# Package 'ggtrendline'

October 13, 2022

**Version** 1.0.3

**Date** 2022-04-24

**Title** Add Trendline and Confidence Interval to 'ggplot'

**Maintainer** Weiping Mei <meiweipingg@163.com>

**Description** Add trendline and confidence interval of linear or nonlinear regression model and show equation to 'ggplot' as simple as possible. For a general overview of the methods used in this package, see Ritz and Streibig (2008) <doi:10.1007/978-0-387-09616-2> and Greenwell and Schubert Kabban (2014) <doi:10.32614/RJ-2014-009>.

**Depends** R (>= 3.5.2)

**Imports** stats, ggplot2

**Suggests** investr, AICcmodavg

**BugReports** https://github.com/PhDMeiwp/ggtrendline/issues

**License** GPL-3

**URL** https://github.com/PhDMeiwp/ggtrendline

**LazyLoad** true

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Weiping Mei [aut, cre, cph],
Guangchuang Yu [aut],
Brandon M. Greenwell [aut],
Jiangshan Lai [ctb],
Zhendu Mao [ctb],
Yu Umezawa [ctb],
Jun Zeng [ctb],
Jun Wang [ctb]

**Repository** CRAN

**Date/Publication** 2022-04-27 08:20:06 UTC

# R topics documented:



---

`ggtrendline` *Add Trendline and Confidence Interval to 'ggplot'*

---

## Description

Add trendline and confidence interval of linear or nonlinear regression model to 'ggplot', by using different models built in the 'ggtrendline()' function.
The function includes the following models:
"line2P" (formula as: y=a*x+b),
"line3P" (y=a*x^2+b*x+c),
"log2P" (y=a*ln(x)+b),
"exp2P" (y=a*exp(b*x)),
"exp3P" (y=a*exp(b*x)+c),
"power2P" (y=a*x^b),
and "power3P" (y=a*x^b+c).

## Usage

```
ggtrendline(
  x,
  y,
  model = "line2P",
  linecolor = "black",
  linetype = 1,
  linewidth = 0.6,
  CI.level = 0.95,
  CI.fill = "grey60",
  CI.alpha = 0.3,
  CI.color = "black",
  CI.lty = 2,
  CI.lwd = 0.5,
  summary = TRUE,
  show.eq = TRUE,
  yhat = FALSE,
```

```
  eq.x = NULL,
  eq.y = NULL,
  show.Rsquare = TRUE,
  show.pvalue = TRUE,
  Pvalue.corrected = TRUE,
  Rname = 0,
  Pname = 0,
  rrp.x = NULL,
  rrp.y = NULL,
  text.col = "black",
  eDigit = 3,
  eSize = 3,
  xlab = NULL,
  ylab = NULL
)
```

## Arguments

| | |
|---|---|
| x, y | the x and y arguments provide the x and y coordinates for the 'ggplot'. Any reasonable way of defining the coordinates is acceptable. |
| model | select which model to fit. Default is "line2P". The "model" should be one of c("line2P", "line3P", "log2P", "exp2P", "exp3P", "power2P", "power3P"), their formulas are as follows:<br>"line2P": y=a*x+b<br>"line3P": y=a*x^2+b*x+c<br>"log2P": y=a*ln(x)+b<br>"exp2P": y=a*exp(b*x)<br>"exp3P": y=a*exp(b*x)+c<br>"power2P": y=a*x^b<br>"power3P": y=a*x^b+c |
| linecolor | the color of regression line. Default is "black". |
| linetype | the type of regression line. Default is 1. Notes: linetype can be specified using either text c("blank","solid","dashed","dotted","dotdash","longdash","twodash") or number c(0, 1, 2, 3, 4, 5, 6). |
| linewidth | the width of regression line. Default is 0.6. |
| CI.level | level of confidence interval to use. Default is 0.95. |
| CI.fill | the color for filling the confidence interval. Default is "grey60". |
| CI.alpha | alpha value of filling color of confidence interval. Default is 0.3. |
| CI.color | line color of confidence interval. Default is "black". |
| CI.lty | line type of confidence interval. Default is 2. |
| CI.lwd | line width of confidence interval. Default is 0.5. |
| summary | summarizing the model fits. Default is TRUE. |
| show.eq | whether to show the regression equation, the value is one of c("TRUE", "FALSE"). |
| yhat | whether to add a hat symbol (^) on the top of "y" in equation. Default is FALSE. |
| eq.x, eq.y | equation position. |

