

# Package ‘rAmCharts’

January 13, 2025

**Title** JavaScript Charts Tool

**Description** Provides an R interface for using 'AmCharts' Library. Based on 'htmlwidgets', it provides a global architecture to generate 'JavaScript' source code for charts. Most of classes in the library have their equivalent in R with S4 classes; for those classes, not all properties have been referenced but can easily be added in the constructors. Complex properties (e.g. 'JavaScript' object) can be passed as named list. See examples at [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and <https://www.amcharts.com/> for more information about the library. The package includes the free version of 'AmCharts' Library. Its only limitation is a small link to the web site displayed on your charts. If you enjoy this library, do not hesitate to refer to this page <https://www.amcharts.com/online-store/> to purchase a licence, and thus support its creators and get a period of Priority Support. See also <https://www.amcharts.com/about/> for more information about 'AmCharts' company.

**Version** 2.1.16

**Maintainer** Benoit Thieurmel <bthieurmel@gmail.com>

**License** GPL (>= 2)

**URL** [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/)

**BugReports** <https://github.com/datastorm-open/rAmCharts/issues/>

**Depends** R (>= 3.2.0)

**Collate** 'amChartsAPI.R' 'base\_data.R' 'base\_startupMessage.R'  
'chart\_amBarplot.R' 'chart\_amBoxplot.R' 'chart\_amBullet.R'  
'chart\_amCandlestick.R' 'chart\_amDataset.R'  
'chart\_amFloatingBar.R' 'chart\_amFunnel.R' 'chart\_amGauge.R'  
'chart\_amHist.R' 'chart\_amMekko.R' 'chart\_amOHLC.R'  
'chart\_amPie.R' 'chart\_amPlot.R' 'chart\_amRadar.R'  
'chart\_amTimeSeries.R' 'chart\_amWaterfall.R' 'chart\_amWind.R'  
'utils\_sharedGenerics.R' 'class\_AmObject.R' 'class\_AxisBase.R'  
'class\_ValueAxis.R' 'utils\_basicClassUnions.R'  
'class\_TrendLine.R' 'class\_Title.R' 'class\_AmChart.R'  
'class\_StockPanel.R' 'class\_StockEvent.R'  
'class\_PeriodSelector.R' 'class\_Label.R' 'class\_Guide.R'

'class\_GaugeAxis.R' 'class\_GaugeBand.R' 'class\_GaugeArrow.R'  
 'class\_DataSet.R' 'class\_ChartScrollbar.R'  
 'class\_ChartCursor.R' 'class\_AmLegend.R' 'class\_AmGraph.R'  
 'class\_AmBalloon.R' 'classUnions.R'  
 'class\_AmChart\_constructors.R' 'class\_CategoryAxis.R'  
 'class\_AmChart\_setters.R' 'class\_AmChart\_shinyUtils.R'  
 'class\_AmStockChart.R' 'class\_AmStockChart\_setters.R'  
 'class\_AxisBase\_setters.R' 'class\_ChartCursor\_setters.R'  
 'class\_ChartScrollbar\_setters.R' 'class\_DataSetSelector.R'  
 'class\_DataSet\_setters.R' 'class\_GaugeArrow\_setters.R'  
 'class\_GaugeAxis\_setters.R' 'class\_Guide\_setters.R'  
 'class\_StockEvent\_setters.R' 'class\_StockPanel\_setters.R'  
 'class\_TrendLine\_setters.R' 'shiny\_examples.R'  
 'shiny\_modules\_export.R' 'shiny\_modules\_timeSeries.R'  
 'union\_AmCharts.R' 'utils.R' 'utils\_amOptions.R'  
 'utils\_amTests.R'

**Imports** methods, htmlwidgets, htmltools, data.table (>= 1.9.6),  
 graphics, utils, pipeR, knitr, grDevices, yaml, zoo

**Suggests** magrittr, shiny, testthat, shinydashboard, base64enc,  
 lubridate, jsonlite

**RoxygenNote** 7.2.1

**LazyData** true

**NeedsCompilation** no

**Author** Benoit Thieurmel [aut, cre],  
 Antanas Marcelionis [aut, cph] ('AmCharts' library in th directory  
 htmlwidgets/lib/amcharts, refer to <https://www.amcharts.com/>)

**Repository** CRAN

**Date/Publication** 2025-01-13 11:00:02 UTC

## Contents

addGuide,AxisBase,GuideOrMissing-method . . . . .	5
addListener . . . . .	5
add_animate_dependency . . . . .	7
add_dataloader_dependency . . . . .	8
add_export_dependency . . . . .	8
add_responsive_dependency . . . . .	9
add_theme_dependency . . . . .	9
amAngularGauge . . . . .	10
AmBalloon-class . . . . .	11
amBarplot . . . . .	12
amBoxplot . . . . .	15
amBullet . . . . .	18
amCandlestick . . . . .	20
AmChart-class . . . . .	22

amChartsAPI . . . . .	24
amChartsOutput . . . . .	24
amFloatingBar . . . . .	25
amFunnel . . . . .	27
AmGraph-class . . . . .	29
amHist . . . . .	30
AmLegend-class . . . . .	32
amLines . . . . .	32
amMekko . . . . .	34
AmObject-class . . . . .	35
amOHLC . . . . .	35
amOptions . . . . .	37
amPie . . . . .	40
amPlot . . . . .	41
amRadar . . . . .	45
amSolidGauge . . . . .	46
AmStockChart-class . . . . .	48
amStockMultiSet . . . . .	49
amTimeSeries . . . . .	51
amWaterfall . . . . .	56
amWind . . . . .	57
api . . . . .	58
AxisBase-class . . . . .	59
CategoryAxis-class . . . . .	59
ChartCursor-class . . . . .	60
ChartScrollbar-class . . . . .	61
controlShinyPlot . . . . .	61
DataSet-class . . . . .	62
DataSetSelector-class . . . . .	62
data_AirPassengers . . . . .	63
data_bar . . . . .	63
data_candleStick1 . . . . .	64
data_candleStick2 . . . . .	64
data_fbar . . . . .	65
data_funnel . . . . .	65
data_gantt . . . . .	66
data_gbar . . . . .	66
data_gdp . . . . .	67
data_mekko . . . . .	67
data_pie . . . . .	68
data_radar . . . . .	68
data_stock1 . . . . .	69
data_stock_2 . . . . .	69
data_stock_3 . . . . .	70
data_waterfall . . . . .	70
data_wind . . . . .	71
GaugeArrow-class . . . . .	71
GaugeAxis-class . . . . .	72

GaugeBand-class . . . . .	72
Generics functions . . . . .	73
getCurrentStockData . . . . .	74
getTransformTS . . . . .	75
Guide-class . . . . .	77
initialize,AmBalloon-method . . . . .	77
initialize,AmChart-method . . . . .	79
initialize,AmGraph-method . . . . .	92
initialize,AmLegend-method . . . . .	94
initialize,AmStockChart-method . . . . .	95
initialize,CategoryAxis-method . . . . .	101
initialize,ChartCursor-method . . . . .	102
initialize,ChartScrollbar-method . . . . .	103
initialize,DataSet-method . . . . .	104
initialize,DataSetSelector-method . . . . .	106
initialize,GaugeArrow-method . . . . .	107
initialize,GaugeAxis-method . . . . .	108
initialize,GaugeBand-method . . . . .	109
initialize,Guide-method . . . . .	110
initialize,Label-method . . . . .	111
initialize,PeriodSelector-method . . . . .	113
initialize,StockEvent-method . . . . .	114
initialize,StockPanel-method . . . . .	115
initialize,Title-method . . . . .	118
initialize,TrendLine-method . . . . .	119
initialize,ValueAxis-method . . . . .	121
Label-class . . . . .	122
listProperties . . . . .	123
PeriodSelector-class . . . . .	123
plot,AmCharts-method . . . . .	124
print,AmObject-method . . . . .	124
rAmCharts-shinymodules . . . . .	125
rAmCharts-shinymodules-ts . . . . .	126
renderAmCharts . . . . .	131
runExamples . . . . .	131
setExport . . . . .	132
show,AmChart-method . . . . .	133
show,AmObject-method . . . . .	133
show,AmStockChart-method . . . . .	134
StockEvent-class . . . . .	134
stockGraph . . . . .	135
stockLegend . . . . .	135
StockPanel-class . . . . .	136
Title-class . . . . .	138
TrendLine-class . . . . .	138
ValueAxis-class . . . . .	139

---

addGuide,AxisBase,GuideOrMissing-method  
*Add a Guide for AxisBase*

---

### Description

Update an object of inherited class [AxisBase](#).

### Usage

```
## S4 method for signature 'AxisBase,GuideOrMissing'
addGuide(.Object, guide = NULL, ...)
```

### Arguments

.Object	children class of <a href="#">AxisBase</a> .
guide	(optional) <a href="#">Guide</a> .
...	properties of <a href="#">Guide</a> Argument for method addGuide.

### Value

(possibly updated) .Object of class [AxisBase](#).

### Examples

```
addGuide(.Object = valueAxis(), fillAlpha = .4, adjustBorderColor = TRUE, gridThickness = 1)
# equivalent to:
guide_obj <- guide(fillAlpha = .4, adjustBorderColor = TRUE, gridThickness = 1)
addGuide(.Object = valueAxis(), guide = guide_obj)
```

---

addListener *AmObject methods*

---

### Description

Methods for inherited classes.

**Usage**

```

addListener(.Object, name, expression)

## S4 method for signature 'AmObject,character,character'
addListener(.Object, name, expression)

resetProperties(.Object, ...)

## S4 method for signature 'AmObject'
resetProperties(.Object, ...)

setProperties(.Object, list_prop, ...)

## S4 method for signature 'AmObject'
setProperties(.Object, list_prop, ...)

```

**Arguments**

.Object	<a href="#">AmObject</a> .
name	character, name of the listener.
expression	character, associated function event.
...	other properties
list_prop	(Optional) list containing properties to set. The former properties will be overwritten.

**Details**

Former properties will be overwritten.

Former properties will be overwritten.

**Value**

The updated object.

**Examples**

```

addListener(.Object = amPieChart(),
            name = "clickSlice" ,
            expression = "function(event){ alert('ok !'); }")

addListener(.Object = amLegend(),
            name = "select",
            expression = paste0("function onSelect (properties) {",
                                "alert('selected nodes: ' + properties.nodes);",
                                "}")

library(pipeR)
amPlot(runif(10)) %>>% resetProperties("categoryAxis") %>>% print(withDetail = FALSE)
library(pipeR)

```

```
# either you can set a list
ls <- list(categoryAxis = list(gridPosition = "start"), fontSize = 15)
amSerialChart() %>>% setProperties(list = ls) %>>% print()

# or you can set one or more properties
amPieChart() %>>% setProperties(handDrawn = TRUE, fontSize = 15) %>>% print()

# overwrite a property
amPieChart() %>>% setProperties(fontSize = 15) %>>% setProperties(fontSize = 12) %>>% print()

# Carefull if you try to set a property which is a slot...
# in that case, use the setter methods 'setXX' or 'addXX' which check the validity
## Not run:
amPieChart() %>>% setProperties(type = "serial") %>>% print()

## End(Not run)

amPieChart() %>>% setExport()
```

---

add\_animate\_dependency

*Add animate dependency*

---

## Description

Add the 'animate' dependency to an htmlwidget. You can only manipulate the htmlwidget if you call the method 'plot' with an rAmChart.

## Usage

```
add_animate_dependency(widget)
```

## Arguments

widget            An htmlwidget.

## Value

Return an updated htmlwidget with the dependency.

add\_dataloader\_dependency

*Add dataloader dependency*

---

### **Description**

Add the 'dataloader' dependency to an htmlwidget. You can only manipulate the htmlwidget if you call the method 'plot' with an rAmChart.

### **Usage**

```
add_dataloader_dependency(widget)
```

### **Arguments**

widget            An htmlwidget

### **Value**

Return the updated htmlwidget.

---

add\_export\_dependency *Add the export dependency to an htmlwidget*

---

### **Description**

Add the 'export' dependency to an htmlwidget. You can only manipulate the htmlwidget if you call the method 'plot' with an rAmChart.

### **Usage**

```
add_export_dependency(widget)
```

### **Arguments**

widget            An htmlwidget.

### **Value**

Return the updated widget with the 'export' dependency.



---

add\_responsive\_dependency      *Add responsive dependency*

---

**Description**

Add the 'responsive' dependency to an htmlwidget. You can only manipulate the htmlwidget if you call the method 'plot' with an rAmChart.

**Usage**

```
add_responsive_dependency(widget)
```

**Arguments**

widget                      An htmlwidget.

**Value**

Return an updated htmlwidget with the dependency.

---

add\_theme\_dependency      *Add theme dependency*

---

**Description**

Add the 'theme' dependency to an htmlwidget. You can only manipulate the htmlwidget if you call the method 'plot' with an rAmChart.

**Usage**

```
add_theme_dependency(  
  widget,  
  theme_js = c("light.js", "patterns.js", "dark.js", "chalk.js")  
)
```

**Arguments**

widget                      An htmlwidget.  
theme\_js                    A character indicating the JS file dependency.

**Value**

Return the updated htmlwidget.

**Examples**

```
## Not run:
library(pipeR)
amPlot(1:10, theme = "dark") %>% plot() %>% add_theme_dependency("light.js")

## End(Not run)
```

amAngularGauge

*Plotting gauge using rAmCharts***Description**

amAngularGauge computes a gauge of the given value.

**Usage**

```
amAngularGauge(
  x,
  start = 0,
  end = 100,
  step = 20,
  bands = data.frame(start = numeric(), end = numeric(), color = character(), width =
    numeric(), stringsAsFactors = FALSE),
  text = "",
  textSize = 25,
  secondAxe = FALSE,
  start2 = 0,
  end2 = 100,
  step2 = 20,
  bands2 = data.frame(start = numeric(), end = numeric(), color = character(),
    stringsAsFactors = FALSE),
  ...
)
```

**Arguments**

x	numeric, value for which the angular gauge is desired.
start	numeric, minimum value allowed.
end	numeric, maximum value allowed.
step	numeric, intervals size.
bands	data.frame with 4 columns : start (numeric, minimal value for the band), end (numeric, maximal value for the band), color (character, color of the band, in hexadecimal) and width (numeric, width of the band). If the last column is not defined, it is automatically set to 10.
text	character, text legend.

textSize	numeric, text size.
secondAxe	logical, TRUE if two axes are desired. Default is set to FALSE.
start2	numeric, minimum value allowed for the second axe if secondAxe is TRUE.
end2	numeric, maximum value allowed for the second axe if secondAxe is TRUE.
step2	numeric, intervals size for the second axe if secondAxe is TRUE.
bands2	data.frame with 4 columns : start (numeric, minimal value for the band), end (numeric, maximal value for the band), color (character, color of the band, in hexadecimal) and width (numeric, width of the band). If the last column is not defined, it is automatically set to 10.
...	see <a href="#">amOptions</a> for more options.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
amAngularGauge(10)
```

---

AmBalloon-class

*AmBalloon class*

---

## Description

Creates the balloons (tooltips) of the chart. It follows the mouse cursor when you roll-over the data items. The framework generates the instances automatically, you just have to adjust the appearance to your needs.

## Details

Run `api("AmBalloon")` for more information and all available properties.

**Slots**

`adjustBorderColor` logical. If TRUE, border color will be changed when user rolls-over the slice, graph, etc, instead of background color.

`color` character. Balloon text color.

`cornerRadius` numeric. Balloon corner radius.

`fillColor` character. Balloon background color. It is usually defined by the chart itself. If "adjustBorderColor" is set to TRUE, the balloon background color will be equal to "fillColor".

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. See examples for details. Inherited from [AmObject](#).

`otherProperties` list containing other available properties not yet implemented in the package. Inherited from [AmObject](#).

`value` numeric. Inherited from [AmObject](#).

---

 amBarplot

*Plotting bar chart using rAmCharts*


---

**Description**

amBarplot computes a bar chart of the given values.

**Usage**

