limit
mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data	<p>The data to be displayed in this layer. There are three options:</p> <p>If NULL, the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code>.</p> <p>A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data.</p>
inherit.aes	<p>If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code>.</p>
...	<p>Other arguments passed on to <code>layer()</code>. These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code>. They may also be parameters to the paired geom/stat.</p>

Value

vertical lines for histogram and density plots.

Examples

```
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(ProcessID = as.factor(rep(1,100)),
                      Value = rnorm(100,10,1),
                      Subgroup = rep(1:20, each=5),
                      Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(ProcessID = as.factor(rep(2,100)),
                      Value = rnorm(100,20, 1),
                      Subgroup = rep(1:10, each=10),
                      Process_run_id = 101:200)

df <- rbind(Process1, Process2)

#####
## Example 1 XmR  ##
#####
##You may need to use the r-studio Zoom for these plots or make the size of the
##stat_QC_cap_summary smaller with size = some number"

method <- "XmR"

# Normal Histogram XmR -----
```

```
EX1.1 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1, color="purple") +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +
  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) +
  ylim(0,45)
#Ex1.1
```

```
# Facet Histogram XmR -----
```

```
EX1.2 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_histogram(binwidth = 1) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),#show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  facet_grid(.~ProcessID) + ylim(0,45)
#EX1.2
```

```
# Facet Density Plot XmR -----
```

```
EX1.3 <- ggplot(df[df$ProcessID == 1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="purple", trim=TRUE) +
  geom_hline(yintercept=0, color="grey") +
  stat_QC_cap_vlines(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, show=c("X", "LSL", "USL"), method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=2, size=4) +

  scale_x_continuous(expand = expand_scale(mult = c(0.15,.8))) + ylim(0,.5)
#EX1.3
```

```
# Facet Density Plot XmR -----
```

```

EX1.4 <- ggplot(df[order(df$Process_run_id),],
  aes(x=Value, QC.Subgroup=Subgroup, color=ProcessID)) +
  geom_density(bw = .4, fill="grey", trim=TRUE ) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) +
  # geom_hline(yintercept=0, color="black") +
  facet_grid(.~ProcessID) + ylim(0,.5)
#EX1.4

#####
## Example 2: xBar.rBar or xBar.sBar ##
#####

method <- "xBar.rBar" #Alternatively Use "xBar.sBar" if desired

# Single Histogram xBar.rBar -----

EX2.1 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig","TOL", "DNS"),
    show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15,.8))) #+
#EX2.1

# Faceted Histogram xBar.rBar -----

EX2.2 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_histogram(binwidth = 1) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL","DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),

```

```

        #show=c("Sig", "TOL", "DNS"),
        show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
        color="black", digits=4, size=4) +
    scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))+
    facet_grid(.~ProcessID, scales="free_x")
#EX2.2

# Single Density xBar.rBar -----

EX2.3 <- ggplot(df[df$ProcessID==1,], aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8))) #+
#EX2.3

# Faceted Density xBar.rBar -----

EX2.4 <- ggplot(df, aes(x=Value, QC.Subgroup=Subgroup)) +
  geom_density(bw = .4, fill="grey", alpha=.4) +
  stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
  stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
    #show="ALL",
    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
    #      "LCL", "X", "UCL", "Sig"),
    #show=c("Sig", "TOL", "DNS"),
    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))+
  facet_grid(.~ProcessID, scales="free_x")
#EX2.4

#####
## Example 3: xBar.rMedian ##
#####

## Plots involving medians should give warning: "median based QC methods represent
## at best *potential* process capability"

##These plot work the same as in examples 2.X; below is an example.

method <- "xBar.rMedian"
EX3.1 <- ggplot(df[order(df$Process_run_id),], aes(x=Value, QC.Subgroup=Run)) +
  geom_histogram(binwidth = 1) +

```

```

stat_QC_cap_vlines(LSL = 5, USL = 15, method=method) +
stat_QC_cap_vlabels(LSL = 5, USL = 15, method=method) +
stat_QC_cap_summary(LSL = 5, USL = 15, method=method, #py=.3,
                    #show="ALL",
                    #show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk",
                    #      "LCL", "X", "UCL", "Sig"),
                    #show=c("Sig", "TOL", "DNS"),
                    show=c("TOL", "DNS", "Cp", "Cpk", "Pp", "Ppk"),
                    color="black", digits=4, size=4) +
  scale_x_continuous(expand = ggplot2::expand_scale(mult = c(0.15, .8)))
#EX3.1

```

stat_QC_labels

Write QC Line Labels to ggplot QC Charts.