| | |
|---|---|
| show.Rsquare | whether to show the R-square, the value is one of c("TRUE", "FALSE"). |
| show.pvalue | whether to show the P-value, the value is one of c("TRUE", "FALSE"). |
| Pvalue.corrected | if P-value corrected or not, the value is one of c("TRUE", "FALSE"). |
| Rname | to specify the character of R-square, the value is one of c(0, 1), corresponding to c(r^2, R^2). |
| Pname | to specify the character of P-value, the value is one of c(0, 1), corresponding to c(p, P). |
| rrp.x, rrp.y | the position for R square and P value. |
| text.col | the color used for the equation text. |
| eDigit | the numbers of digits for R square and P value. Default is 3. |
| eSize | font size of R square and P value. Default is 3. |
| xlab, ylab | labels of x- and y-axis. |

## Details

The values of each parameter of regression model can be found by typing trendline_sum function in this package.

The linear models (line2P, line3P, log2P) in this package are estimated by lm function, while the nonlinear models (exp2P, exp3P, power2P, power3P) are estimated by nls function (i.e., least-squares method).

## Value

No return value (called for side effects).

## References

Ritz C., and Streibig J. C. (2007) *Nonlinear Regression with R*. Springer.

Greenwell B. M., and Schubert Kabban C. M. (2014) *investr: An R Package for Inverse Estimation*. The R Journal, 6(1), 90-100.

## See Also

ggtrendline, stat_eq, stat_rrp, trendline_sum, nls, selfStart

## Examples

```
# library(ggplot2)
library(ggtrendline)
x <- c(1, 3, 6, 9,  13,   17)
y <- c(5, 8, 11, 13, 13.2, 13.5)

ggtrendline(x, y, model = "line2P")  # default
ggtrendline(x, y, model = "log2P", CI.fill = NA)  # CI lines only, without CI filling
```

```
ggtrendline(x, y, model = "exp2P", linecolor = "blue", linetype = 1, linewidth = 1) # set line
ggtrendline(x, y, model = "exp3P", CI.level = 0.99,
         CI.fill = "red", CI.alpha = 0.1, CI.color = NA, CI.lty = 2, CI.lwd = 1.5) # set CI
```

---

predFit *Predictions from a Fitted Model*

---

## Description

Generic prediction method for various types of fitted models. predFit can be used to obtain standard errors of fitted values and adjusted/unadjusted confidence/prediction intervals for objects of class "lm", "nls".

## Usage

```
predFit(object, ...)

## Default S3 method:
predFit(object, ...)

## S3 method for class 'nls'
predFit(
  object,
  newdata,
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = 0.95,
  adjust = c("none", "Bonferroni", "Scheffe"),
  k,
  ...
)
```

## Arguments

| | |
|---|---|
| object | An object that inherits from class "lm", "nls". |
| ... | Additional optional arguments. At present, no optional arguments are used. |
| newdata | An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used. |
| se.fit | A logical vaue indicating if standard errors are required. Default is FALSE. |
| interval | Type of interval to be calculated. Can be one of "none" (default), "confidence", or "prediction". Default is "none". |
| level | A numeric scalar between 0 and 1 giving the confidence level for the intervals (if any) to be calculated. Default is 0.95. |

| | |
|---|---|
| adjust | A logical value indicating if an adjustment should be made to the critical value used in calculating the confidence interval. This is useful for when the calibration curve is to be used multiple, say k, times. Default is FALSE. |
| k | The number times the calibration curve is to be used for computing a confidence/prediction interval. Only needed when adjust = "Bonferroni". |

## Value

No return value (called for side effects).

## Note

predFit function is from 'investr' package written by Brandon M. Greenwell.

## References

Greenwell B. M., and Schubert-Kabban, C. M. (2014) *investr: An R Package for Inverse Estimation*. The R Journal, 6(1), 90-100.

## See Also

predFit

---

SSexp2P *Self-Starting Nls 'exp2P' Regression Model*

---

## Description

This selfStart model evaluates the power regression function (formula as: y=a*exp(b*x)). It has an initial attribute that will evaluate initial estimates of the parameters 'a' and 'b' for a given set of data.

## Usage

```
SSexp2P(predictor, a, b)
```

## Arguments

| | |
|---|---|
| predictor | a numeric vector of values at which to evaluate the model. |
| a, b | The numeric parameters responding to the exp2P model. |

## Value

No return value (called for side effects).