```
amBarplot(
  x,
  y,
  data,
  xlab = "",
  ylab = "",
  ylim = NULL,
  groups_color = NULL,
  horiz = FALSE,
  stack_type = c("none", "regular", "100"),
  layered = FALSE,
  show_values = FALSE,
  depth = 0,
  dataDateFormat = NULL,
  minPeriod = ifelse(!is.null(dataDateFormat), "DD", ""),
  ...
)
```

**Arguments**

x	character, column name for x-axis or numeric, value of the corresponding column. It is optional if argument data has row names.
y	character, column name for y-axis or numeric vector of the corresponding column. If you want to display a grouped barchart or a stacked barchart, y is a vector of characters or numerics.
data	data.frame, dataframe with values to display. You can add a column "color" (character, colors in hexadecimal). You can also add a column "description" (character) containing the text you want to display when mouse is on the graphic (' ' for a new line). See <a href="#">data_bar</a> and <a href="#">data_gbar</a> .
xlab	character, label for x-axis.
ylab	character, label for y-axis.
ylim	limits for the y axis.
groups_color	character, vector of colors in hexadecimal, same length as y.
horiz	logical, TRUE for an horizontal chart, FALSE for a vertical one. If 'horiz' is set to TRUE, the setting 'labelRotation' will be ignored.
stack_type	character, "regular" if you wish stacked bars, "100" if you want 100 percent stacked bars. Default is set to "none".
layered	logical, TRUE for layered bars. If TRUE, stack_type must be set to "none".
show_values	logical, TRUE to display values.
depth	numeric, if > 0, chart is displayed in 3D. Value between 0 and 100.
dataDateFormat	character, default set to NULL. Even if your chart parses dates, you can pass them as strings in your dataframe - all you need to do is to set data date format and the chart will parse dates to date objects. Check this page for available formats. Please note that two-digit years (YY) as well as literal month names (MMM) are NOT supported in this setting.
minPeriod	Specifies the shortest period of your data. This should be set only if dataDateFormat is not 'NULL'. Possible period values: fff - milliseconds, ss - seconds, mm - minutes, hh - hours, DD - days, MM - months, YYYY - years. It's also possible to supply a number for increments, i.e. '15mm' which will instruct the chart that your data is supplied in 15 minute increments.
...	see <a href="#">amOptions</a> for more options.

**Details**

**Notice about labels:** if the chart has many columns, several labels might be hidden. It depends on the width of the container where the chart is displayed. Zoom on the chart to see if the chart can contain all labels. If not, use the parameter labelRotation. You can also add a cursor to your chart...

**Value**

An object of class [AmChart](#).

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
# Data
data(data_bar)
data(data_gbar)

amBarplot(x = "country", y = "visits", data = data_bar, main = "example")

## Not run:

# Other examples available which can be time consuming depending on your configuration.

# fixed value axis
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar, ylim = c(0, 26))
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar, stack_type = "100")

# Test with label rotation
amBarplot(x = "country", y = "visits", data = data_bar, labelRotation = -45)

# Horizontal bar
amBarplot(x = "country", y = "visits", data = data_bar, horiz = TRUE, labelRotation = -45)

# 3D bar
amBarplot(x = "country", y = "visits", data = data_bar, depth = 15, labelRotation = -45)

# Display values
amBarplot(x = "country", y = "visits", data = data_bar, show_values = TRUE, labelRotation = -45)

# Grouped columns
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar)

# Parse dates
# Default label: first day of each year
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar,
          dataDateFormat = "YYYY", minPeriod = "YYYY")

# Default label: first day of each month
amBarplot(x = "month", y = c("income", "expenses"), data = data_gbar,
          dataDateFormat = "MM/YYYY", minPeriod = "MM")

amBarplot(x = "day", y = c("income", "expenses"), data = data_gbar,
          dataDateFormat = "DD/MM/YYYY")
```

```

# Change groups colors
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar,
          groups_color = c("#87cefa", "#c7158"))

# Regular stacked bars
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar, stack_type = "regular")

# 100% stacked bars
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar, stack_type = "100")

# Layered bars
amBarplot(x = "year", y = c("income", "expenses"), data = data_gbar, layered = TRUE)

# Data with row names
dataset <- data.frame(get(x = "USArrests", pos = "package:datasets"))
amBarplot(y = c("Murder", "Assault", "UrbanPop", "Rape"), data = dataset, stack_type = "regular")

# Round values
amBarplot(x = "year", y = c("in", "ex"), data = data_gbar, precision = 0)

## End(Not run)

```

---

amBoxplot

*Plotting boxplot using rAmCharts*


---

## Description

amBoxplot computes a boxplot of the given data values. Can be a vector, a data.frame, or a matrix.

## Usage

```

amBoxplot(object, ...)

## Default S3 method:
amBoxplot(
  object,
  xlab = NULL,
  ylab = NULL,
  ylim = NULL,
  names = NULL,
  col = "#1e90ff",
  horiz = FALSE,
  ...
)

## S3 method for class 'data.frame'
amBoxplot(

```

```

    object,
    id = NULL,
    xlab = NULL,
    ylab = NULL,
    ylim = NULL,
    col = NULL,
    horiz = FALSE,
    ...
)

## S3 method for class 'matrix'
amBoxplot(
  object,
  use.cols = TRUE,
  xlab = NULL,
  ylab = NULL,
  ylim = NULL,
  col = NULL,
  horiz = FALSE,
  ...
)

## S3 method for class 'formula'
amBoxplot(
  object,
  data = NULL,
  id = NULL,
  xlab = NULL,
  ylab = NULL,
  ylim = NULL,
  col = NULL,
  horiz = FALSE,
  ...
)

```

### Arguments

<code>object</code>	a vector, data.frame, a matrix, or a formula.
<code>...</code>	see <a href="#">amOptions</a> for more options.
<code>xlab, ylab</code>	character, labels of the axis.
<code>ylim</code>	numeric, y values range with sensible defaults.
<code>names</code>	character, name on x-axis, if object is a vector.
<code>col</code>	character, color(s) to be used to fill the boxplot.
<code>horiz</code>	logical, TRUE to rotate chart.
<code>id</code>	character, column name of id to identify outliers, if object is a dataframe.
<code>use.cols</code>	logical, for matrix only. Set to TRUE to display boxplot based on columns.
<code>data</code>	data.frame, from which the variables in formula should be taken.



**Value**

An object of class [AmChart](#).

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
## Not run:
# 'numeric' (default)
amBoxplot(rnorm(100))

# 'formula'
amBoxplot(count ~ spray, data = InsectSprays)

# 'formula', two group
data <- InsectSprays
data$group <- c("H", "F")
amBoxplot(count ~ spray + group, data = data, col = c("purple", "darkblue"))

# 'matrix'
x <- matrix(nrow = 10, ncol = 5, rnorm(50))
amBoxplot(x)

# 'data.frame'
amBoxplot(iris[, 1:4])

## End(Not run)
# Other examples available which can be time consuming depending on your configuration.

## Not run:
don <- data.frame(a = 1:10, b = 1:5)
amBoxplot(don, ylim = c(0,15))

## End(Not run)
## Not run:
# --- matrix
x <- matrix(nrow = 10, ncol = 5, rnorm(50))

amBoxplot(x) # on columns
colnames(x) <- LETTERS[1:5]
amBoxplot(x) # with names
amBoxplot(x, use.cols = FALSE, col = c("blue", "red"))
```

```

# Parameter for amOptions
amBoxplot(x, export = TRUE, exportFormat = "SVG")

## End(Not run)
## Not run:
# --- Formula
(obj <- amBoxplot(count ~ spray, data = InsectSprays))

# Adding parameters
amBoxplot(count ~ spray, data = InsectSprays, ylim = c(0,50),
          xlab = "spray", col = c("darkblue", "gray"))

# Transpose
amBoxplot(count ~ spray, data = InsectSprays, ylim = c(0,50), xlab = "spray", horiz = FALSE)

# Using a custom colum to identify outliers
InsectSprays$id <- paste0("ID : ", 1:nrow(InsectSprays))
amBoxplot(count ~ spray, data = InsectSprays, id = "id")

# Parameter for amOptions
amBoxplot(count ~ spray, data = InsectSprays, main = "amcharts")

## End(Not run)

```

---

amBullet

*Plotting bullet chart using rAmCharts*


---

## Description

amBullet computes a bullet chart of the given value.

## Usage

```

amBullet(
  value,
  min = 0,
  max = 100,
  val_color = "#000000",
  limit = 85,
  limit_color = "#000000",
  steps = TRUE,
  label = "",
  horiz = TRUE,
  rates,
  ...
)

```

**Arguments**

value	numeric, value to display.
min	numeric, minimum value allowed.
max	numeric, maximum value allowed.
val_color	character, color of the bar value, in hexadecimal.
limit	numeric, target value.
limit_color	character, color of the target line.
steps	logical, default set to TRUE.
label	character, label of the bullet.
horiz	logical, TRUE (default) for an horizontal bullet chart, FALSE for a vertical one.
rates	data.frame with 4 columns: name (character), min (numeric), max (numeric), and color (character, color in hexadecimal).
...	see <a href="#">amOptions</a> for more options.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
amBullet(value = 65)

## Not run:
# Other examples available which can be time consuming depending on your configuration.

# Remove steps for background
amBullet(value = 65, steps = FALSE)

# Tune the colors with name or HTML code
amBullet(value = 65, val_color = "purple", limit_color = "#3c8dbc")

# Change the orientation
amBullet(value = 65, steps = FALSE, horiz = FALSE)

# Add text
amBullet(value = 65, label = "Evaluation")

# Change min and max values
amBullet(value = 65, min = 20, max = 90)
```

```
## End(Not run)
```

---

amCandlestick

*Plotting candlestick chart using rAmCharts*


---

## Description

amCandlestick computes a candlestick chart of the given value.

## Usage

```
amCandlestick(
  data,
  xlab = "",
  ylab = "",
  horiz = FALSE,
  positiveColor = "#7f8da9",
  negativeColor = "#db4c3c",
  names = c("low", "open", "close", "high"),
  dataDateFormat = NULL,
  minPeriod = ifelse(!is.null(dataDateFormat), "DD", ""),
  ...
)
```

## Arguments

data	data.frame, dataframe with at least 5 columns: category, open (numeric), close (numeric), low (numeric), high (numeric). See <a href="#">data_candleStick1</a> and <a href="#">data_candleStick2</a> .
xlab	character, label for x-axis.
ylab	character, label for y-axis.
horiz	logical, TRUE for an horizontal chart, FALSE for a vertical one
positiveColor	character, color for positive values (in hexadecimal).
negativeColor	character, color for negative values (in hexadecimal).
names	character, names for the tooltip. Default set to c("low", "open", "close", "high").
dataDateFormat	character, default set to NULL. Even if your chart parses dates, you can pass them as strings in your dataframe - all you need to do is to set data date format and the chart will parse dates to date objects. Check this page for available formats. Please note that two-digit years (YY) as well as literal month names (MMM) are NOT supported in this setting.

minPeriod character, minPeriod Specifies the shortest period of your data. This should be set only if dataDateFormat is not NULL. Possible period values: fff - milliseconds, ss - seconds, mm - minutes, hh - hours, DD - days, MM - months, YYYY - years. It's also possible to supply a number for increments, i.e. '15mm' which will instruct the chart that your data is supplied in 15 minute increments.

... see [amOptions](#) for more options.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
data("data_candleStick2")
amCandlestick(data = data_candleStick2)

## Not run:
# Change colors
amCandlestick(data = data_candleStick2, positiveColor = "black", negativeColor = "green")

# Naming the axes
amCandlestick(data = data_candleStick2, xlab = "categories", ylab = "values")

# Rotate the labels for x axis
amCandlestick(data = data_candleStick2, labelRotation = 90)

# Change names
amCandlestick(data = data_candleStick2, names = c("min", "begin", "end", "max"))

# Horizontal chart :
amCandlestick(data = data_candleStick2, horiz = TRUE)

# Parse date
amCandlestick(data = data_candleStick2, dataDateFormat = "YYYY-MM-DD")

# Datas over months
data_candleStick2$category <- c("2015-01-01", "2015-02-01", "2015-03-01",
                               "2015-04-01", "2015-05-01", "2015-06-01",
                               "2015-07-01", "2015-08-01", "2015-09-01",
                               "2015-10-01", "2015-11-01", "2015-12-01")

amCandlestick(data = data_candleStick2, dataDateFormat = "YYYY-MM-DD", minPeriod = "MM")

# Decimal precision
```

```

if (requireNamespace("pipeR", quietly = TRUE)) {
  require(pipeR)

  amCandlestick(data = data_candleStick2, horiz = TRUE) %>>%
    setProperties(precision = 2)
}

## End(Not run)

```

---

AmChart-class

*AmChart*


---

## Description

Defines the AmChart properties.

## Details

API for plotting AmChart with R.

## Slots

**allLabels** list of [Label](#). Example of a label object, with all possible properties: `label(x = 20, y = 20, text = "this is a label", align = "left", size = 12, color = "#CC0000", alpha = 1, rotation = 0, bold = TRUE, url = "http://www.amcharts.com")`. Run `api("Label")` for more informations.

**arrows** list of [GaugeArrow](#). Only valid for gauge charts. Run `api("GaugeArrow")` for more informations.

**axes** list of [GaugeAxis](#) properties. Only valid for gauge charts. Run `api("GaugeAxis")` for more informations.

**balloon** [AmBalloon](#). Creates the balloons (tooltips) of the chart, It follows the mouse cursor when you roll-over the data items. The framework generates the instances automatically you just have to adjust the appearance to your needs. Run `api("AmBalloon")` for more informations.

**categoryAxis** [CategoryAxis](#). Read-only. Chart creates category axis itself. If you want to change some properties, you should get this axis from the chart and set properties to this object. Run `api("CategoryAxis")` for more informations.

**categoryField** character. Category field name indicates the name of the field in your dataProvider object which will be used for category axis values.

**ChartCursor** [ChartCursor](#). Chart's cursor. Run `api("ChartCursor")` for more informations.

**ChartScrollbar** [ChartScrollbar](#). Chart's scrollbar. Run `api("ChartScrollbar")` for more informations.

**creditsPosition** character, specifies position of the amCharts' website link. Allowed values are: "top-left", "top-right", "bottom-left" and "bottom-right".

**dataProvider** `data.frame`, containing the data.

- `graphs` list of [AmGraph](#). Creates the visualization of the data in following types: line, column, step line, smoothed line, olhc and candlestick. Run `api("AmGraph")` for more informations.
- `graph` [AmGraph](#). Only valid for Gantt charts. Gant chart actually creates multiple graphs (separate for each segment). Properties of this graph are passed to each of the created graphs - this allows you to control the look of segments. Run `api("AmGraph")` for more informations.
- `guides` list of [Guide](#). Instead of adding guides to the axes, you can push all of them to this array. In case guide has category or date defined, it will automatically be assigned to the category axis, otherwise to the first value axis, unless you specify a different `valueAxes` for the guide. Run `api("Guide")` for more informations.
- `legend` [AmLegend](#). Legend of a chart. Run `api("AmLegend")` for more informations.
- `segmentsField` character. Segments field in your data provider. Only valid for Gantt Charts.
- `subChartProperties` list. Only valid for Drilldown charts.
- `theme` character. Theme of a chart. Config files of themes can be found in `amcharts/themes/` folder. See <https://www.amcharts.com/docs/v3/tutorials/working-with-themes/>.
- `titles` list of [Title](#). Run `api("Title")` for more informations.
- `trendLines` list of [TrendLine](#) objects added to a chart. You can add trend lines to a chart using this list or access already existing trend lines. Run `api("TrendLine")` for more informations.
- `type` character. Possible types are: "serial", "pie", "radar", "xy", "radar", "funnel", "gauge", "stock". See details about using argument type. (type map is in development).
- `valueAxes` list of [ValueAxis](#). Chart creates one value axis automatically, so if you need only one value axis, you don't need to create it. Run `api("ValueAxis")` for more informations.
- `valueAxis` [ValueAxis](#). Only valid for Gantt Charts. Set it's type to "date" if your data is date or time based. Run `api("ValueAxis")` for more informations.
- `valueScrollbar` [ChartScrollbar](#). Value scrollbar, enables scrolling value axes.
- `listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. Run `runShinyExamples()` for examples.
- `otherProperties` list containing other available properties not yet implemented in the package.
- `value` numeric.

### See Also

<https://docs.amcharts.com/3/javascriptcharts/>

### Examples