Description

Write QC line labels to ggplot QC Charts. Useful if you want to see the value of the center line and QC limits. see method argument for methods supported.

Usage

```

stat_QC_labels(mapping = NULL, data = NULL, geom = "label",
              position = "identity", na.rm = FALSE, show.legend = NA,
              inherit.aes = TRUE, n = NULL, digits = 1, method = "xBar.rBar",
              color.qc_limits = "red", color.qc_center = "black", text.size = 3,
              physical.limits = c(NA, NA), limit.txt.label = c("LCL", "UCL"), ...)

```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_() . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to ggplot() . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
geom	The geometric object to use display the data
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
n	number, for <ul style="list-style-type: none"> • Studentized Charts, used for custom or hypothetical subgroup size. • np Charts, used to specify a fixed area of opportunity.
digits	integer, indicating the number of decimal places
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: mR, XmR, • Attribute Charts: c, np, p, u, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian • Dispersion Charts: rBar, rMedian, sBar.
color.qc_limits	color, used to colorize the plot's upper and lower mR control limits.
color.qc_center	color, used to colorize the plot's center line.
text.size	number, size of the text label
physical.limits	vector, specify lower physical boundary and upper physical boundary
limit.txt.label	vector, provides option for naming or not showing the limit text labels (e.g., UCL, LCL) <ul style="list-style-type: none"> • limit.txt.label = c("LCL", "UCL"): default • limit.txt.label = c("Low", "High"): changes the label text to low and high • limit.txt.label = NA: does not show label text.
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.

Value

data need to produce the mR plot in ggplot.

Examples

```
#####
# Example 1: mR Chart #
#####

# Load Libraries -----
require(ggQC)
```

```

require(ggplot2)

# Setup Data -----
set.seed(5555)
Process1 <- data.frame(processID = as.factor(rep(1,100)),
                       metric_value = rnorm(100,0,1),
                       subgroup_sample=rep(1:20, each=5),
                       Process_run_id = 1:100)

set.seed(5556)
Process2 <- data.frame(processID = as.factor(rep(2,100)),
                       metric_value = rnorm(100,5, 1),
                       subgroup_sample=rep(1:10, each=10),
                       Process_run_id = 101:200)

Both_Processes <- rbind(Process1, Process2)

# Facet Plot - Both Processes -----
EX1.1 <- ggplot(Both_Processes, aes(x=Process_run_id, y = metric_value)) +
  geom_point() + geom_line() + stat_QC(method="XmR") +
  stat_QC_labels(method="XmR", digits = 2) +
  facet_grid(~processID, scales = "free_x")
#EX1.1

EX1.2 <- ggplot(Both_Processes, aes(x=Process_run_id, y = metric_value)) +
  stat_mR() + ylab("Moving Range") +
  stat_QC_labels(method="mR", digits = 2) +
  facet_grid(~processID, scales = "free_x")
#EX1.2

#####
# Example 2: XbarR Chart #
#####
# Facet Plot - Studentized Process -----

EX2.1 <- ggplot(Both_Processes, aes(x=subgroup_sample,
                                   y = metric_value,
                                   group = processID)) +
  geom_point(alpha=.2) +
  stat_summary(fun.y = "mean", color="blue", geom=c("point")) +
  stat_summary(fun.y = "mean", color="blue", geom=c("line")) +
  stat_QC() + facet_grid(~processID, scales = "free_x") +
  stat_QC_labels(text.size =3, label.size=.1)
#EX2.1

EX2.2 <- ggplot(Both_Processes, aes(x=subgroup_sample,
                                   y = metric_value,
                                   group = processID)) +
  stat_summary(fun.y = "QCrange", color="blue", geom = "point") +
  stat_summary(fun.y = "QCrange", color="blue", geom = "line") +
  stat_QC(method="rBar") +
  stat_QC_labels(digits=2, method="rBar") +
  ylab("Range") +
  facet_grid(~processID, scales = "free_x")

```

#EX2.2

 stat_qc_violations *Inspect QC Violations*

Description

ggplot stat function that renders a faceted plot of QC violations based on the following 4 rules:

- **Violation Same Side:** 8 or more consecutive, same-side points
- **Violation 1 Sigma:** 4 or more consecutive, same-side points exceeding 1 sigma
- **Violation 2 Sigma:** 2 or more consecutive, same-side points exceeding 2 sigma
- **Violation 3 Sigma:** any points exceeding 3 sigma