## See Also

ggtrendline, SSexp3P, SSpower3P, nls, selfStart

## Examples

```
library(ggtrendline)
x<-1:5
y<-c(2,4,8,20,25)
xy<-data.frame(x,y)
getInitial(y ~ SSexp2P(x,a,b), data = xy)
## Initial values are in fact the converged values

fitexp2P <- nls(y~SSexp2P(x,a,b), data=xy)
summary(fitexp2P)

prediction <- predFit(fitexp2P , data.frame(x=x), se.fit = TRUE,
                          level = 0.95, interval = "confidence")
yfitexp2P <- prediction$fit
yfitexp2P  # output a matrix of predictions and bounds with column names fit, lwr, and upr.
```

---

SSexp3P *Self-Starting Nls 'exp3P' Regression Model*

---

## Description

This selfStart model evaluates the exponential regression function (formula as: y=a*exp(b*x)+c). It has an initial attribute that will evaluate initial estimates of the parameters a, b, and c for a given set of data.

## Usage

```
SSexp3P(predictor, a, b, c)
```

## Arguments

| | |
|---|---|
| predictor | a numeric vector of values at which to evaluate the model. |
| a, b, c | Three numeric parameters responding to the exp3P model. |

## Value

No return value (called for side effects).

## See Also

ggtrendline, SSexp3P, SSpower3P, nls, selfStart

## Examples

```
library(ggtrendline)
x<-1:5
y<-c(2,4,8,16,28)
xy<-data.frame(x,y)
getInitial(y ~ SSexp3P(x,a,b,c), data = xy)
## Initial values are in fact the converged values

fitexp3P <- nls(y~SSexp3P(x,a,b,c), data=xy)
summary(fitexp3P)

prediction <- predFit(fitexp3P , data.frame(x=x), se.fit = TRUE,
                          level = 0.95, interval = "confidence")
yfitexp3P <- prediction$fit
yfitexp3P  # output a matrix of predictions and bounds with column names fit, lwr, and upr.
```

---

SSpower2P *Self-Starting Nls 'power2P' Regression Model*

---

## Description

This selfStart model evaluates the power regression function (formula as: y=a*x^b). It has an initial attribute that will evaluate initial estimates of the parameters 'a' and 'b' for a given set of data.

## Usage

```
SSpower2P(predictor, a, b)
```

## Arguments

| | |
|---|---|
| predictor | a numeric vector of values at which to evaluate the model. |
| a, b | The numeric parameters responding to the exp2P model. |

## Value

No return value (called for side effects).

## See Also

ggtrendline, SSexp3P, SSpower3P, nls, selfStart

## Examples

```
library(ggtrendline)
x<-1:5
y<-c(2,4,8,20,25)
xy<-data.frame(x,y)
getInitial(y ~ SSpower2P(x,a,b), data = xy)
## Initial values are in fact the converged values

fitpower2P <- nls(y~SSpower2P(x,a,b), data=xy)
summary(fitpower2P)

prediction <- predFit(fitpower2P , data.frame(x=x), se.fit = TRUE,
                          level = 0.95, interval = "confidence")
yfitpower2P <- prediction$fit
yfitpower2P # output a matrix of predictions and bounds with column names fit, lwr, and upr.
```

---

SSpower3P *Self-Starting Nls 'power3P' Regression Model*

---

## Description

This selfStart model evaluates the power regression function (formula as: y=a*x^b+c). It has an initial attribute that will evaluate initial estimates of the parameters a, b, and c for a given set of data.

## Usage

```
SSpower3P(predictor, a, b, c)
```

## Arguments

| | |
|---|---|
| predictor | a numeric vector of values at which to evaluate the model. |
| a, b, c | Three numeric parameters responding to the exp3P model. |

## Value

No return value (called for side effects).

## See Also

ggtrendline, SSexp3P, SSpower3P, nls, selfStart

## Examples

```
library(ggtrendline)
x<-1:5
y<-c(2,4,8,20,25)
xy<-data.frame(x,y)
getInitial(y ~ SSpower3P(x,a,b,c), data = xy)
## Initial values are in fact the converged values

fitpower3P <- nls(y~SSpower3P(x,a,b,c), data=xy)
summary(fitpower3P)

prediction <- predFit(fitpower3P , data.frame(x=x), se.fit = TRUE,
                          level = 0.95, interval = "confidence")
yfitpower3P <- prediction$fit
yfitpower3P # output a matrix of predictions and bounds with column names fit, lwr, and upr.
```

---

`stat_eq` *Add Equation to 'ggplot'*

---

## Description

Add regression equation to 'ggplot', by using different models built in the 'ggtrendline()' function.
The function includes the following models:
"line2P" (formula as: y=a*x+b),
"line3P" (y=a*x^2+b*x+c),
"log2P" (y=a*ln(x)+b),
"exp2P" (y=a*exp(b*x)),
"exp3P" (y=a*exp(b*x)+c),
"power2P" (y=a*x^b),
and "power3P" (y=a*x^b+c).