```
# Run runShinyExamples() for examples.
```

---

amChartsAPI	<i>View full API of amcharts.js</i>
-------------	-------------------------------------

---

**Description**

View full API of amcharts.js

**Usage**

```
amChartsAPI(viewer = getOption("viewer"))
```

**Arguments**

viewer : Set to NULL to open in a browser

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/)

**Examples**

```
## Not run:  
amChartsAPI()  
amChartsAPI(NULL)  
  
## End(Not run)
```

---

amChartsOutput	<i>SHINY</i>
----------------	--------------

---

**Description**

Widget output function for use in Shiny

**Usage**

```
amChartsOutput(outputId, type = NULL, width = "100%", height = "400px")
```

**Arguments**

outputId	character, output variable to read the chart from.
type	character, indicating the chart type.
width	character, the width of the chart container.
height	character, the height of the chart container.



amFloatingBar

*Plotting floating bar chart using rAmCharts***Description**

amFloatingBar computes a floating bar chart of the given values.

**Usage**

```
amFloatingBar(
  x,
  y_inf,
  y_sup,
  data,
  xlab = "",
  ylab = "",
  groups_color = NULL,
  horiz = FALSE,
  show_values = FALSE,
  depth = 0,
  dataDateFormat = NULL,
  minPeriod = ifelse(!is.null(dataDateFormat), "DD", ""),
  ...
)
```

**Arguments**

x	character, column name for x-axis or numeric value of the corresponding column. It is optional if argument data has row names.
y_inf	character, column name for the lower value or numeric vector of the corresponding column.
y_sup	character, column name for the upper value or numeric vector of the corresponding column.
data	data.frame, dataframe with values to display. You can add a column "color" (character, colors in hexadecimal). You can also add a column "description" (character) containing the text you want to display when mouse is on the graphic (' ' for a new line). See <a href="#">data_fbar</a> .
xlab	character, label for x-axis.
ylab	character, label for y-axis.
groups_color	character, vector of colors in hexadecimal, same length as y_inf or y_sup.
horiz	logical, TRUE for an horizontal chart, FALSE for a vertical one. If 'horiz' is set to TRUE, the setting 'labelRotation' will be ignored.
show_values	logical, TRUE to display values.
depth	numeric, if > 0, chart is displayed in 3D. Value between 0 and 100.

dataDateFormat	character, default set to NULL. Even if your chart parses dates, you can pass them as strings in your dataframe - all you need to do is to set data date format and the chart will parse dates to date objects. Check this page for available formats. Please note that two-digit years (YY) as well as literal month names (MMM) are NOT supported in this setting.
minPeriod	Specifies the shortest period of your data. This should be set only if dataDateFormat is not 'NULL'. Possible period values: fff - milliseconds, ss - seconds, mm - minutes, hh - hours, DD - days, MM - months, YYYY - years. It's also possible to supply a number for increments, i.e. '15mm' which will instruct the chart that your data is supplied in 15 minute increments.
...	see <a href="#">amOptions</a> for more options.

### Details

**Notice about labels:** if the chart has many columns, several labels might be hidden. It depends on the width of the container where the chart is displayed. Zoom on the chart to see if the chart can contain all labels. You can also add a cursor to your chart...

### Value

An object of class [AmChart](#).

### References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

### See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

### Examples

```
# Load data
data(data_fbar)
data(data_gbar)

amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
              data = data_fbar, labelRotation = -45)

amFloatingBar(x = "year", y_inf = "expenses", y_sup = "income", data = data_gbar,
              dataDateFormat = "YYYY", minPeriod = "YYYY", zoom = TRUE)

## Not run:
# Other examples available which can be time consuming depending on your configuration.
library(pipeR)

# Reference example : column chart
```

```

amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
             data = data_fbar, labelRotation = -45)

# Label rotation modification
amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
             data = data_fbar, labelRotation = -90)

# Horizontal bar
amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
             data = data_fbar, horiz = TRUE)

# 3D bar
amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
             data = data_fbar, labelRotation = -45, depth = 15)

# Display values
amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
             data = data_fbar, labelRotation = -90, show_values = TRUE)

# Change colors
amFloatingBar(x = "country", y_inf = "visits_inf", y_sup = "visits_sup",
             data = data_fbar[,1:3], labelRotation = -45, groups_color = "#67b7dc")

# Grouped columns
# Parse dates

# Default label: first day of each year

amFloatingBar(x = "year", y_inf = "expenses", y_sup = "income", data = data_gbar,
             dataDateFormat = "YYYY", minPeriod = "YYYY", zoom = TRUE)

# Default label: first day of each month

amFloatingBar(x = "month", y_inf = "expenses", y_sup = "income", data = data_gbar,
             dataDateFormat = "MM/YYYY", minPeriod = "MM", zoom = TRUE)

amFloatingBar(x = "day", y_inf = "expenses", y_sup = "income", data = data_gbar,
             dataDateFormat = "DD/MM/YYYY", zoom = TRUE)

## End(Not run)

```

---

amFunnel

*Plotting funnel chart using rAmCharts*


---

### Description

amFunnel computes a funnel chart of the given value.

**Usage**

```
amFunnel(
  data,
  inverse = FALSE,
  neck_height = NULL,
  neck_width = NULL,
  depth = 0,
  label_side = "right",
  margin_right = 200,
  margin_left = 200,
  ...
)
```

**Arguments**

data	data.frame of at least 2 columns : value (numeric, positive), and description (character). You can add a third column "color" (character, colors in hexadecimal) see <a href="#">data_funnel</a> .
inverse	logical, if TRUE, the funnel chart will be inverted.
neck_height	numeric, value between 0 and 100 : if a bottleneck is desired, this value determines its heigh. Default to NULL.
neck_width	numeric, value between 0 and 100 : if a bottleneck is desired, this value determines its width. Default to NULL.
depth	numeric, if > 0, chart is displayed in 3D, only for pyramid chart (without a bottleneck). Value between 0 and 100.
label_side	character, label position : "right" or "left".
margin_right	numeric, margin at the right side.
margin_left	numeric, margin at the left side.
...	see <a href="#">amOptions</a> for more options.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
data(data_funnel)
amFunnel(data = data_funnel, inverse = TRUE)

## Not run:
```

```

# Other examples available which can be time consuming depending on your configuration.

# Change the orientation and legend side
amFunnel(data = data_funnel, inverse = FALSE,
          label_side = "left", margin_right = 15, margin_left = 160)

# Basic example : Funnel chart
amFunnel(data = data_funnel, neck_height = 30, neck_width = 40)

# 3D pyramid
amFunnel(data = data_funnel, depth = 50, inverse = TRUE)

## End(Not run)

```

---

AmGraph-class

*AmGraph class*


---

## Description

Creates the visualization of the data in following types: line, column, step line, smoothed line, ohlc and candlestick.

## Details

Run `api("AmGraph")` for more details and all available properties.

## Slots

`balloonText` character. Balloon text. You can use tags like `[[value]]`, `[[description]]`, `[[percents]]`, `[[open]]`, `[[category]]` or any other field name from your data provider. HTML tags can also be used.

`title` character. Graph title.

`type` character. Type of the graph. Possible values are: "line", "column", "step", "smoothed-Line", "candlestick", "ohlc". XY and Radar charts can only display "line" otherArguments graphs.

`valueField` character. Name of the value field in your dataProvider.

`listeners` "list" containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. See examples for details.

`otherProperties` "list" containing other available properties not yet implemented in the package.

`value` numeric.

---

amHist *Plotting histogram*

---

### Description

amHist computes a histogram of the given data values.

### Usage

```
amHist(x, ...)

## S3 method for class 'numeric'
amHist(
  x,
  col = "#1e90ff",
  border = "#1e90ff",
  freq = TRUE,
  plot = TRUE,
  labels = FALSE,
  xlab,
  ylab,
  ylim,
  control_hist,
  ...
)
```

### Arguments

x	numeric, a vector of values for which the histogram is desired.
...	see <a href="#">amOptions</a> for more options.
col	character, a color to be used to fill the bars.
border	character, a color for the borders.
freq	logical, if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE if and only if breaks are equidistant (and probability is not specified).
plot	logical, if TRUE (default), an histogram is plotted, otherwise a list of breaks and counts is returned. In the second case, a warning is used if (typically graphical) arguments are specified that only apply to the plot = TRUE case.
labels	logical, set to TRUE to display labels. Default set to FALSE. Additionally draw labels on top of bars. if TRUE, draw the counts or rounded densities; if labels is a character, draw itself.
xlab, ylab	character, labels of the axis.
ylim	numeric, the range of y values with sensible defaults.
control_hist	(optional) named list() containing parameters to compute the histogram.

**Value**

An object of class `AmChart`.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

`amOptions`, `amBarplot`, `amBoxplot`, `amHist`, `amPie`, `amPlot`, `amTimeSeries`, `amStockMultiSet`, `amBullet`, `amRadar`, `amWind`, `amFunnel`, `amAngularGauge`, `amSolidGauge`, `amMekko`, `amCandlestick`, `amFloatingBar`, `amOHLC`, `amWaterfall`

**Examples**

```
amHist(x = rnorm(100))

## Not run:
# Other examples available which can be time consuming depending on your configuration.

x <- replicate(1000, {
  if (round(runif(1))) {
    rnorm(1)
  } else {
    rnorm(1, mean = 5)
  }
})

# Without plot
amHist(x = x, plot = FALSE)

# With options
amHist(x = x, border = "blue")
amHist(x = x, col = "lightblue", control_hist = list(breaks = 100))
amHist(x = x, col = "grey")
amHist(x = x, col = "gray")
amHist(x = x, main = "Histogram", ylab = "y-axis", xlab = "x-axis", col = "red")
amHist(x = x, main = "Histogram", ylab = "y-axis", xlab = "x-axis", ylim = c(10, 15))
amHist(x = x, main = "Histogram", ylab = "y-axis", xlab = "x-axis")

# Options for computing the histogram
amHist(x = x, control_hist = list(breaks = "Scott"))

## End(Not run)
```

---

AmLegend-class	<i>AmLegend class</i>
----------------	-----------------------

---

**Description**

Creates the legend for the chart, automatically adapts the color settings of the graphs.

**Details**

Run `api("AmLegend")` for more information and all available properties.

**Slots**

`useGraphSettings` logical. If TRUE, border color will be changed when user rolls-over the slice, graph, etc, instead of background color.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

amLines	<i>amLines adds a serie to a graph.</i>
---------	---

---

**Description**

amLines adds a new serie to an existing serial chart.

**Usage**

```
amLines(
  chart,
  x = NULL,
  y = NULL,
  type = c("points", "line", "smoothedLine", "both", "step"),
  col = "#0066cc",
  title,
  fill_alphas = 0,
  balloon = T
)
```



**Arguments**

chart	<a href="#">AmChart</a> . Chart you wish to add the new serie.
x	numeric, equivalent to y, deprecated.
y	numeric.
type	(optionnal) character. Possible values are : "l" for line, "p" for points, "b" for both, "sl" for smoothed line, "st" for step
col	character, color of the new serie.
title	character, name of the new serie, used when legend is enabled.
fill_alphas	a numeric between 0 and 1 for printed area.
balloon	logical, add balloon with value or not

**Note**

It is supposed here that x or y corresponds to the y-axis, and the x-axis is automatically linked to the x values of the chart "chart". That is why it makes sense to give the y argument.

**Examples**

```

if (requireNamespace("pipeR", quietly = TRUE)) {
  require(pipeR)
  amPlot(x = rnorm(100), type = 'sl') %>>%
    amLines(x = rnorm(100), type = "p")
}

## Not run:
amPlot(x = rnorm(100), type = 'sl') %>>%
  amLines(x = rnorm(100), col = "blue") %>>%
  amLines(x = rnorm(100), type = "sl") %>>%
  amLines(x = rnorm(100), type = "p")

# For an XY chart
x <- sort(rnorm(100))
y1 <- rnorm(100, sd = 10)
y2 <- rnorm(100, sd = 10)
y3 <- rnorm(100, sd = 10)
amPlot(x = x, y = y1) %>>%
  amLines(x = y2, col = "blue") %>>%
  amLines(x = y3, type = "p")

## End(Not run)

```

amMekko

*Plotting mekko chart (quali vs quali)***Description**

amMekko computes a mekko chart of the given values.

**Usage**

```
amMekko(
  x,
  y,
  data,
  xlab = "",
  ylab = "",
  groups_color = NULL,
  horiz = FALSE,
  show_values = FALSE,
  ...
)
```

**Arguments**

x	character, column name for x-axis.
y	character, column name for y-axis.
data	data.frame, dataframe with values to display. See <a href="#">data_mekko</a>
xlab	character, label for x-axis.
ylab	character, label for y-axis.
groups_color	character vector of colors in hexadecimal, same length as the number of y modalities.
horiz	logical, TRUE for an horizontal chart, FALSE for a vertical one.
show_values	logical, TRUE to display values.
...	see <a href="#">amOptions</a> for more options.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```

data(data_mekko)
amMekko(x = "var1", y = "var2", data = data_mekko)

## Not run:
# Other examples available which can be time consuming depending on your configuration.
library(pipeR)

# Horizontal
amMekko(x = "var1", y = "var2", data = data_mekko, horiz = TRUE)

# Display values
amMekko(x = "var1", y = "var2", data = data_mekko, show_values = TRUE)

## End(Not run)

```

---

AmObject-class

*AmObject class*

---

**Description**

This is a virtual class for representing any Am\*\* class

**Slots**

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

amOHLC

*Plotting OHLC chart*

---

**Description**

amOHLC computes an OHLC chart of the given value.

**Usage**

```

amOHLC(
  data,
  xlab = "",
  ylab = "",
  horiz = FALSE,
  zoom = TRUE,
  positiveColor = "#7f8da9",
  negativeColor = "#db4c3c",
  names = c("low", "open", "close", "high"),
  dataDateFormat = NULL,
  minPeriod = ifelse(!is.null(dataDateFormat), "DD", ""),
  ...
)

```

**Arguments**

data	data.frame, dataframe with at least 5 columns : category, open (numeric), close (numeric), low (numeric), high (numeric).
xlab	character, label for x-axis.
ylab	character, label for y-axis.
horiz	logical, TRUE for an horizontal chart, FALSE for a vertical one
zoom	logical, default set to TRUE : a cursor is added to the chart.
positiveColor	character, color for positive values (in hexadecimal).
negativeColor	character, color for negative values (in hexadecimal).
names	character, names for the tooltip. Default to c("low", "open", "close", "high").
dataDateFormat	character, default set to NULL. Even if your chart parses dates, you can pass them as strings in your dataframe - all you need to do is to set data date format and the chart will parse dates to date objects. Check this page for available formats. Please note that two-digit years (YY) as well as literal month names (MMM) are NOT supported in this setting.
minPeriod	character, minPeriod Specifies the shortest period of your data. This should be set only if dataDateFormat is not 'NULL'. Possible period values: fff - milliseconds, ss - seconds, mm - minutes, hh - hours, DD - days, MM - months, YYYY - years. It's also possible to supply a number for increments, i.e. '15mm' which will instruct the chart that your data is supplied in 15 minute increments.
...	see <a href="#">amOptions</a> for more options.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
data("data_candleStick2")
amOHLC(data = data_candleStick2)

## Not run:
# Other examples available which can be time consuming depending on your configuration.
if (requireNamespace("pipeR", quietly = TRUE)) {
  require(pipeR)

# Change colors
amOHLC(data = data_candleStick2, positiveColor = "green", negativeColor = "red")

# Naming the axes
amOHLC(data = data_candleStick2, xlab = "categories", ylab = "values") %>% setChartCursor()

# Rotate the labels for x axis
amOHLC(data = data_candleStick2, labelRotation = 90)

# Change names
amOHLC(data = data_candleStick2, names = c("min", "begin", "end", "max")) %>% setChartCursor()

# Use amOptions
amOHLC(data = data_candleStick2, zoom = FALSE)
}

## End(Not run)
```

---

amOptions

*amOptions*


---

**Description**

amOptions sets the most common options for chart customization. You can set other properties with the method [setProperties](#). See details for exception.