Usage

```
stat_qc_violations(mapping = NULL, data = NULL, geom = "point",
  position = "identity", show.legend = NA, inherit.aes = TRUE,
  na.rm = FALSE, method = "xBar.rBar", geom_points = TRUE,
  geom_line = TRUE, point.size = 1.5, point.color = "black",
  violation_point.color = "red", line.color = NULL,
  rule.color = "darkgreen", show.facets = c(1:4), ...)
```

Arguments

mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data.
geom	The geometric object to use display the data
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .

na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
method	string, calling the following methods: <ul style="list-style-type: none"> • Individuals Charts: XmR, • Studentized Charts: xBar.rBar, xBar.rMedian, xBar.sBar, xMedian.rBar, xMedian.rMedian
geom_points	boolean, draw points
geom_line	boolean, draw line
point.size	number, size of points on chart
point.color	string, color of points on charts (e.g., "black")
violation_point.color	string, color of violation points on charts (e.g., "red")
line.color	string, color of lines connecting points
rule.color	string, color or horizontal rules indicating distribution center and sigma levels
show.facets	vector, selects violation facet 1 through 4. eg., c(1:4), c(1,4)
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.

Value

faceted plot.

Examples

```
#####
# Example 1: XmR Check Violations #
#####
# Load Libraries -----
require(ggQC)
require(ggplot2)

# Setup Data -----

set.seed(5555)
QC_XmR <- data.frame(
  data = c(c(-1, 2.3, 2.4, 2.5), #Outlier Data
           sample(c(rnorm(60),5,-5), 62, replace = FALSE), #Normal Data
           c(1,-.3, -2.4,-2.6,-2.5,-2.7, .3)), #Outlier Data
  Run_Order = 1:73 #Run Order
)

# Render QC Violation Plot -----

EX1 <- ggplot(QC_XmR, aes(x = Run_Order, y = data)) +
  stat_qc_violations(method = "XmR") #Makes facet graph with violations
```

```

#EX1
#####
# Example 2: Xbar Check Violations #
#####

# Setup Some Data -----
QC_xBar.rBar <- do.call(rbind, lapply(1:3, function(X){
  set.seed(5555+X)                                #Loop over 3 seeds
  data.frame(
    sub_group = rep(1:42),                          #Define Subgroups
    sub_class = letters[X],
    c(
      c(runif(n = 5, min = 2.0,3.2)),                #Outlier Data
      sample(c(rnorm(30),5,-4), 32, replace = FALSE), #Normal Data
      c(runif(n = 5, min = -3.2, max = -2.0))        #Outlier Data
    )
  )
})
)

colnames(QC_xBar.rBar) <- c("sub_group","sub_class", "value")

# Render QC Violation Plot -----
EX2 <- ggplot(QC_xBar.rBar, aes(x = sub_group, y = value)) +
  stat_qc_violations(method = "xBar.rBar")
  #stat_qc_violations(method="xBar.rMedian")
  #stat_qc_violations(method="xBar.sBar")
  #stat_qc_violations(method="xMedian.rBar")
  #stat_qc_violations(method="xMedian.rMedian")
#EX2

#####
# Example 3: Selected Facets #
#####

# Render QC Violation Plot -----
EX3 <- ggplot(QC_xBar.rBar, aes(x = sub_group, y = value)) +
  stat_qc_violations(method = "xBar.rBar", show.facets = c(4))

#EX3

#####
# Complete User Control - Bypass stat_qc_violation #
#####
#### The code below has two options if you are looking for complete
#### control over the look and feel of the graph. Use option 1 or option
#### 2 as appropriate. If you want something quick and easy use examples above.

##### Option 1: Setup for XmR Type Data
# QC_XmR: Defined in Example 1
QC_Vs <- QC_Violations(data = QC_XmR$data, method = "XmR")