## Usage

```
stat_eq(
  x,
  y,
  model = "line2P",
  show.eq = TRUE,
  xname = "x",
  yname = "y",
  yhat = FALSE,
  eq.x = NULL,
  eq.y = NULL,
  text.col = "black",
  eDigit = 3,
  eSize = 3
)
```

## Arguments

| | |
|---|---|
| x, y | the x and y arguments provide the x and y coordinates for the 'ggplot'. Any reasonable way of defining the coordinates is acceptable. |
| model | select which model to fit. Default is "line2P". The "model" should be one of c("line2P", "line3P", "log2P", "exp2P", "exp3P", "power2P", "power3P"), their formulas are as follows:<br>"line2P": y=a*x+b<br>"line3P": y=a*x^2+b*x+c<br>"log2P": y=a*ln(x)+b<br>"exp2P": y=a*exp(b*x)<br>"exp3P": y=a*exp(b*x)+c<br>"power2P": y=a*x^b<br>"power3P": y=a*x^b+c |
| show.eq | whether to show the regression equation, the value is one of c("TRUE", "FALSE"). |
| xname | to specify the expression of "x" in equation, i.e., expression('x'), see Examples. |
| yname | to specify the expression of "y" in equation, i.e., expression('y'), see Examples. |
| yhat | whether to add a hat symbol (^) on the top of "y" in equation. Default is FALSE. |
| eq.x, eq.y | equation position. |
| text.col | the color used for the equation text. |
| eDigit | the numbers of digits for equation parameters. Default is 3. |
| eSize | font size of equation. Default is 3. |

## Details

The values of each parameter of regression model can be found by typing `trendline_sum` function in this package.

The linear models (line2P, line3P, log2P) in this package are estimated by `lm` function, while the nonlinear models (exp2P, exp3P, power2P, power3P) are estimated by `nls` function (i.e., least-squares method).

## Value

No return value (called for side effects).

## See Also

`ggtrendline`, `stat_rrp`, `trendline_sum`

stat_rrp *Add R square and P-value to 'ggplot'*

## Description

Add R-square and P-value of regression models to 'ggplot', by using models built in the 'ggtrendline()' function. The function includes the following models:
"line2P" (formula as: y=a*x+b),
"line3P" (y=a*x^2+b*x+c),
"log2P" (y=a*ln(x)+b),
"exp2P" (y=a*exp(b*x)),
"exp3P" (y=a*exp(b*x)+c),
"power2P" (y=a*x^b),
and "power3P" (y=a*x^b+c).

## Usage

```
stat_rrp(
  x,
  y,
  model = "line2P",
  Pvalue.corrected = TRUE,
  show.Rsquare = TRUE,
  show.pvalue = TRUE,
  Rname = 0,
  Pname = 0,
  rrp.x = NULL,
  rrp.y = NULL,
  text.col = "black",
  eDigit = 3,
  eSize = 3
)
```

## Arguments

| | |
|---|---|
| x, y | the x and y arguments provide the x and y coordinates for the 'ggplot'. Any reasonable way of defining the coordinates is acceptable. |
| model | select which model to fit. Default is "line2P". The "model" should be one of c("line2P", "line3P", "log2P", "exp2P", "exp3P", "power2P", "power3P"), their formulas are as follows:<br>"line2P": y=a*x+b<br>"line3P": y=a*x^2+b*x+c<br>"log2P": y=a*ln(x)+b<br>"exp2P": y=a*exp(b*x)<br>"exp3P": y=a*exp(b*x)+c<br>"power2P": y=a*x^b<br>"power3P": y=a*x^b+c |

| | |
|---|---|
| Pvalue.corrected | if P-value corrected or not, the value is one of c("TRUE", "FALSE"). |
| show.Rsquare | whether to show the R-square, the value is one of c("TRUE", "FALSE"). |
| show.pvalue | whether to show the P-value, the value is one of c("TRUE", "FALSE"). |
| Rname | to specify the character of R-square, the value is one of c(0, 1), corresponding to c(r^2, R^2). |
| Pname | to specify the character of P-value, the value is one of c(0, 1), corresponding to c(p, P). |
| rrp.x, rrp.y | the position for R square and P value. |
| text.col | the color used for the equation text. |
| eDigit | the numbers of digits for R square and P value. Default is 3. |
| eSize | font size of R square and P value. Default is 3. |

## Details

The values of each parameter of regression model can be found by typing `trendline_sum` function in this package.