**Usage**

```
amOptions(
  chart,
  theme = c("none", "light", "dark", "patterns", "chalk"),
  legend = FALSE,
  legendPosition = "right",
```

```

    legendAlign = "left",
    export = FALSE,
    exportFormat = character(),
    creditsPosition = "top-left",
    main = character(),
    mainColor = "#000000",
    mainSize = 15,
    zoom = FALSE,
    scrollbar = FALSE,
    scrollbarHeight = 20,
    valuescrollbar = FALSE,
    valuescrollbarHeight = 20,
    labelRotation = 0,
    ...
)

```

### Arguments

chart	<a href="#">AmChart</a> .
theme	character, possible values are : "none", "light", "dark", "patterns", "chalk", default set to "none".
legend	logical, default FALSE. TRUE to add a legend to the chart.
legendPosition	character, possible values are : "left", "right", "top" or "bottom", default set to "right".
legendAlign	character, controls the legend alignment. Possible values are : "left","right" or "center", default set to "left". Only used if legend = TRUE.
export	logical, default set to FALSE. TRUE to display export feature.
exportFormat	character, desired export format. Possible values are : "JPG", "PNG" , "SVG", "CSV", "JSON", "PDF", "XLSX", "PRINT".
creditsPosition	character, controls credits position. Possible values are : "top-left", "top-right", "bottom-left" or "bottom-right", default set to "top-left".
main	character, chart's title.
mainColor	character, main color (in hexadecimal), default set to "#000000".
mainSize	numeric, main size, default set to 15.
zoom	logical, TRUE to add a chart cursor, default set to FALSE.
scrollbar	logical, default FALSE, TRUE to display scrollbar.
scrollbarHeight	numeric, height in pixels, must be > 0.
valuescrollbar	logical, default FALSE, TRUE to display valuescrollbar.
valuescrollbarHeight	numeric, height in pixels, must be > 0.
labelRotation	numeric, rotation angle of a label. Only horizontal axis' values can be rotated. Value must be between -90 and 90.
...	Other properties added to the chart using setProperties.

## Details

### Exception:

- It's not possible to export a gauge chart data as CSV.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
## Not run:
library(pipeR)
data(data_pie)

# Export
amPie(data = data_pie) %>>%
  amOptions(export = TRUE)

# Legend
amPie(data = data_pie) %>>%
  amOptions(legend = TRUE)

# Legend position
amPie(data = data_pie) %>>%
  amOptions(legend = TRUE, legendPosition = "bottom")

# Credits position
amPie(data = data_pie) %>>%
  amOptions(creditsPosition = "bottom-right")

# Theme
amPie(data = data_pie) %>>%
  amOptions(theme = "chalk")

# Title
amPie(data = data_pie) %>>%
  amOptions(main = "Social network", mainColor = "#FFFFFF", mainSize = 40, theme = "chalk")

# Custom exemple
amPie(data = data_pie) %>>%
  amOptions(main = "Social network", mainColor = "#FFFFFF", mainSize = 40,
            theme = "dark", legend = TRUE, legendPosition = "bottom",
            creditsPosition = "bottom-right" )
```

```
## End(Not run)
```

---

amPie	<i>Plotting pie chart</i>
-------	---------------------------

---

## Description

amPie computes a pie chart of the given value.

## Usage

```
amPie(data, show_values = TRUE, depth = 0, inner_radius = 0, ...)
```

## Arguments

data	data.frame, dataframe with at least 2 columns : label (character), value (numeric). See <a href="#">data_pie</a> You can add a third column "color" (character, colors in hexadecimal).
show_values	logical, TRUE to display values.
depth	numeric, if > 0, chart is displayed in 3D, value between 0 and 100
inner_radius	numeric, value between 0 and 100
...	see <a href="#">amOptions</a> for more options.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
data("data_pie")
amPie(data = data_pie)

## Not run:
# Other examples available which can be time consuming depending on your configuration.

# Don't display values
amPie(data = data_pie, show_values = FALSE)

# 3D pie
```



```
amPie(data = data_pie, depth = 10)

# Donut chart
amPie(data = data_pie, inner_radius = 50)

# All parameters
amPie(data = data_pie, inner_radius = 50, depth = 10, show_values = FALSE)

## End(Not run)
```

---

amPlot

*Plot serial data*

---

### Description

amPlot computes a plot of the given data values (can be a vector, dataframe or formula).

### Usage

```
amPlot(x, ...)
```

## Default S3 method:

```
amPlot(x, ...)
```

## S3 method for class 'numeric'

```
amPlot(
  x,
  y,
  bullet = c("round", "diamond", "square", "bubble", "yError", "xError", "triangleLeft",
             "triangleRight", "triangleUp", "triangleDown"),
  type = c("points", "line", "smoothedLine", "step", "both"),
  col = "#0066cc",
  fill_alphas = 0,
  weights = NULL,
  precision = 2,
  title = NULL,
  id,
  error,
  xlab,
  ylab,
  lty,
  cex,
  lwd,
  xlim,
  ylim,
  ...
)
```

```
## S3 method for class 'character'
amPlot(
  x,
  y,
  bullet = c("round", "diamond", "square", "bubble", "yError", "xError", "triangleLeft",
    "triangleRight", "triangleUp", "triangleDown"),
  type = c("points", "line", "smoothedLine", "step", "both"),
  col = "#0066cc",
  fill_alphas = 0,
  weights = NULL,
  precision = 2,
  parseDates = FALSE,
  title = NULL,
  dataDateFormat,
  id,
  error,
  xlab,
  ylab,
  lty,
  cex,
  lwd,
  xlim,
  ylim,
  ...
)

## S3 method for class 'factor'
amPlot(
  x,
  y,
  bullet = "round",
  type = "p",
  col = "gray",
  weights = NULL,
  precision = 2,
  parseDates = FALSE,
  dataDateFormat = NULL,
  id,
  error,
  xlab,
  ylab,
  lty,
  cex,
  lwd,
  xlim,
  ylim,
  ...
)
```

```
## S3 method for class 'data.frame'
amPlot(x, columns, type = "l", precision = 2, xlab, ylab, fill_alphas = 0, ...)

## S3 method for class 'formula'
amPlot(x, data, type = "p", fill_alphas = 0, xlab, ylab, main = "", ...)
```

## Arguments

x	the coordinates of points in the plot : numeric, data.frame, or formula.
...	see <a href="#">amOptions</a> for more options.
y	numeric, the y coordinates of points in the plot, optional if x is an appropriate structure.
bullet	character, point shape. Possible values are : "diamond", "square", "bubble", "yError", "xError", "round", "triangleLeft", "triangleRight", "triangleUp", "triangleDown". Default set to "round".
type	character, type of plot. Possible values are : "l" for a line, "sl" for a smoothed line, "st" for step, "p" for points, and "b" for line and points. Default set to "p".
col	either a factor or a character, default set to "gray".
fill_alphas	a numeric between 0 and 1 for printed area.
weights	numeric, weights for x/y charts only. Small values are preferred for lisibility.
precision	numeric, precision you wish to display. Default set to 2.
title	character, name of the new serie, used when legend is enabled.
id	numeric, point id, for x/y charts only. Default 1:length(x).
error	numeric, only when type is "xError" "yError" default NULL,
xlab	character, label for x-axis.
ylab	character, label for y-axis.
lty	numeric, line type (dashes).
cex	numeric, bullet size.
lwd	numeric, line width
xlim	numeric, x range.
ylim	numeric, y range.
parseDates	logical, default set to FALSE, if TRUE argument dataDateFormat has to be provided.
dataDateFormat	character, default set to NULL. Even if your chart parses dates, you can pass them as strings in your dataframe - all you need to do is to set data date format and the chart will parse dates to date objects. Check this page for available formats. Please note that two-digit years (YY) as well as literal month names (MMM) are NOT supported in this setting.
columns	(optional) either a vector of character containing the names of the series to draw, or a numeric vector of indices. By default all numeric columns will be drawn.
data	dataset
main	title

**Value**

Return an Amchart.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
## Not run:
# 'numeric':
amPlot(x = rnorm(100))

# add line
chart <- amPlot(x = rnorm(100), type = 'sl', legend = T)
amLines(chart, x = rnorm(100), type = "p")

# 'character':
start <- as.POSIXct('2015-01-01')
end <- as.POSIXct('2015-12-31')
date <- seq.POSIXt(from = start, to = end, by = 'day')
date <- format(date)

y <- rnorm(length(date))
amPlot(x = date, y = y, type = 'l', parseDates = TRUE, dataDateFormat = "YYYY-DD-MM")
# notice that by default 'parseDates = FALSE'

# 'data.frame'
amPlot(iris, col = colnames(iris)[1:2], type = c("l", "st"), zoom = TRUE, legend = TRUE)

# 'formula':
amPlot(Petal.Length + Sepal.Length ~ Sepal.Width, data = iris, legend = TRUE, zoom = TRUE)

## End(Not run)

## Not run:
# Other examples available which can be time consuming depending on your configuration.
library(data.table)

iris <- as.data.table(get("iris", "package:datasets"))
x <- rnorm(100)

# Simple scatter plot with title and color
# Also change type (set to "p" by default), available "l", "sl", "st", "p", "b"
amPlot(x = x, main = "Title", col = "lightblue", type = "b")
```

```
x <- sort(rnorm(100))
y <- runif(100)
weights <- runif(100, 0, 15)
amPlot(x = x, y = y, weights = weights)

## End(Not run)
```

---

amRadar

*Plotting radar*

---

## Description

radar computes a radarplot of the given data values.

## Usage

```
amRadar(
  data,
  col = NULL,
  backTransparency = 0.5,
  type = "polygons",
  pch = "round",
  xlim = NULL,
  ...
)
```

## Arguments

data	data.frame first column is named "label" (character), other columns are series of values, see <a href="#">data_radar</a> .
col	character, color(s) of serie(s) hexadecimal like "#00FF00".
backTransparency	numeric, background transparency, between 0 and 1.
type	character, type of radar. Possible values are : "polygons" or "circle".
pch	character, points symbols. Possible values are : "round", "square", "triangleUp", "triangleDown", "triangleLeft", "triangleRight", "bubble", "diamond", "xError", "yError".
xlim	numeric, x range.
...	see <a href="#">amOptions</a> for more options.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
data("data_radar")
amRadar(data_radar)

## Not run:

# Other examples available which can be time consuming depending on your configuration.

if (requireNamespace("pipeR", quietly = TRUE)) {
  require(pipeR)

  # Change color
  amRadar(data_radar, col = "#FF0000")
  amRadar(data_radar, col = c("#0000FF", "#00FF00", "#FF0000"))

  # Change backTransparency
  amRadar(data_radar, backTransparency = 0.6)
  amRadar(data_radar, backTransparency = c(0, 0.4, 0.6))

  # Change type
  amRadar(data_radar, type = "circles")

  # Change pch
  amRadar(data_radar, pch = "triangleRight")
  amRadar(data_radar, pch = "triangleLeft")

  # Min-Max
  amRadar(data_radar, xlim = c(0, 8))
}

## End(Not run)
```

---

amSolidGauge

*Plotting solid gauge using rAmCharts*

---

**Description**

amSolidGauge computes a gauge of the given value.

## Usage

```
amSolidGauge(  
  x,  
  min = 0,  
  max = 100,  
  type = "full",  
  width = 20,  
  color = "",  
  text = "",  
  textSize = 20,  
  ...  
)
```

## Arguments

x	numeric, value for which the angular gauge is desired.
min	numeric, minimal possible value.
max	numeric, maximal possible value.
type	character, type of gauge : "full" or "semi".
width	numeric, width of the gauge.
color	character, hexadecimal color value or a vector of colors.
text	character, text.
textSize	numeric, text size.
...	see <a href="#">amOptions</a> for more options.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
amSolidGauge(x = 65)  
  
## Not run:  
# Other examples available which can be time consuming depending on your configuration.  
  
if (requireNamespace("pipeR", quietly = TRUE)) {  
  require(pipeR)  
  
  # Change min and max values
```

```

amSolidGauge(x = 65, min = 0, max = 200)

# Semi solid gauge
amSolidGauge(x = 65, type = "semi")

# Change width
amSolidGauge(x = 65, width = 50)

# Change color
amSolidGauge(x = 65, color = "#2F4F4F")

# Put a color scale
amSolidGauge(x = 10, color = c("#00ff00", "#ffd700", "#ff0000"))
amSolidGauge(x = 35, color = c("#00ff00", "#ffd700", "#ff0000"))
amSolidGauge(x = 70, color = c("#00ff00", "#ffd700", "#ff0000"))
amSolidGauge(x = 90, color = c("#00ff00", "#ffd700", "#ff0000"))

# Add some text to the printed value
amSolidGauge(x = 65, text = "Pct")

# Modify textSize value
amSolidGauge(x = 65, text = "Pct", textSize = 50)
}

## End(Not run)

```

---

AmStockChart-class      *AmStockChart*

---

## Description

Class to draw stock charts

## Slots

balloon [AmBalloon](#).

comparedDataSets list of [DataSet](#). Properties of data sets selected for comparison.

dataSets list of [DataSet](#). Each element must be a list of [DataSet](#) properties.

dataSetSelector list of [DataSetSelector](#). You can add it if you have more than one data set and want users to be able to select/compare them.

mainDataSet [DataSet](#). Data set selected as main.

panels list of [StockPanel](#).

periodSelector [PeriodSelector](#). You can add it if you want users to be able to enter date ranges or zoom chart with predefined period buttons.

theme character

type equals "stock"



group character for synchronization

is\_ts\_module logicalOrMissing. Don't use. For [rAmChartsTimeSeriesUI](#)

listeners list containing the listeners to add to the chart. The list must be named as in the official API. Each element must be a character string.

otherProperties list containing other available properties not yet implemented in the package.

value numeric.

### Author(s)

datastorm-open

### See Also

<https://docs.amcharts.com/3/javascriptstockchart/AmStockChart>

---

amStockMultiSet      *Plotting multi data-sets*

---

### Description

amStockMultiSet compute a stock of multi data-sets

### Usage

```
amStockMultiSet(
  data,
  panelColumn = NULL,
  ZoomButtonPosition = "bottom",
  ZoomButton = data.frame(Unit = "MAX", multiple = 1, label = "All"),
  color = c("#2E2EFE", "#31B404", "#FF4000"),
  precision = 1,
  export = FALSE,
  percentHeightPanel = NULL,
  creditsPosition = "top-right",
  ...
)
```

### Arguments

data	list, list of data.frame (same structure) first column is date, others are values
panelColumn	numeric, numeric vector, controle panel adding for selected series
ZoomButtonPosition	character, zoom button position. Possible values are : "left", "right", "bottom", "top"
ZoomButton	data.frame, 3 columns : Unit, times unit multiple : multiple*unit label : button's label

color	character, color of data-sets (in hexadecimal).
precision	numeric, digits precision
export	logical, default set to FALSE. TRUE to display export feature.
percentHeightPanel	numeric, vector of size panel, same length than data
creditsPosition	character, credits position. Possible values are : "top-right", "top-left", "bottom-right", "bottom-left"
...	other first level attributes

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
## Not run:
data(data_stock_3)

amStockMultiSet(data = data_stock_3)
amStockMultiSet(data = data_stock_3, panelColumn = c(1,2,1,1))

amStockMultiSet(data = data_stock_3, panelColumn = c(1,2,3,4))

ZoomButton <- data.frame(Unit = c("DD", "DD", "MAX"), multiple = c(1, 10 ,1),
  label = c("Day", "10 days", "MAX"))
ZoomButtonPosition <- "bottom"
amStockMultiSet(data = data_stock_3, panelColumn = c(1,2,1,1), ZoomButton = ZoomButton,
ZoomButtonPosition = "top")

amStockMultiSet(data = data_stock_3, precision = 2)

amStockMultiSet(data = data_stock_3, panelColumn = c(1,2,1,1), percentHeightPanel = c(3,1))

## End(Not run)
```

---

amTimeSeries	<i>Plotting times series (with or without aggregation)</i>
--------------	--

---

## Description

amTimeSeries computes a stock chart.