```

```

QC_Stats <- QC_Lines(data = QC_XmR$data, method = "XmR")
MEAN <- QC_Stats$mean
SIGMA <- QC_Stats$sigma

##### Option 2: Setup for xBar.rBar Type Data
# QC_xBar.rBar: Defined in Example 2
QC_Vs <- QC_Violations(data = QC_xBar.rBar,
                       formula = value~sub_group,
                       method = "xBar.rBar")
QC_Stats <- QC_Lines(data = QC_xBar.rBar,
                    formula = value~sub_group,
                    method = "xBar.rBar")
MEAN <- QC_Stats$xBar_Bar
SIGMA <- QC_Stats$sigma

##### Setup second table for horizontal rules
FacetNames <- c("Violation Same Side",
               "Violation 1 Sigma",
               "Violation 2 Sigma",
               "Violation 3 Sigma")

QC_Vs$Violation_Result <- ordered(QC_Vs$Violation_Result,
                                 levels=FacetNames)

QC_Stats_df <- data.frame(
  Violation_Result = factor(x = FacetNames, levels = FacetNames),
  SigmaPlus = MEAN+SIGMA*0:3,
  MEAN = MEAN,
  SigmaMinus = MEAN-SIGMA*0:3
)

##### Make the Plot
ggplot(QC_Vs, aes(x=Index, y=data, color=Violation, group=1)) +
  geom_point() + geom_line() +
  facet_grid(.~Violation_Result) +
  geom_hline(data = QC_Stats_df, aes(yintercept = c(SigmaPlus))) +
  geom_hline(data = QC_Stats_df, aes(yintercept = c(SigmaMinus))) +
  geom_hline(data = QC_Stats_df, aes(yintercept = c(MEAN)))

```

uBar

Mean Rate: Count Data (u-chart)

Description

Calculates overall mean rate for count data acquired over a variable area of opportunity.

Usage

```
uBar(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of counts per unit opportunity (rate). Observations may have a different area of opportunity, n.
n	A vector representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A vector of mean rate, length equal to length of parameter y.

Examples

```
set.seed(5555)
counts <- rpois(100, 25)
n <- rpois(100, 15)
uBar(y = counts / n, n = n)
```

uBar_LCL

Lower Control Limit: Count Data (u-chart)

Description

Calculates point-wise lower control limit (LCL) for count data acquired over a variable area of opportunity.

Usage

```
uBar_LCL(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of counts per unit opportunity (rate). Observations may have a different area of opportunity, n.
n	A vector representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A vector; point-wise 3-sigma lower control limit (LCL)

Examples

```
set.seed(5555)
counts <- rpois(100, 25)
n <- rpois(100, 15)
uBar_LCL(y = counts / n, n = n)
```

uBar_UCL

Upper Control Limit: Count Data (u-chart)

Description

Calculates point-wise upper control limit (UCL) for count data acquired over a variable area of opportunity.

Usage

```
uBar_UCL(y, n, na.rm = FALSE, ...)
```

Arguments

y	Vector of counts per unit opportunity (rate). Observations may have a different area of opportunity, n.
n	A vector representing the area of opportunity.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A vector; point-wise 3-sigma upper control limit (UCL)

Examples

```
set.seed(5555)
counts <- rpois(100, 25)
n <- rpois(100, 15)
uBar_UCL(y = counts / n, n = n)
```

UD *Calculate Distance to Upper Specification Limit*

Description

function to calculate a standardized distance to the Upper specification limit (sigma units)

Usage

UD(LSL, USL, QC.Center, QC.Sigma)

Arguments

LSL	number, customer's lower specification limit.
USL	number, customer's upper specification limit.
QC.Center	number, the mean or median value determined from an XmR plot or a Studentized (e.g., xBar) analysis.
QC.Sigma	number, the sigma value determined from an XmR plot or a Studentized (e.g., xBar) analysis.

Value

numeric, standardized distance to the upper specification limit (sigma units)

xBar_Bar *Mean of Subgroup Means*

Description

Calculates the mean subgroup means used when constructing a xBar-R or xBar-S charts.

Usage

xBar_Bar(data, value, grouping, formula = NULL, ...)

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean of subgroup means.

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_Bar(data = df, formula = v~g)
```

xBar_one_LCL

xBar_One Lower Control Limit (LCL)

Description

Calculates the xBar_One LCL used when constructing a xBar-One chart.

Usage

```
xBar_one_LCL(y, na.rm = FALSE, ...)
```

Arguments

y	Vector of values
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; xBar_One Lower Control Limit (LCL)

Examples

```
set.seed(5555)
values <- rnorm(n = 100, mean = 25, sd = 1)
xBar_one_LCL(values)
```

xBar_one_UCL	<i>xBar_One Upper Control Limit (UCL)</i>
--------------	---

Description

Calculates the xBar_One UCL used when constructing a xBar-One chart.

Usage

```
xBar_one_UCL(y, na.rm = FALSE, ...)
```

Arguments

y	Vector of values
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.

Value

A number; xBar_One Upper Control Limit (UCL)

Examples

```
set.seed(5555)
values <- rnorm(n = 100, mean = 25, sd = 1)
xBar_one_UCL(values)
```

xBar_rBar_LCL	<i>Mean of Subgroup Means Lower Control Limit (LCL)</i>
---------------	---

Description

Calculates the mean of subgroup means lower control limit used when constructing a xBar-R charts.

Usage

```
xBar_rBar_LCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```


Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means lower control limit.