The linear models (line2P, line3P, log2P) in this package are estimated by `lm` function, while the nonlinear models (exp2P, exp3P, power2P, power3P) are estimated by `nls` function (i.e., least-squares method).

The argument 'Pvalue.corrected' is only valid for non-linear regression.

If "Pvalue.corrected = TRUE", the P-value is calculated by using "Residual Sum of Squares" and "Corrected Total Sum of Squares (i.e. sum((y-mean(y))^2))".

If "Pvalue.corrected = FALSE", the P-value is calculated by using "Residual Sum of Squares" and "Uncorrected Total Sum of Squares (i.e. sum(y^2))".

## Value

No return value (called for side effects).

## See Also

`ggtrendline`, `stat_eq`, `trendline_sum`

---

`trendline_sum` *Summarized Results of Each Regression Model*

---

## Description

Summarizing the results of linear or nonlinear regression model which built in the 'ggtrendline()' function. The function includes the following models:
"line2P" (formula as: y=a*x+b),
"line3P" (y=a*x^2+b*x+c),
"log2P" (y=a*ln(x)+b),
"exp2P" (y=a*exp(b*x)),
"exp3P" (y=a*exp(b*x)+c),
"power2P" (y=a*x^b),
and "power3P" (y=a*x^b+c).

## Usage

```
trendline_sum(
  x,
  y,
  model = "line2P",
  Pvalue.corrected = TRUE,
  summary = TRUE,
  eDigit = 5
)
```

## Arguments

| | |
|---|---|
| x, y | the x and y arguments provide the x and y coordinates for the 'ggplot'. Any reasonable way of defining the coordinates is acceptable. |
| model | select which model to fit. Default is "line2P". The "model" should be one of c("line2P", "line3P", "log2P", "exp2P", "exp3P", "power2P", "power3P"), their formulas are as follows:<br>"line2P": y=a*x+b<br>"line3P": y=a*x^2+b*x+c<br>"log2P": y=a*ln(x)+b<br>"exp2P": y=a*exp(b*x)<br>"exp3P": y=a*exp(b*x)+c<br>"power2P": y=a*x^b<br>"power3P": y=a*x^b+c |
| Pvalue.corrected | if P-value corrected or not, the value is one of c("TRUE", "FALSE"). |
| summary | summarizing the model fits. Default is TRUE. |
| eDigit | the numbers of digits for summarized results. Default is 3. |

## Details

The linear models (line2P, line3P, log2P) in this package are estimated by `lm` function,
while the nonlinear models (exp2P, exp3P, power2P, power3P) are estimated by `nls` function (i.e., least-squares method).

The argument 'Pvalue.corrected' is workful for non-linear regression only.

If "Pvalue.corrected = TRUE", the P-vlaue is calculated by using "Residual Sum of Squares" and "Corrected Total Sum of Squares (i.e. sum((y-mean(y))^2))".

If "Pvalue.corrected = TRUE", the P-vlaue is calculated by using "Residual Sum of Squares" and "Uncorrected Total Sum of Squares (i.e. sum(y^2))".

## Value

R^2, indicates the R-Squared value of each regression model.

p, indicates the p-value of each regression model.

N, indicates the sample size.

AIC, AICc, or BIC, indicate the Akaike's Information Criterion (AIC), the second-order AIC (AICc) for small samples, or Bayesian Information Criterion (BIC) for fitted model. Click AIC for details. The smaller the AIC, AICc or BIC, the better the model.

RSS, indicate the value of "Residual Sum of Squares".

## Note

If the output of 'AICc' is 'Inf', not an exact number, please try to expand the sample size of your dataset to >=6.

## See Also

ggtrendline, SSexp2P, SSexp3P, SSpower2P, SSpower3P, nls, selfStart, AICc

## Examples

```
library(ggtrendline)
x <- c(1, 3, 6, 9,  13,   17)
y <- c(5, 8, 11, 13, 13.2, 13.5)

trendline_sum(x, y, model="exp3P", summary=TRUE, eDigit=3)
```

# Index