## Usage

```
amTimeSeries(
  data,
  col_date,
  col_series,
  main = "",
  ylab = "",
  color = c("#2E2EFE", "#31B404", "#FF4000", "#AEB404"),
  type = c("line"),
  bullet = NULL,
  bulletSize = 2,
  linetype = c(0, 5, 10, 15, 20),
  linewidth = c(1, 1, 1, 1, 1, 1),
  fillAlphas = 0,
  precision = 1,
  connect = FALSE,
  export = FALSE,
  legend = TRUE,
  legendPosition = "bottom",
  legendHidden = FALSE,
  aggregation = c("Average", "Low", "High", "Sum"),
  maxSeries = 300,
  groupToPeriods = c("ss", "mm", "hh", "DD", "MM", "YYYY"),
  checkGroupToPeriods = TRUE,
  ZoomButton = data.frame(Unit = "MAX", multiple = 1, label = "All"),
  ZoomButtonPosition = "bottom",
  periodFieldsSelection = FALSE,
  scrollbar = TRUE,
  scrollbarPosition = "bottom",
  scrollbarHeight = 40,
  scrollbarGraph = NULL,
  cursor = TRUE,
  cursorValueBalloonsEnabled = TRUE,
  creditsPosition = "top-right",
  group = NULL,
  is_ts_module = FALSE,
  dataDateFormat = "YYYY-MM-DD JJ:NN:SS",
  categoryBalloonDateFormats = list(list(period = "YYYY", format = "YYYY"), list(period =
    "MM", format = "YYYY-MM"), list(period = "WW", format = "YYYY-MM-DD"), list(period =
```

```

"DD", format = "YYYY-MM-DD"), list(period = "hh", format = "YYYY-MM-DD JJ:NN"),
list(period = "mm", format = "YYYY-MM-DD JJ:NN"), list(period = "ss", format =
"YYYY-MM-DD JJ:NN:SS"), list(period = "fff", format = "YYYY-MM-DD JJ:NN:SS")),
dateFormats = list(list(period = "YYYY", format = "YYYY"), list(period = "MM", format =
"MMM"), list(period = "WW", format = "MMM DD"), list(period = "DD", format =
"MMM DD"), list(period = "hh", format = "JJ:NN"), list(period = "mm", format =
"JJ:NN"), list(period = "ss", format = "JJ:NN:SS"), list(period = "fff", format =
"JJ:NN:SS")),
thousandsSeparator = " ",
decimalSeparator = ".",
balloonFontSize = 10,
balloonMaxWidth = 400,
...
)

```

### Arguments

data	data.frame, data of graph.
col_date	character name of date column
col_series	character names of series columns
main	character, title.
ylab	character, value axis label.
color	character, color of series (in hexadecimal).
type	character, Type of graph. Possible values are : "line" (default), "column", "step", "smoothedLine"
bullet	character, point shape. Possible values are : "diamond", "square", "bubble", "yError", "xError", "round", "triangleLeft", "triangleRight", "triangleUp", "triangleDown"
bulletSize	: numeric, size of bullet.
linetype	: numeric, line type, 0 : solid, number : dashed length
linewidth	: numeric, line width.
fillAlphas	: numeric, fill. Between 0 (no fill) to 1.
precision	numeric, default set to 1.
connect	logical, default set to FALSE. Specifies whether to connect data points if data is missing.
export	logical, default set to FALSE. TRUE to display export feature.
legend	logical, enabled or not legend ? Defaut to TRUE.
legendPosition	character, legend position. Possible values are : "left", "right", "bottom", "top"
legendHidden	logical hide some series on rendering ? Defaut to FALSE
aggregation	character, aggregation type. Possible values are : "Low", "High", "Average" and "Sum"

maxSeries	numeric Maximum series shown at a time. In case there are more data points in the selection than maxSeries, the chart will group data to longer periods, for example - you have 250 days in the selection, and maxSeries is 150 - the chart will group data to weeks.
groupToPeriods	character, Periods to which data will be grouped in case there are more data items in the selected period than specified in maxSeries property. Possible value are : 'ss', 'mm', 'hh', 'DD', 'MM', 'YYYY'. It's also possible to add multiple like "30mm". Or NULL to disable.
checkGroupToPeriods	logical. Check groupToPeriods ? Default to TRUE then check validity between data and groupToPeriods, else use directly groupToPeriods
ZoomButton	data.frame, 3 or 4 columns : <ul style="list-style-type: none"> <li>• "Unit" : Character. Times unit. 'ss', 'mm', 'hh', 'DD', 'MM', 'YYYY'</li> <li>• "multiple" : Numeric. multiple*unit</li> <li>• "label" : Character. button's label</li> <li>• "selected" : Boolean. Optional. To set initial selection. (One TRUE, others FALSE)</li> </ul>
ZoomButtonPosition	character, zoom button position. Possible values are : "left", "right", "bottom", "top"
periodFieldsSelection	boolean, using zoom button, add also two fields to select period ?
scrollbar	boolean, enabled or not scrollbar ? Default to TRUE.
scrollbarPosition	character, scrollbar position. Possible values are : "left", "right", "bottom", "top"
scrollbarHeight	numeric, height of scroll bar. Default : 40.
scrollbarGraph	character, name of serie (column) to print in scrollbar. Default to NULL.
cursor	boolean, enabled or not cursor ? Default to TRUE.
cursorValueBalloonsEnabled	boolean, if cursor, enabled or not balloons on cursor ? Default to TRUE.
creditsPosition	character, credits position. Possible values are : "top-right", "top-left", "bottom-right", "bottom-left"
group	character, like in dygraphs, for synchronization in shiny or rmarkdown.
is_ts_module	boolean. Don't use. For <a href="#">rAmChartsTimeSeriesUI</a>
dataDateFormat	character Data date format. Default to 'YYYY-MM-DD JJ:NN:ss'. See details.
categoryBalloonDateFormats	list Date format objects for chart cursor. See details.
dateFormats	list Date format objects for x-axis. See details.
thousandsSeparator	character, default set to " "

decimalSeparator  
                                   character, default set to ".",  
 balloonFontSize  
                                   numeric, text font size on balloon. Default : 10.  
 balloonMaxWidth  
                                   numeric. Default : 400.  
 ...                               other first level attributes

## Details

For dateFormats & categoryBalloonDateFormats

Available periods are: fff - millisecond, ss - second, mm - minute, hh - hour, DD - date, WW - week, MM - month, YYYY - year

Available formats :

Year. The number of Y letters represents digits in the resulting number. I.e.: YY = 05 (always two digits), YYYY = 2005

Month of the year. The output depends on the number of M's: M = 8 (one or two digits), MM = 08 (always two digits), MMM = Aug (3-letter month abbreviation), MMMM = August (full month name)

Week of the year W

Day of the month: D = 7 (one or two digits), DD = 07 (always two digits)

Day of week: E = 2 (single digit day of week), EE = 02 (two-digit day of week), EEE = Tue (3-letter abbreviation of the literal representation of the day of week), EEEE = Tuesday (full day of week name)

Hour: 0-23: J = 3 (one or two digits), JJ = 03 (always two digits)

Hour: 1-24: H = 3 (one or two digits), HH = 03 (always two digits) K Hour in am/pm: 0-11 L Hour in am/pm: 1-12 \*

Minute in hour: N = 8 (one or two digits), NN = 08 (always two digits)

Second in minute: S = 5 (one or two digits), SS = 05 (always two digits)

Milliseconds: QQ = 51, QQQ = 051

Other characters Other characters will be displayed as they are without changing them. I.e.:YYYY-MM-DD = 2013-03-01

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[rAmChartsTimeSeriesUI](#) for shiny module, [amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```

## Not run:

data("data_stock_2")
amTimeSeries(data_stock_2, "date", c("ts1", "ts2"))

# upper /lower
data <- data_stock_2[1:50, ]
data$ts1low <- data$ts1-100
data$ts1up <- data$ts1+100

amTimeSeries(data, "date", list(c("ts1low", "ts1", "ts1up"), "ts2"))
amTimeSeries(data, "date", list(c("ts1low", "ts1", "ts1up"), "ts2"),
              color = c("red", "blue"), bullet = c("round", "square"))

# column / step
amTimeSeries(data_stock_2, "date", c("ts1", "ts2"),
              type = c("column", "step"), fillAlphas = c(1, 0),
              linewidth = c(0, 1))

# some parameters
amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round")
amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",
              groupToPeriods = c('hh', 'DD', '10DD'))

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",
              groupToPeriods = c('12hh', 'DD', '10DD'),
              maxSeries = 50)

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",
              groupToPeriods = c('hh', 'DD', '10DD'),
              linewidth = c(3, 1))

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), aggregation = "Sum")

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",
              groupToPeriods = c('12hh', 'DD', '10DD'),
              maxSeries = 50, precision = 5)

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = c("diamond", "square"),
              linetype = 0, bulletSize = c(5, 10),
              groupToPeriods = c('12hh', 'DD', '10DD'),
              maxSeries = 50, aggregation = "Sum")

ZoomButton <- data.frame(Unit = c("DD", "DD", "MAX"), multiple = c(1, 2 ,1),
                          label = c("Day", "2 days", "MAX"))
amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",
              ZoomButton = ZoomButton, main = "My title", ylab = "Interest")

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",

```

```

ZoomButton = ZoomButton, main = "My title", ylab = "Interest",
export = TRUE, ZoomButtonPosition = "right",
legendPosition = "bottom", scrollbarPosition = "top")

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"), bullet = "round",
ZoomButton = ZoomButton, main = "My title",
ylab = "Interest", export = TRUE,
creditsPosition = "bottom-left")

# date formats

amTimeSeries(data_stock_2, "date", c("ts1", "ts2"),
type = "column", fillAlphas = 1,
linewidth = 0, legendHidden = T,
categoryBalloonDateFormats = list(list(period = 'YYYY', format = 'YYYY'),
list(period='MM', format = 'MM'),
list(period = 'WW', format = 'MM-DD'),
list(period='DD', format = 'MM-DD'),
list(period = 'hh', format = 'MM-DD JJ:NN'),
list(period='mm', format = 'MM-DD JJ:NN'),
list(period = 'ss', format = 'MM-DD JJ:NN:SS'),
list(period='fff', format = 'MM-DD JJ:NN:SS')),
dateFormats = list(list(period = 'YYYY', format = 'YYYY'),
list(period='MM', format = 'MMMM'),
list(period = 'WW', format = 'MMMM-DD'),
list(period='DD', format = 'MMMM-DD'),
list(period = 'hh', format = 'MM-DD JJ:NN'),
list(period='mm', format = 'MM-DD JJ:NN'),
list(period = 'ss', format = 'MM-DD JJ:NN:SS'),
list(period='fff', format = 'MM-DD JJ:NN:SS'))))

## End(Not run)

```

---

amWaterfall

*Plotting waterfall*


---

## Description

amWaterfall computes a waterfall chart of the given value.

## Usage

```
amWaterfall(data, start = 0, horiz = FALSE, show_values = FALSE, ...)
```

## Arguments

**data** data.frame, dataframe with at least 3 columns : label (character), value (numeric), operation (character : "plus", "minus", "total"). You can add a third



column "color" (character, colors in hexadecimal). You can also add a column "description" (character) containing the text you want to display when mouse is on the graphic ('<br>' for a new line). See [data\\_waterfall](#).

start            numeric, value from which to start.

horiz            logical, TRUE for an horizontal chart, FALSE for a vertical one.

show\_values     logical, TRUE to display values on the chart.

...              see [amOptions](#) for more options.

## References

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

## See Also

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

## Examples

```
data("data_waterfall")
amWaterfall(data = data_waterfall, show_values = TRUE)

## Not run:
# Other examples available which can be time consuming depending on your configuration.

# Change the orientation :
amWaterfall(data = data_waterfall, horiz = TRUE)

## End(Not run)
```

---

amWind

*Plotting wind*

---

## Description

amWind computes a windplot of the given data values.

## Usage

```
amWind(data, col = NULL, backTransparency = 0.5, ...)
```

**Arguments**

data	data.frame, a dataframe which columns are series of values, from weakest wind (first column) to stronger wind (last column). See <a href="#">data_wind</a> .
col	character, color(s) of serie(s) hexadecimal like "#00FF00".
backTransparency	numeric, background transparency, between 0 and 1.
...	see <a href="#">amOptions</a> for more options.

**References**

See online documentation [https://datastorm-open.github.io/introduction\\_ramcharts/](https://datastorm-open.github.io/introduction_ramcharts/) and [amChartsAPI](#)

**See Also**

[amOptions](#), [amBarplot](#), [amBoxplot](#), [amHist](#), [amPie](#), [amPlot](#), [amTimeSeries](#), [amStockMultiSet](#), [amBullet](#), [amRadar](#), [amWind](#), [amFunnel](#), [amAngularGauge](#), [amSolidGauge](#), [amMekko](#), [amCandlestick](#), [amFloatingBar](#), [amOHLC](#), [amWaterfall](#)

**Examples**

```
## Not run:
data("data_wind")
amWind(data_wind)

# Other examples available which can be time consuming depending on your configuration.

# Change color
amWind(data = data_wind, col = "#0404B4")
amWind(data = data_wind, col = c("#0404B4", "#01DF01", "#FFBF00"))

# Change backTransparency
amWind(data = data_wind, col = c("#0404B4", "#01DF01", "#FFBF00"), backTransparency = 0.1)
amWind(data = data_wind, col = c("#0404B4", "#01DF01", "#FFBF00"), backTransparency = 1)
amWind(data = data_wind, col = c("#0404B4", "#01DF01", "#FFBF00"), backTransparency = c(0.1, 0.1, 1))

## End(Not run)
```

---

 api

 See *AmCharts API*


---

**Description**

Open a window in your browser at the referenced documentation under <https://docs.amcharts.com/3/javascriptstockchart/>.

**Usage**

```
api(class = NULL)
```

**Arguments**

`class` Object of class character. Name of the class to see documentation. Please respect lower and upper case.

**Examples**

```
api()
api("AmChart")
```

---

AxisBase-class      *AxisBase class*

---

**Description**

Base class for ValueAxis and CategoryAxis. It can not be explicitly instantiated.

**Slots**

`guides` list.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.

`value` numeric. Guides of this axis. Use addGuide method.

**Author(s)**

datastorm-open

---

CategoryAxis-class      *CategoryAxis class*

---

**Description**

Children class of AxisBase. Automatically set.

**Details**

Run `api("CategoryAxis")` for more information and all available properties.

**Slots**

`gridPosition` character. Specifies if a grid line is placed on the center of a cell or on the beginning of a cell. Possible values are: "start" and "middle" This setting doesn't work if `parseDates` is set to TRUE.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

ChartCursor-class      *ChartCursor class*

---

**Description**

Creates a cursor for the chart which follows the mouse movements. In case of AmSerialChart charts it shows the balloons of hovered data points.

**Details**

Run `api("ChartCursor")` for more information and all available properties.

**Slots**

`oneBalloonOnly` logical. If TRUE, border color will be changed when user rolls-over the slice, graph, etc, instead of background color.

`valueLineAxis` list. Properties of Axis of value line. If you set `valueLineBalloonEnabled` to true, but you have more than one axis, you can use this property to indicate which axis should display balloon.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

ChartScrollbar-class *ChartScrollbar class*

---

### Description

Creates a scrollbar for amSerialChart and amXYChart charts.

### Details

Run `api("ChartScrollbar")` for more information and all available properties.

### Slots

`enabled` `logical`. Specifies if the chart should be updated while dragging/resizing the scrollbar or only at the moment when user releases mouse button.

`graph` `list`. Specifies which graph properties will be displayed in the scrollbar. Only Serial chart's scrollbar can display a graph.

`listeners` `list` containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` `list` containing other available properties not yet implemented in the package.  
`value` `numeric`.

### Author(s)

datastorm-open

---

`controlShinyPlot` *Tests the class of an expression.*

---

### Description

Only used in 'renderAmCharts'.

### Usage