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_rBar_LCL(data = df, formula = v~g)
```

xBar_rBar_UCL	<i>Mean of Subgroup Means Upper Control Limit (UCL)</i>
---------------	---

Description

Calculates the mean of subgroup means upper control limit used when constructing a xBar-R charts.

Usage

```
xBar_rBar_UCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means upper control limit.

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_rBar_UCL(data = df, formula = v~g)
```

xBar_rMedian_LCL	<i>Mean of Subgroup Means Lower Control Limit (LCL) based on Median Range</i>
------------------	---

Description

Calculates the mean of subgroup means lower control limit based on the median range. The result is used when constructing a xBar-rMedian charts.

Usage

```
xBar_rMedian_LCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means Lower Control Limit (LCL) based on Median Range

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_rMedian_LCL(data = df, formula = v~g)
```

xBar_rMedian_UCL	<i>Mean of Subgroup Means Upper Control Limit (UCL) based on Median Range</i>
------------------	---

Description

Calculates the mean of subgroup means upper control limit based on the median range. The result is used when constructing a xBar-rMedian charts.

Usage

```
xBar_rMedian_UCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means Upper Control Limit (UCL) based on Median Range

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_rMedian_UCL(data = df, formula = v~g)
```

xBar_sBar_LCL	<i>Mean of Subgroup Means Lower Control Limit (LCL) based on Standard Deviation</i>
---------------	---

Description

Calculates the mean of subgroup means lower control limit based on the standard deviation. The result is used when constructing a xBar-S charts.

Usage

```
xBar_sBar_LCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means Lower Control Limit (LCL) based on standard deviation

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_sBar_LCL(data = df, formula = v~g)
```

xBar_sBar_UCL	<i>Mean of Subgroup Means Upper Control Limit (UCL) based on Standard Deviation</i>
---------------	---

Description

Calculates the mean of subgroup means upper control limit based on the standard deviation. The result is used when constructing a xBar-S charts.

Usage

```
xBar_sBar_UCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means Upper Control Limit (UCL) based on standard deviation

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_sBar_UCL(data = df, formula = v~g)
```

xMedian_Bar	<i>Mean of Subgroup Medians</i>
-------------	---------------------------------

Description

Calculates the mean of subgroup medians used when constructing a xMedian-R charts.

Usage

```
xMedian_Bar(data, value, grouping, formula = NULL, ...)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
formula	a formula, such as $y \sim x1 + x2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables
...	further arguments passed to or from other methods.

Value

A number; mean of subgroup medians.

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xMedian_Bar(data = df, formula = v~g)
```

xMedian_rBar_LCL	<i>Mean of Subgroup Medians Lower Control Limit (LCL) based on Mean Range</i>
------------------	---

Description

Calculates the mean of subgroup medians lower control limit based on the mean range. The result is used when constructing a xMedian-R charts.

Usage

```
xMedian_rBar_LCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup medians Lower Control Limit (LCL) based on mean range

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xBar_rMedian_LCL(data = df, formula = v~g)
```

xMedian_rBar_UCL	<i>Mean of Subgroup Medians Upper Control Limit (UCL) based on mean Range</i>
------------------	---

Description

Calculates the mean of subgroup medians upper control limit based on the mean subgroup range. The result is used when constructing a xMedian-R charts.

Usage

```
xMedian_rBar_UCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup means Upper Control Limit (UCL) based on Median Range

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xMedian_rBar_UCL(data = df, formula = v~g)
```

xMedian_rMedian_LCL	<i>Mean of Subgroup Medians Lower Control Limit (LCL) based on Median Range</i>
---------------------	---

Description

Calculates the mean of subgroup medians lower control limit based on the median subgroup range. The result is used when constructing a xMedian-rMedian charts.

Usage

```
xMedian_rMedian_LCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup median Lower Control Limit (LCL) based on Median Range

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xMedian_rMedian_LCL(data = df, formula = v~g)
```

xMedian_rMedian_UCL *Mean of Subgroup Medians Upper Control Limit (UCL) based on Median Range*

Description

Calculates the mean of subgroup medians upper control limit based on the median subgroup range. The result is used when constructing a xMedian-rMedian charts.

Usage

```
xMedian_rMedian_UCL(data, value, grouping, n = NULL, natural = F,
  formula = NULL)
```

Arguments

data	data frame to be processed
value	numeric vector in a data frame with values of interest.
grouping	single factor/variable to split the data frame "values" by.
n	a number indicating a hypothetical subgroup size other than, function determined subgroup n determined by the floor length of subgroup values.
natural	logical, if TRUE calculate limits for individuals (n=1) else calculate for n determined by the floor length of subgroup values
formula	a formula, such as $y \sim x_1 + x_2$, where the y variable is numeric data to be split into groups according to the grouping x factors/variables

Value

A number; mean of subgroup median upper Control Limit (UCL) based on Median Range

Examples

```
set.seed(5555)
df <- data.frame(v=rnorm(60, 0, 1), g=rep(c("A","B","C","D","E"), each=12))
xMedian_rMedian_UCL(data = df, formula = v~g)
```

Index

`aes()`, [28](#), [30](#), [32](#), [37](#), [45](#), [49](#), [55](#), [60](#), [64](#), [69](#), [72](#)
`aes_()`, [28](#), [30](#), [32](#), [37](#), [45](#), [49](#), [55](#), [60](#), [64](#), [69](#),
[72](#)

`borders()`, [29](#), [30](#), [33](#), [37](#), [45](#), [50](#), [55](#), [60](#), [65](#),
[70](#), [72](#)

`capability.summary`, [3](#)
`cBar_LCL`, [4](#)
`cBar_UCL`, [4](#)
`Cp`, [5](#)
`Cpk`, [6](#)

`DNS`, [6](#)

`fortify()`, [28](#), [30](#), [32](#), [37](#), [45](#), [50](#), [55](#), [60](#), [65](#),
[69](#), [72](#)

`ggplot()`, [28](#), [30](#), [32](#), [37](#), [45](#), [50](#), [55](#), [60](#), [65](#),
[69](#), [72](#)

`layer()`, [29](#), [31](#), [33](#), [37](#), [45](#), [50](#), [55](#), [60](#), [65](#), [70](#),
[73](#)

`LD`, [7](#)

`mR`, [7](#)
`mR_points`, [8](#)
`mR_UCL`, [8](#)

`npBar`, [9](#)
`npBar_LCL`, [10](#)
`npBar_UCL`, [10](#)

`pBar`, [11](#)
`pBar_LCL`, [12](#)
`pBar_UCL`, [13](#)
`Pp`, [13](#)
`Ppk`, [14](#)
`process_tolerance`, [14](#)

`QC_Capability`, [16](#)

`QC_Lines`, [17](#)
`QC_Violations`, [20](#)
`QCrange`, [15](#)

`rBar`, [22](#)
`rBar_LCL`, [23](#)
`rBar_UCL`, [23](#)
`rMedian`, [24](#)
`rMedian_LCL`, [25](#)
`rMedian_UCL`, [25](#)

`sBar`, [26](#)
`sBar_LCL`, [27](#)
`sBar_UCL`, [27](#)
`stat_mR`, [28](#)
`stat_pareto`, [30](#)
`stat_QC`, [32](#)
`stat_QC_cap_hlabels`, [43](#), [44](#)
`stat_QC_cap_hlines`, [43](#), [49](#)
`stat_QC_cap_summary`, [43](#), [54](#)
`stat_QC_cap_vlabels`, [43](#), [59](#)
`stat_QC_cap_vlines`, [43](#), [64](#)
`stat_QC_CAPA`, [36](#)
`stat_QC_Capability`, [41](#)
`stat_QC_labels`, [32](#), [69](#)
`stat_qc_violations`, [72](#)

`uBar`, [75](#)
`uBar_LCL`, [76](#)
`uBar_UCL`, [77](#)
`UD`, [78](#)

`xBar_Bar`, [78](#)
`xBar_one_LCL`, [79](#)
`xBar_one_UCL`, [80](#)
`xBar_rBar_LCL`, [80](#)
`xBar_rBar_UCL`, [81](#)
`xBar_rMedian_LCL`, [82](#)
`xBar_rMedian_UCL`, [83](#)
`xBar_sBar_LCL`, [84](#)

xBar_sBar_UCL, [85](#)
xMedian_Bar, [86](#)
xMedian_rBar_LCL, [86](#)
xMedian_rBar_UCL, [87](#)
xMedian_rMedian_LCL, [88](#)
xMedian_rMedian_UCL, [89](#)