```
controlShinyPlot(x)
```

### Arguments

`x` expression passed to 'renderAmCharts'. Either an expression that generates an HTML widget, or an expression that generates an AmChart.

### Details

This function has only an internal purpose. Never use it.

---

DataSet-class

*DataSet class*

---

### Description

DataSet is an object which holds all information about data for [AmStockChart](#)

### Details

Run `api("DataSet")` for more information.

### Slots

`dataProvider` list, the data set data. Important: the data sets need to come pre-ordered in ascending order. Data with incorrect order might result in visual and functional glitches on the chart.

`fieldMappings` list, field mappings. Field mapping is an object with `fromField` and `toField` properties. `fromField` is the name of your value field in `dataProvider`. `toField` might be chosen freely, it will be used to set value/open/close/high/low fields for the StockChart. Example: (`fromField:"val1", toField:"value"`).

`stockEvents` list of [StockEvent](#).

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

### Author(s)

datastorm-open

---

DataSetSelector-class *DataSetSelector class*

---

### Description

DataSetSelector is a tool used to select data sets as main and to compare to main data set.

### Details

Run `api("DataSetSelector")` for more information and all available properties.

**Slots**

`position` character. Possible values: "right", "left", "top", "bottom". "top" and "bottom" positions has a limitation - only one data set can be selected for comparing.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

data\_AirPassengers      *Air passengers for example*

---

**Description**

Based on the dataset 'AirPassengers' from the package 'datasets'.

**Usage**

data\_AirPassengers

**Format**

2 column, 144 rows :

**AirPassengers** numeric

**Period** character, MM/YYYY

---

data\_bar      *Random data for plotting bar chart examples*

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**

data\_bar

**Format**

Dataset of 3 columns and 12 rows

**country** character

**visits** numeric

**color** character

---

data\_candleStick1      *Random data for plotting candlestick chart examples*

---

**Description**

This dataset is used in the tutorial, moreover if you notice a bug, use this dataset to give us an example.

**Usage**

data\_candleStick1

**Format**

Dataset of 5 columns and 12 rows

**category** character, can be parsed as a date

**open** numeric

**close** numeric

**low** numeric

**high** numeric

---

data\_candleStick2      *Random data for plotting candlestick chart examples*

---

**Description**

This dataset is used in the tutorial, moreover if you notice a bug, use this dataset to give us an example.

**Usage**

data\_candleStick2

**Format**

Dataset of 5 columns and 12 rows

**category** character, can be parsed as a date

**open** numeric

**close** numeric

**low** numeric

**high** numeric



---

`data_fbar`*Random data for plotting floating bar chart examples*

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**`data_fbar`**Format**

Dataset of 2 columns and 1000 rows

**country** character

**visits\_inf** numeric

**visits\_sup** numeric

**color** character

---

`data_funnel`*Random data for plotting funnel chart examples*

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**`data_funnel`**Format**

Dataset of 2 columns and 7 rows

**description** character

**value** numeric

---

`data_gantt`*Random data for plotting gantt chart examples*

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**`data_gantt`**Format**

Dataset of 5 columns and 4 rows

**category** character

**begin** date

**end** date

**color** character

---

`data_gbar`*Random data for plotting bar chart examples*

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**`data_gbar`**Format**

Dataset of 5 columns and 12 rows

**year** character

**day** character

**month** character

**income** numeric

**expenses** numeric

---

data\_gdp

*10 Richest Countries in the World by 2015 GDP*

---

**Description**

Value in \$ trillion

**Usage**

data\_gdp

**Format**

Dataset of 2 columns and 10 rows

**country** character

**gdp** numeric

**Source**

<https://www.insidermonkey.com/blog/10-richest-countries-in-the-world-by-2015-gdp-344692/>

---

data\_mekko

*Random data for plotting mekko chart examples*

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**

data\_mekko

**Format**

Dataset of 2 columns and 1000 rows

**var1** character

**var2** numeric

---

data\_pie

*Random data for plotting pie chart examples*

---

### **Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

### **Usage**

data\_pie

### **Format**

Dataset of 2 columns and 5 rows

**label** character

**value** numeric

---

data\_radar

*Random data for plotting radar chart examples*

---

### **Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

### **Usage**

data\_radar

### **Format**

Dataset of 4 columns and 5 rows

**label** character

**Product1** numeric

**Product2** numeric

**Product3** numeric

---

data_stock1	<i>Random data for example</i>
-------------	--------------------------------

---

**Description**

A list containing 4 datasets.

**Usage**

```
data_stock1
```

**Format**

Each dataset is a data.table with 21 rows and 4 variables:

- date** vector of dates
- a** random vector of data
- b** random vector of data

---

data_stock_2	<i>Random data for example</i>
--------------	--------------------------------

---

**Description**

Times Series on 2015-2016, one data by hour

**Usage**

```
data_stock_2
```

**Format**

Each dataset is a data.table with 21 rows and 4 variables:

- date** vector of dates
- ts1** random vector of data
- ts2** random vector of data

---

data_stock_3	<i>Random data for example</i>
--------------	--------------------------------

---

**Description**

Times Series on 2017, by months

**Usage**

data\_stock\_3

**Format**

List of 4 datasets, 4 variables in each

**date** vector of datesTimes

**value** random vector of data

**volume** random vector of data

**value2** random vector of data

**value3** random vector of data

---

data_waterfall	<i>Random data for plotting candlestick chart examples</i>
----------------	--

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**

data\_waterfall

**Format**

Dataset of 3 columns and 15 rows

**label** character, can be parsed as a date

**value** numeric

**operation** numeric

---

data_wind	<i>Random data for plotting wind chart examples</i>
-----------	---

---

**Description**

This dataset is used in the examples, moreover it can be used as a reference if you notice a bug.

**Usage**

data\_wind

**Format**

Dataset of 3 columns and 8 rows

**weak** numeric

**middle** numeric

**strong** numeric

---

GaugeArrow-class	<i>GaugeArrow class</i>
------------------	-------------------------

---

**Description**

Creates an arrow for AmAngularGaugeChart, multiple can be assigned.

**Details**

Run `api("GaugeArrow")` for more information and all available properties.

**Slots**

`axis` list containing properties of [GaugeAxis](#). Axis of the arrow. You can use reference to the axis or id of the axis. If you don't set any axis, the first axis of the chart will be used.

`listeners` list containing the listeners to add to the chart. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

GaugeAxis-class	<i>GaugeAxis class</i>
-----------------	------------------------

---

**Description**

Creates an axis for AmAngularGaugeChart, multiple can be assigned.

**Details**

Run `api("GaugeAxis")` for more information and all available properties.

**Slots**

`bands` list containing properties of one or several [GaugeBand](#) objects. Bands are used to draw color fills between specified values.

`listeners` list containing the listeners to add to the chart. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

GaugeBand-class	<i>GaugeBand class</i>
-----------------	------------------------

---

**Description**

Creates a band for a specified value range on the GaugeAxis. Multiple bands can be assigned to a single GaugeAxis.

**Details**

Run `api("GaugeBand")` for more information and all available properties.

**Slots**

`id` character. Unique id of a band.

`listeners` list containing the listeners to add to the chart. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open



---

Generics functions      *General shared generic methods*

---

### Description

These methods are shared by at least two different classes

### Usage

```
setBalloon(.Object, amBalloon = NULL, ...)  
setDataProvider(.Object, dataProvider, keepNA = TRUE)  
setTitle(.Object, title)  
setType(.Object, type)  
setGraph(.Object, graph = NULL, ...)  
addGuide(.Object, guide = NULL, ...)  
setText(.Object, text)  
setValueAxis(.Object, valueAxis = NULL, ...)
```

### Arguments

.Object	<a href="#">AmObject</a> .
amBalloon	<a href="#">AmBalloon</a> .
...	Other properties.
dataProvider	data.frame.
keepNA	logical, default set to TRUE. Indicates if NULL values have to be kept or ignored.
title	character.
type	character.
graph	<a href="#">AmGraph</a> .
guide	<a href="#">Guide</a> .
text	character.
valueAxis	<a href="#">ValueAxis</a> .

**Details**

Be cautious when using one of these functions since they have several signatures (S4).

setBalloon is shared by AmChart and AmStockChart.

setDataProvider(..) is shared by AmGraph and DataSet.

setTitle(...) is Shared by AmGraph and ValueAxis.

setType(...) is shared by AmGraph and AmChart.

setGraph(...) is shared by AmChart and ChartScrollbar.

addGuide(...) is shared by AxisBase and AmChart.

setText(...) is shared by Title and Label.

setValueAxis(...) is shared by AmChart(type = "gantt"), TrendLine and Guide.

**Value**

An updated 'AmObject'.

---

getCurrentStockData    *Get data in shiny module*

---

**Description**

Get data in shiny module

**Usage**

```
getCurrentStockData(
  data,
  col_date,
  col_series,
  zoom = NULL,
  maxPoints = 1000,
  tz = "UTC",
  ts = c("5 min", "10 min", "30 min", "hour", "3 hour", "12 hour", "day", "week",
        "month", "year"),
  fun_aggr = "mean",
  treat_missing = FALSE,
  maxgap = Inf,
  type_aggr = "first",
  na.rm = TRUE
)
```

**Arguments**

<code>data</code>	: data.frame to transform
<code>col_date</code>	: Date column name, default to "date". Must be "POSIXct" or "CET24" column
<code>col_series</code>	: Column name of quantitative variable(s) to be transformed. Default to <code>setdiff(colnames(data), "date")</code>
<code>zoom</code>	: List for init subset. NULL to keep all
<code>maxPoints</code>	: Maximal number of rows in results
<code>tz</code>	: Timezone of result. Default to "UTC".
<code>ts</code>	: Increment of the sequence. Default to "10 min". Can be a number, in seconds, or a character string containing one of "min", "hour", "day". This can optionally be preceded by a positive integer and a space
<code>fun_aggr</code>	: Aggregation function to use ("min", "max", "sum", "mean", "first", "last", "minabs", "maxabs"). Default to "mean".
<code>treat_missing</code>	: Boolean. Default to FALSE Whether or not to interpolate missing values ? see <code>na.approx</code>
<code>maxgap</code>	: When interpolate missing values with <code>na.approx</code> . Maximum number of consecutive NAs to fill. Default to Inf.
<code>type_aggr</code>	: Character. Type of aggregation <ul style="list-style-type: none"> <li>• "first" : Date/Time result is equal to minimum of sequence, and this minimum is included in aggregation</li> <li>• "last" : Date/Time result is equal to maximum of sequence, and this maximum is included in aggregation</li> </ul>
<code>na.rm</code>	: aggregation only. a logical value indicating whether NA values should be stripped before the computation proceeds.

---

<code>getTransformTS</code>	<i>Transform quantitative variables.</i>
-----------------------------	--

---

**Description**

Transform quantitative variables. Aggregate or interpolate time series data.

**Usage**

```
getTransformTS(
  data,
  col_date = "date",
  col_series = setdiff(colnames(data), c(col_date, col_by)),
  col_by = NULL,
  ts = "10 min",
  tz = "UTC",
  fun_aggr = "mean",
```

```

    treat_missing = FALSE,
    control_date = TRUE,
    maxgap = Inf,
    keep_last = TRUE,
    type_aggr = "first",
    showwarn = FALSE,
    na.rm = TRUE
  )

```

### Arguments

**data** : data.frame to transform

**col\_date** : Date column name, default to "date". Must be "POSIXct"

**col\_series** : Column name of quantitative variable(s) to be transformed. Default to `setdiff(colnames(data), "date")`

**col\_by** : Column name of a optional grouping variable. Default to NULL

**ts** : Increment of the sequence. Default to "10 min". Can be a number, in seconds, or a character string containing one of "min", "hour", "day". This can optionally be preceded by a positive integer and a space

**tz** : Timezone of result. Default to "UTC".

**fun\_aggr** : Aggregation function to use ("min", "max", "sum", "mean", "first", "last", "maxabs", "minabs"). Default to "mean".

**treat\_missing** : Boolean. Default to FALSE Whether or not to interpolate missing values ? see `na.approx`

**control\_date** : Boolean. Control full data sequence ? Default to TRUE and set to TRUE if `treat_missing`

**maxgap** : When interpolate missing values with `na.approx`. Maximum number of consecutive NAs to fill. Default to Inf.

**keep\_last** : Boolean. Keep last date/time value after interpolation ?

**type\_aggr** : Character. Type of aggregation

- "first" : Date/Time result is equal to minimum of sequence, and this minimum is included in aggregation
- "last" : Date/Time result is equal to maximum of sequence, and this maximum is included in aggregation

**showwarn** : Boolean. Show warnings ?

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

### Value

a data.frame

---

 Guide-class

*Guide class*


---

**Description**

Creates a horizontal/vertical guideline-/area for `amSerialChart`, `amXYChart` and `amRadarChart` charts, automatically adapts it's settings from the axes if none has been specified.

**Details**

Run `api("Guide")` for more information and all available properties.

**Slots**

`fillAlpha` numeric. Specifies if a grid line is placed on the center of a cell or on the beginning of a cell. Possible values are: "start" and "middle" This setting doesn't work if `parseDates` is set to `TRUE`.

`valueAxis` [ValueAxis](#). As you can add guides directly to the chart, you might need to specify which value axis should be used.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

 initialize,AmBalloon-method

*Initializes an AmBalloon*


---

**Description**

Initializes or updates an object [AmBalloon](#).

**Usage**

```
## S4 method for signature 'AmBalloon'
initialize(Object, adjustBorderColor, color, cornerRadius, fillColor, ...)

amBalloon(adjustBorderColor, color, cornerRadius, fillColor, ...)

setAdjustBorderColor(Object, adjustBorderColor)
```

```

## S4 method for signature 'AmBalloon,logical'
setAdjustBorderColor(.Object, adjustBorderColor)

setColor(.Object, color)

## S4 method for signature 'AmBalloon,character'
setColor(.Object, color)

setCornerRadius(.Object, cornerRadius)

## S4 method for signature 'AmBalloon,numeric'
setCornerRadius(.Object, cornerRadius)

setFillColor(.Object, fillColor)

## S4 method for signature 'AmBalloon,character'
setFillColor(.Object, fillColor)

```

### Arguments

.Object	<a href="#">AmBalloon</a> .
adjustBorderColor	logical, if TRUE, border color will be changed when user rolls-over the slice, graph, etc, instead of background color.
color	character, balloon text color.
cornerRadius	numeric, balloon corner radius.
fillColor	character, balloon background color. It is usually defined by the chart itself. If "adjustBorderColor" is set to TRUE, the balloon background color will be equal to "fillColor".
...	other properties of AmBalloon. See <a href="https://docs.amcharts.com/3/javascriptcharts/AmBalloon">https://docs.amcharts.com/3/javascriptcharts/AmBalloon</a> .

### Value

An object, possibly updated, of class [AmBalloon](#).

### Examples

```

new("AmBalloon", cornerRadius = 10)

amBalloon(adjustBorderColor = TRUE, color = "#000000", other = TRUE)

setAdjustBorderColor(.Object = amBalloon(), adjustBorderColor = TRUE)

setColor(.Object = amBalloon(), color = "#000000")

setCornerRadius(.Object = amBalloon(), cornerRadius = 5)

```

```
setFillColor(.Object = amBalloon(), fillColor = "#FFFFFF")
```

---

initialize,AmChart-method

*Creates an AmChart*

---

## Description

Method to initialize any S4 class provided by the package.

## Usage

```
## S4 method for signature 'AmChart'  
initialize(  
  .Object,  
  allLabels,  
  arrows,  
  axes,  
  balloon,  
  categoryAxis,  
  categoryField,  
  chartCursor,  
  chartScrollbar,  
  creditsPosition,  
  dataProvider,  
  graphs,  
  graph,  
  guides,  
  legend,  
  segmentsField,  
  theme,  
  titles,  
  trendLines,  
  type,  
  valueAxes,  
  valueAxis,  
  valueScrollbar,  
  ...  
)  
  
amChart(  
  allLabels,  
  arrows,  
  axes,  
  balloon,  
  categoryAxis,
```

```
categoryField,  
chartCursor,  
chartScrollbar,  
creditsPosition,  
dataProvider,  
graph,  
graphs,  
guides,  
legend,  
segmentsField,  
theme,  
titles,  
trendLines,  
type,  
valueAxes,  
valueAxis,  
...  
)  
  
amAngularGaugeChart(arrows, titles, axes, ...)  
  
amFunnelChart(dataProvider, marginLeft = 10, marginRight = 10, ...)  
  
amRadarChart(  
  allLabels,  
  balloon,  
  categoryField,  
  creditsPosition,  
  dataProvider,  
  graphs,  
  guides,  
  legend,  
  titles,  
  valueAxes,  
  ...  
)  
  
amSerialChart(  
  allLabels,  
  balloon,  
  categoryAxis,  
  categoryField,  
  chartCursor,  
  chartScrollbar,  
  creditsPosition,  
  dataProvider,  
  graphs,  
  guides,
```



```
    legend,  
    titles,  
    trendLines,  
    valueAxes,  
    ...  
)  
  
amPieChart(  
    allLabels,  
    balloon,  
    creditsPosition,  
    dataProvider,  
    legend,  
    titles,  
    ...  
)  
  
amGanttChart(categoryField, dataProvider, graph, segmentsField, valueAxis, ...)  
  
amXYChart(creditsPosition, dataProvider, graphs, ...)  
  
setAllLabels(.Object, allLabels)  
  
## S4 method for signature 'AmChart,list'  
setAllLabels(.Object, allLabels)  
  
addLabel(.Object, label = NULL, ...)  
  
## S4 method for signature 'AmChart,LabelOrMissing'  
addLabel(.Object, label = NULL, ...)  
  
setArrows(.Object, arrows = NULL)  
  
## S4 method for signature 'AmChart'  
setArrows(.Object, arrows = NULL)  
  
addArrow(.Object, arrow = NULL, ...)  
  
## S4 method for signature 'AmChart,GaugeArrowOrMissing'  
addArrow(.Object, arrow = NULL, ...)  
  
setAxes(.Object, axes, ...)  
  
## S4 method for signature 'AmChart,list'  
setAxes(.Object, axes)  
  
addAxe(.Object, axe = NULL, ...)
```

```
## S4 method for signature 'AmChart,GaugeAxisOrMissing'
addAxe(.Object, axe = NULL, ...)

addAxis(.Object, axis = NULL, ...)

## S4 method for signature 'AmChart,GaugeAxisOrMissing'
addAxis(.Object, axis = NULL, ...)

## S4 method for signature 'AmChart,AmBalloonOrMissing'
setBalloon(.Object, amBalloon = NULL, ...)

setCategoryAxis(.Object, categoryAxis = NULL, ...)

## S4 method for signature 'AmChart'
setCategoryAxis(.Object, categoryAxis = NULL, ...)

setCategoryField(.Object, categoryField)

## S4 method for signature 'AmChart,character'
setCategoryField(.Object, categoryField)

setChartCursor(.Object, chartCursor = NULL, ...)

## S4 method for signature 'AmChart,ChartCursorOrMissing'
setChartCursor(.Object, chartCursor = NULL, ...)

setChartScrollbar(.Object, chartScrollbar = NULL, ...)

## S4 method for signature 'AmChart,ChartScrollbarOrMissing'
setChartScrollbar(.Object, chartScrollbar = NULL, ...)

setCreditsPosition(.Object, creditsPosition)

## S4 method for signature 'AmChart,character'
setCreditsPosition(.Object, creditsPosition)

setDataLoader(.Object, url, format, ...)

## S4 method for signature 'AmChart,character,character'
setDataLoader(.Object, url, format, ...)

## S4 method for signature 'AmChart,ANY,logicalOrMissing'
setDataProvider(.Object, dataProvider, keepNA = TRUE)

setGraphs(.Object, graphs)

## S4 method for signature 'AmChart,list'
setGraphs(.Object, graphs)
```

```
addGraph(.Object, amGraph = NULL, ...)  
  
## S4 method for signature 'AmChart,AmGraphOrMissing'  
addGraph(.Object, amGraph = NULL, ...)  
  
## S4 method for signature 'AmChart,AmGraphOrMissing'  
setGraph(.Object, graph = NULL, ...)  
  
setGuides(.Object, guides)  
  
## S4 method for signature 'AmChart,list'  
setGuides(.Object, guides)  
  
## S4 method for signature 'AmChart,GuideOrMissing'  
addGuide(.Object, guide = NULL, ...)  
  
setLegend(.Object, amLegend = NULL, ...)  
  
## S4 method for signature 'AmChart,AmLegendOrMissing'  
setLegend(.Object, amLegend = NULL, ...)  
  
addSegment(.Object, categoryIDs, sgts)  
  
## S4 method for signature 'AmChart,numeric'  
addSegment(.Object, categoryIDs, sgts)  
  
addSubData(.Object, categoryIDs, data)  
  
## S4 method for signature 'AmChart,numeric'  
addSubData(.Object, categoryIDs, data)  
  
setSubChartProperties(.Object, .subObject = NULL, ...)  
  
## S4 method for signature 'AmChart'  
setSubChartProperties(.Object, .subObject = NULL, ...)  
  
setTheme(.Object, theme)  
  
## S4 method for signature 'AmChart,character'  
setTheme(.Object, theme)  
  
setTitles(.Object, titles)  
  
## S4 method for signature 'AmChart,list'  
setTitles(.Object, titles)  
  
addTitle(.Object, title = NULL, ...)
```

```

## S4 method for signature 'AmChart,TitleOrMissing'
addTitle(.Object, title = NULL, ...)

setTrendLines(.Object, trendLines)

## S4 method for signature 'AmChart,list'
setTrendLines(.Object, trendLines)

addTrendLine(.Object, trendLine = NULL, ...)

## S4 method for signature 'AmChart,TrendLineOrMissing'
addTrendLine(.Object, trendLine = NULL, ...)

## S4 method for signature 'AmChart,character'
setType(.Object, type)

setValueAxes(.Object, valueAxes)

## S4 method for signature 'AmChart,list'
setValueAxes(.Object, valueAxes)

addValueAxes(.Object, valueAxis = NULL, ...)

## S4 method for signature 'AmChart,ValueAxisOrMissing'
addValueAxes(.Object, valueAxis = NULL, ...)

addValueAxis(.Object, valueAxis = NULL, ...)

## S4 method for signature 'AmChart,ValueAxisOrMissing'
addValueAxis(.Object, valueAxis = NULL, ...)

## S4 method for signature 'AmChart,ValueAxisOrMissing'
setValueAxis(.Object, valueAxis = NULL, ...)

setValueScrollbar(.Object, valueScrollbar, ...)

## S4 method for signature 'AmChart,ChartScrollbarOrMissing'
setValueScrollbar(.Object, valueScrollbar, ...)

```

## Arguments

.Object	<a href="#">AmChart</a> .
allLabels	list of <a href="#">Label</a> . Example of a label object, with all possible properties: label(x = 20, y = 20, text = "this is a label", align = "left", size = 12, color = "#CC0000", alpha = 1, rotation = 0, bold = TRUE, url = "http://www.amcharts.com"). Run <code>api("Label")</code> for more informations.

arrows	list of <a href="#">GaugeArrow</a> . Only valid for gauge charts. Run <code>api("GaugeArrow")</code> for more informations.
axes	list of <a href="#">GaugeAxis</a> properties. Only valid for gauge charts. Run <code>api("GaugeAxis")</code> for more informations.
balloon	<a href="#">AmBalloon</a> . Creates the balloons (tooltips) of the chart. It follows the mouse cursor when you roll-over the data items. The framework automatically generates the instances you just have to adjust the appearance to your needs. Run <code>api("AmBalloon")</code> for more informations.
categoryAxis	<a href="#">CategoryAxis</a> . Read-only. Chart creates category axis itself. If you want to change some properties, you should get this axis from the chart and set properties to this object. Run <code>api("CategoryAxis")</code> for more informations.
categoryField	character, category field name indicates the name of the field in your <code>dataProvider</code> object which will be used for category axis values.
chartCursor	<a href="#">ChartCursor</a> . Chart's cursor. Run <code>api("ChartCursor")</code> for more informations.
chartScrollbar	<a href="#">ChartScrollbar</a> . Chart's scrollbar. Run <code>api("ChartScrollbar")</code> for more informations.
creditsPosition	character, specifies position of the amCharts' website link. Allowed values are: "top-left", "top-right", "bottom-left" and "bottom-right".
dataProvider	<code>data.frame</code> , containing the data.
graphs	list of <a href="#">AmGraph</a> . Creates the visualization of the data in following types: line, column, step line, smoothed line, olhc and candlestick. Run <code>api("AmGraph")</code> for more informations.
graph	<a href="#">AmGraph</a> . Only valid for Gantt charts. Gant chart actually creates multiple graphs (separate for each segment). Properties of this graph are passed to each of the created graphs - this allows you to control the look of segments. Run <code>api("AmGraph")</code> for more informations.
guides	list of <a href="#">Guide</a> . Instead of adding guides to the axes, you can push all of them to this array. In case guide has category or date defined, it will automatically be assigned to the category axis, otherwise to the first value axis, unless you specify a different <code>valueAxes</code> for the guide. Run <code>api("Guide")</code> for more informations.
legend	<a href="#">AmLegend</a> . Legend of a chart. Run <code>api("AmLegend")</code> for more informations.
segmentsField	character, segments field in your data provider. Only valid for Gantt Charts.
theme	character, theme of a chart. Config files of themes can be found in <code>amcharts/themes/</code> folder. See <a href="https://www.amcharts.com/docs/v3/tutorials/working-with-themes/">https://www.amcharts.com/docs/v3/tutorials/working-with-themes/</a> .
titles	list of <a href="#">Title</a> . Run <code>api("Title")</code> for more informations.
trendLines	list of <a href="#">TrendLine</a> objects added to the chart. You can add trend lines to a chart using this list or access already existing trend lines. Run <code>api("TrendLine")</code> for more informations.
type	character, possible types are: "serial", "pie", "radar", "xy", "radar", "funnel", "gauge", "stock". See details about using argument type. (type map is in development).

valueAxes	list of <a href="#">ValueAxis</a> . Chart creates one value axis automatically, so if you need only one value axis, you don't need to create it. Run <code>api("ValueAxis")</code> for more informations.
valueAxis	<a href="#">ValueAxis</a> . Only valid for Gantt Charts. Set it's type to "date" if your data is date or time based. Run <code>api("ValueAxis")</code> for more informations.
valueScrollbar	<a href="#">ChartScrollbar</a> . Value scrollbar, enables scrolling value axes.
...	In case of constructor <code>new("AmChart")</code> or <code>amChart()</code> Dots represent other properties to set to the <a href="#">AmChart</a> object. See <a href="https://docs.amcharts.com/3/javascriptstockchart/AmChart">https://docs.amcharts.com/3/javascriptstockchart/AmChart</a> . In case of setters, dots represent properties of the object to add. See examples.
marginLeft	character, left margin of the chart.
marginRight	character, right margin of the chart.
label	(optional) <a href="#">Label</a> . Argument of method <code>addLabel</code> .
arrow	(optional) <a href="#">GaugeArrow</a> . Argument of method <code>addArrow</code> .
axe	(optional) <a href="#">GaugeAxis</a> . Argument of deprecated method <code>addAxe</code> .
axis	(optional) <a href="#">GaugeAxis</a> . same as <code>axe</code> .
amBalloon	<a href="#">AmBalloon</a> , argument of method <code>'setBalloon'</code> .
url	character.
format	character.
keepNA	object of class <code>logical</code> , default <code>TRUE</code> . Indicates if <code>NULL</code> values have to be kept or ignored.
amGraph	(optional) <a href="#">AmGraph</a> .
guide	(optional) <a href="#">Guide</a> . Argument of method <code>addGuide</code> .
amLegend	(optional) <a href="#">AmLegend</a> .
categoryIDs	numeric, see details.
sgts	<code>data.frame</code> ( or list of <code>data.frame</code> for multiple add ).
data	<code>data.frame</code> . Data to draw at the second level, after clicking on the column.
.subObject	<a href="#">AmChart</a> .
title	(optional) <a href="#">Title</a> , argument of method <code>addTitle</code> .
trendLine	(optional) <a href="#">TrendLine</a> . Argument of method <code>addTrendLine</code> .

## Details

`amAngularGaugeChart` is a shortcut for instantiating `AmChart` of type `gauge`.  
`amFunnelChart` is a shortcut for instantiating `AmChart` of type `funnel`.  
`amRadarChart` is a shortcut for instantiating `AmChart` of type `radar`.  
`amSerialChart` is a shortcut constructor for instantiating `AmChart` of type `serial`.  
`amPieChart` is a shortcut constructor for instantiating `AmChart` of type `pie`.  
`amGanttChart` is a constructor for instantiating `AmChart` of type `ganttt`.  
`amXYChart` is a shortcut constructor for instantiating `AmChart` of type `xy`.

Method 'addAxe' is deprecated, use 'addAxis'.

Method setGraph is only valid for Gantt Charts.

'addSubData' allows to add subdata for a chart with drilldown. In this case, categoryIDs indicates corresponding indice(s) of the dataProvider where to add the data.

For method addValueAxis: valueAxis is optional. Method addValueAxes is deprecated.

Method setValueAxis is only valid for Gantt charts.

## Value

(updated) [AmChart](#) with given properties.

## See Also

Refer to <https://docs.amcharts.com/3/javascriptcharts/>.

## Examples

```
new("AmChart", valueField = "value", theme = "patterns")
```

```
## Not run:  
amChart(type = "pie")
```

```
## End(Not run)  
## Not run:  
amAngularGaugeChart()
```

```
## End(Not run)  
## Not run:  
amFunnelChart(marginLeft = 15)
```

```
## End(Not run)  
## Not run:  
amRadarChart()
```

```
## End(Not run)  
## Not run:  
amSerialChart(creditsPostion = "top-right")
```

```
## End(Not run)  
## Not run:  
amPieChart()
```

```
## End(Not run)  
## Not run:  
amGanttChart(segmentsField = "segments")
```

```
## End(Not run)  
## Not run:  
amXYChart()
```

```
## End(Not run)
```

```

## Not run:
allLabels <- list(label(text = "balloonText"), label(text = "column"))
amSerialChart(allLabels = allLabels)

## End(Not run)
# ---
## Not run:
addLabel(.Object = amSerialChart(), text = "balloonText")
# equivalent to:
label_obj <- label(text = "balloonText")
addLabel(.Object = amSerialChart(), label = label_obj)

## End(Not run)
# ---
## Not run:
arrows_ls <- list(gaugeArrow(value = 130), gaugeArrow(value = 150))
amAngularGaugeChart(arrows = arrows_ls)

## End(Not run)
# ---
## Not run:
chart <- addArrow(.Object = amAngularGaugeChart(), value = 10); print(chart)
# equivalent to:
gaugeArrow_obj <- gaugeArrow(value = 10)
addArrow(.Object = amAngularGaugeChart(), arrow = gaugeArrow_obj)

## End(Not run)
# ---
## Not run:
axes_ls <- list(gaugeAxis(value = 130), gaugeAxis(value = 150))
setAxes(.Object = amAngularGaugeChart(), axes = axes_ls)
# If possible, simplify your code by using the constructor:
amAngularGaugeChart(axes = axes_ls)

## End(Not run)
# ---
## Not run:
addAxis(.Object = amAngularGaugeChart(), startValue = 0, endValue = 100, valueInterval = 10)
# equivalent to:
gaugeAxis_obj <- gaugeAxis(startValue = 0, enValue = 100, valueInterval = 10)
addAxis(.Object = amAngularGaugeChart(), axis = gaugeAxis_obj)

## End(Not run)
# ---
## Not run:
setBalloon(.Object = amSerialChart(), adjustBorderColor = TRUE, fillColor = "#FFFFFF",
           color = "#000000", cornerRadius = 5)
# equivalent to:
amBalloon_obj <- amBalloon(adjustBorderColor = TRUE, fillColor = "#FFFFFF",
                          color = "#000000", cornerRadius = 5)
setBalloon(.Object = amSerialChart(), amBalloon = amBalloon_obj)

## End(Not run)

```



```

# ---
## Not run:
setCategoryAxis(.Object = amSerialChart(), gridPosition = "start")
# equivalent to:
categoryAxis_obj <- categoryAxis(gridPosition = "start")
setCategoryAxis(.Object = amSerialChart(), categoryAxis = categoryAxis_obj)

## End(Not run)
# ---
setCategoryField(.Object = amSerialChart(), categoryField = "country")
# ---
## Not run:
# with default value, no argument needed
setChartCursor(.Object = amSerialChart())
# other example
setChartCursor(.Object = amSerialChart(), oneBallOnly = TRUE)
# equivalent to
chartCursor_obj <- chartCursor(oneBallOnly = TRUE)
setChartCursor(.Object = amSerialChart(), chartCursor = chartCursor_obj)

## End(Not run)
# ---
## Not run:
# Add the default scrollbar
setChartScrollbar(.Object = amSerialChart())
# equivalent to:
chartScrollbar_obj <- chartScrollbar(updateOnReleaseOnly = FALSE)
setChartScrollbar(.Object = amSerialChart(), chartScrollbar = chartScrollbar_obj)

## End(Not run)
# ---
## Not run:
setCreditsPosition(.Object = amPieChart(), creditsPosition = "top-right")

## End(Not run)
# ---
## Not run:
setDataLoader(.Object = amSerialChart(), url = "data.json", format = "json")

## End(Not run)
# ---
## Not run:
dataProvider_obj <- data.frame(key = c("FR", "US", "GER", "ENG", "IT" ),
                              value = round(runif(5, max = 100)))
setDataProvider(.Object = amPieChart(), dataProvider = dataProvider_obj)

## End(Not run)
# ---
## Not run:
graphs_ls <- list(graph(balloonText = "balloonText"), graph(type = "column"))
setGraphs(.Object = amSerialChart(), graphs = graphs_ls)

## End(Not run)

```

```

# ---
## Not run:
addGraph(.Object = amSerialChart(), balloonText = "balloonText", "type" = "column")
# equivalent to
amGraph_obj <- amGraph(balloonText = "balloonText", "type" = "column")
addGraph(.Object = amSerialChart(), amGraph = amGraph_obj)

## End(Not run)
# ---
## Not run:
print(setGraph(.Object = amGanttChart(), id = "amGraph-1"))
# equivalent to:
amGraph_obj <- amGraph(id = "amGraph-1")
setGraph(.Object = amGanttChart(), amGraph = amGraph_obj)

## End(Not run)
# ---
## Not run:
guides_ls <- list(guide(fillAlpha = .1), guide(fillAlpha = .5))
amSerialChart(guides = guides_ls)

## End(Not run)
# ---
## Not run:
chart <- addGuide(.Object = amSerialChart(), fillAlpha = .1, value = 0, toValue = 10)
print(chart)
# equivalent to
guide_obj <- guide(fillAlpha = .1, value = 0, toValue = 10, valueAxis = "1")
addGuide(.Object = amSerialChart(), guide = guide_obj)

## End(Not run)
## Not run:
setLegend(.Object = amChart(), amLegend = amLegend(useGraphSettings = TRUE))
# equivalent to:
setLegend(.Object = amChart(), useGraphSettings = TRUE)

## End(Not run)
# ---
## Not run:
pipeR::pipeline(
  amGanttChart(segmentsField = "segments"),
  setDataProvider(data.frame(category = c("John", "Julia"))),
  addSegment(1, data.frame(start = 7, duration = 2:3, task = c("Task #1", "Task #2"))),
  addSegment(2, data.frame(start = 10, duration = 2:3, task = c("Task #1", "Task #2")))
)
# ---
ls <- list(data.frame(start = 7, duration = 2:3, task = c("Task #1", "Task #2")),
  data.frame(start = 10, duration = 2:3, task = c("Task #1", "Task #2")))
pipeR::pipeline(
  amGanttChart(segmentsField = "segments"),
  setDataProvider(data.frame(category = c("John", "Julia"))),
  addSegment(1:2, ls)
)

```

```
## End(Not run)
# ---
## Not run:
amChart_obj <- amChart(dataProvider = data.frame(a = 1:5, b = 6:10))
addSubData(.Object = amChart_obj, categoryIDs = 3, data = data.frame(a = 1:10, b = 11:20))

## End(Not run)
# ---
## Not run:
setSubChartProperties(.Object = amSerialChart(), type = "serial")

## End(Not run)
# ---
## Not run:
setTheme(.Object = amPieChart(), theme = "dark")

## End(Not run)
# ---
## Not run:
titles_ls <- list(amTitle(text = "balloonText"), amTitle(text = "column"))
setTitles(.Object = amXYChart(), titles = titles_ls)
# or...
amXYChart(titles = titles_ls)

## End(Not run)
# ---
## Not run:
addTitle(.Object = amPieChart(), text = "balloonText", size = 15)
# equivalent to
title_obj <- amTitle(text = "balloonText", size = 15)
addTitle(.Object = amPieChart(), title = title_obj)

## End(Not run)
# ---
## Not run:
trendLines <- list(trendLine(initialValue = 1, finalValue = 5),
                  trendLine(initialValue = 7, finalValue = 19))
setTrendLines(.Object = amSerialChart(), trendLines = trendLines)
# or...
amSerialChart(trendLines = trendLines) # Equivalent

## End(Not run)
# ---
## Not run:
addTrendLine(.Object = amSerialChart(), initialValue = 1, initialXValue = 1,
             finalValue = 11, finalXValue = 12)
# equivalent to:
trendLine_obj <- trendLine(initialValue = 1, initialXValue = 1, finalValue = 11, finalXValue = 12)
chart <- addTrendLine(.Object = amSerialChart(), trendLine = trendLine_obj); print(chart)

## End(Not run)
# ---
```

```

## Not run:
setType(.Object = amChart(), type = "pie")
# equivalent to:
amPieChart()

## End(Not run)
## Not run:
valueAxes <- list(valueAxis(axisTitleOffset = 12, tickLength = 10),
                 valueAxis(axisTitleOffset = 10, tickLength = 10))
setValueAxes(.Object = amSerialChart(), valueAxes = valueAxes)
# or...
amSerialChart(valueAxes = valueAxes)

## End(Not run)
# ---
## Not run:
print(addValueAxis(.Object = amSerialChart(), axisTitleOffset = 12, tickLength = 10, title = "foo"))
# equivalent to:
valueAxis_obj <- valueAxis(axisTitleOffset = 12, tickLength = 10, title = "foo")
addValueAxis(.Object = amSerialChart(), valueAxis = valueAxis_obj)

## End(Not run)
# ---
## Not run:
setValueAxis(.Object = amGanttChart())
setValueAxis(.Object = amGanttChart(), type = "date")

## End(Not run)
## Not run:
valueScrollbar_obj <- chartScrollbar(updateOnReleaseOnly = FALSE)
chart <- setValueScrollbar(.Object = amSerialChart(), valueScrollbar = valueScrollbar_obj)
print(chart)
# or...
amSerialChart(updateOnReleaseOnly = FALSE)

## End(Not run)
# ---

```

---

```
initialize,AmGraph-method
```

*Initializes an AmGraph*

---

## Description

To create an AmGraph, you can use the usual method `Initialize` or the constructor. You can update properties with setters.

## Usage

```
## S4 method for signature 'AmGraph'
```

```

initialize(
  .Object,
  animationPlayed = FALSE,
  balloonText,
  title,
  type,
  valueField,
  ...
)

amGraph(animationPlayed = FALSE, balloonText, title, type, valueField, ...)

graph(animationPlayed = FALSE, balloonText, title, type, valueField, ...)

setBalloonText(.Object, balloonText)

## S4 method for signature 'AmGraph,character'
setBalloonText(.Object, balloonText)

## S4 method for signature 'AmGraph,character'
setTitle(.Object, title)

## S4 method for signature 'AmGraph,character'
setType(.Object, type)

setValueField(.Object, valueField)

## S4 method for signature 'AmGraph,character'
setValueField(.Object, valueField)

```

## Arguments

.Object	<a href="#">AmGraph</a> .
animationPlayed	logical.
balloonText	character, balloon text. You can use tags like <code>[[value]]</code> , <code>[[description]]</code> , <code>[[percents]]</code> , <code>[[open]]</code> , <code>[[category]]</code> or any other field name from your data provider. HTML tags can also be used.
title	character, graph title.
type	character, type of the graph. Possible values are: "line", "column", "step", "smoothedLine", "candlestick", "ohlc". XY and Radar charts can only display "line" otherArguments graphs.
valueField	character, name of the value field in your dataProvider.
...	other properties of AmGraph. See <a href="https://docs.amcharts.com/3/javascriptcharts/AmGraph">https://docs.amcharts.com/3/javascriptcharts/AmGraph</a> .

**Value**

An object of class [AmGraph](#) with the given properties.

**Examples**

```
# --- method 'initialize'
new("AmGraph", valueField = "value")

# constructor
amGraph(balloonText = "My text")
## Not run:
amGraph(balloonText = "balloonText", "type" = "column", title = "myGraph!",
        valueField = "value", animationPlayed = TRUE, other = TRUE)

## End(Not run)
amGraph(balloonText = "some text")
# --- shortcut constructor
graph(balloonText = "balloonText", "type" = "column",
      valueField = "value", animationPlayed = TRUE)

# --- update 'balloonText'
setBalloonText(.Object = amGraph(), balloonText = "performance")

# --- update 'title'
setTitle(.Object = amGraph(), title = "Power")

# --- update 'type'
setType(.Object = amGraph(), type = "type")

# --- update valueField
setValueField(.Object = amGraph(), valueField = "score")
```

---

initialize,AmLegend-method

*Initializes legend of the chart*

---

**Description**

Constructor for an AmLegend.

**Usage**

```
## S4 method for signature 'AmLegend'
initialize(.Object, useGraphSettings, ...)

amLegend(useGraphSettings, ...)

legend(useGraphSettings, ...)
```

```

setUseGraphSettings(.Object, useGraphSettings)

## S4 method for signature 'AmLegend,logical'
setUseGraphSettings(.Object, useGraphSettings)

```

### Arguments

.Object            [AmLegend](#).

useGraphSettings    logical, if TRUE, border color will be changed when user rolls-over the slice, graph, etc, instead of background color.

...                Other properties of [AmLegend](#). See <https://docs.amcharts.com/3/javascriptstockchart/AmLegend>.

### Examples

```

new("AmLegend", useGraphSettings = TRUE)
amLegend(useGraphSettings = FALSE)
rAmCharts:::legend(useGraphSettings = FALSE)
setUseGraphSettings(.Object = amLegend(), useGraphSettings = TRUE)

```

---

```

initialize,AmStockChart-method
                                  Initializes an AmStockChart

```

---

### Description

Method to initialize any S4 class provided by the package.

### Usage

```

## S4 method for signature 'AmStockChart'
initialize(
  .Object,
  balloon,
  comparedDataSets,
  dataSets,
  dataSetSelector,
  mainDataSet,
  panels,
  periodSelector,
  theme,
  group,
  is_ts_module,
  ...
)

```

```
amStockChart(  
    balloon,  
    comparedDataSets,  
    dataSets,  
    dataSetSelector,  
    mainDataSet,  
    panels,  
    periodSelector,  
    theme,  
    group,  
    is_ts_module,  
    ...  
)  
  
## S4 method for signature 'AmStockChart,AmBalloonOrMissing'  
setBalloon(.Object, amBalloon = NULL, ...)  
  
setCategoryAxesSettings(.Object, ...)  
  
## S4 method for signature 'AmStockChart'  
setCategoryAxesSettings(.Object, ...)  
  
setChartCursorSettings(.Object, ...)  
  
## S4 method for signature 'AmStockChart'  
setChartCursorSettings(.Object, ...)  
  
setChartScrollbarSettings(.Object, chartScrollbarSettings = NULL, ...)  
  
## S4 method for signature 'AmStockChart,ChartScrollbarOrMissing'  
setChartScrollbarSettings(.Object, chartScrollbarSettings = NULL, ...)  
  
setComparedDataSets(.Object, comparedDataSets)  
  
## S4 method for signature 'AmStockChart'  
setComparedDataSets(.Object, comparedDataSets)  
  
addComparedDataSet(.Object, dataSet = NULL, ...)  
  
## S4 method for signature 'AmStockChart,DataSetOrMissing'  
addComparedDataSet(.Object, dataSet = NULL, ...)  
  
setDataSets(.Object, dataSets)  
  
## S4 method for signature 'AmStockChart'  
setDataSets(.Object, dataSets)
```



```
addDataSet(.Object, dataSet = NULL, ...)  
  
## S4 method for signature 'AmStockChart,DataSetOrMissing'  
addDataSet(.Object, dataSet = NULL, ...)  
  
setDataSetSelector(.Object, dataSetSelector = NULL, ...)  
  
## S4 method for signature 'AmStockChart'  
setDataSetSelector(.Object, dataSetSelector = NULL, ...)  
  
setLegendSettings(.Object, ...)  
  
## S4 method for signature 'AmStockChart'  
setLegendSettings(.Object, ...)  
  
setMainDataSet(.Object, dataSet = NULL, ...)  
  
## S4 method for signature 'AmStockChart,DataSetOrMissing'  
setMainDataSet(.Object, dataSet = NULL, ...)  
  
setPanels(.Object, panels)  
  
## S4 method for signature 'AmStockChart,list'  
setPanels(.Object, panels)  
  
addPanel(.Object, panel = NULL, ...)  
  
## S4 method for signature 'AmStockChart,StockPanelOrMissing'  
addPanel(.Object, panel = NULL, ...)  
  
setPanelsSettings(.Object, ...)  
  
## S4 method for signature 'AmStockChart'  
setPanelsSettings(.Object, ...)  
  
setPeriodSelector(.Object, periodSelector = NULL, ...)  
  
## S4 method for signature 'AmStockChart,PeriodSelectorOrMissing'  
setPeriodSelector(.Object, periodSelector = NULL, ...)  
  
setStockEventsSettings(.Object, ...)  
  
## S4 method for signature 'AmStockChart'  
setStockEventsSettings(.Object, ...)  
  
setValueAxesSettings(.Object, ...)  
  
## S4 method for signature 'AmStockChart'
```

```
setValueAxesSettings(.Object, ...)
```

### Arguments

.Object	<a href="#">AmStockChart</a> .
balloon	<a href="#">AmBalloon</a> .
comparedDataSets	list of <a href="#">DataSet</a> . Properties of data sets selected for comparing.
dataSets	list of <a href="#">DataSet</a> . Each element must be a list of DataSet properties.
dataSetSelector	list of <a href="#">DataSetSelector</a> . You can add it if you have more than one data set and want users to be able to select/compare them.
mainDataSet	<a href="#">DataSet</a> . Data set selected as main.
panels	list of <a href="#">StockPanel</a> .
periodSelector	<a href="#">PeriodSelector</a> . You can add it if you want user's to be able to enter date ranges or zoom chart with predefined period buttons.
theme	character.
group	character for synchronization
is_ts_module	boolean. Don't use. For <a href="#">rAmChartsTimeSeriesUI</a>
...	other properties of AmStockChart.
amBalloon	<a href="#">AmBalloon</a> . Argument for method setBalloon.
chartScrollbarSettings	<a href="#">ChartScrollbar</a> . If you change a property after the chart is initialized, you should call <code>stockChart.validateNow()</code> method in order for it to work. If there is no default value specified, default value of ChartScrollbar class will be used.
dataSet	<a href="#">DataSet</a> .
panel	<a href="#">StockPanel</a> .

### Details

CategoryAxesSettings sets common settings for all CategoryAxes of StockPanels. If you change a property after the chart is initialized, you should call `stockChart.validateNow()` method. If there is no specified value, default value of CategoryAxis class will be used. you should get this axis from the chart and set properties to this object.

ChartCursorSettings sets settings for chart cursor. If you change a property after the chart is initialized, you should call `stockChart.validateNow()` method. If there is no specified value, default value of ChartCursor class will be used.

You can add it if you have more than one data set and want users to be able to select/compare them.

### Value

An object of class [AmStockChart](#).

**Examples**

```

## Not run:
# --- method 'initialize'
new("AmStockChart", theme = "dark")

## End(Not run)

## Not run:
# --- constructor
amStockChart()

## End(Not run)
library(pipeR)

## Not run:
# Dummy example
amStockChart() %>>% setBalloon(gridPosition = "start")

## End(Not run)
## Not run:
# Dummy example
setCategoryAxesSettings(.Object = amStockChart(), gridPosition = "start")

## End(Not run)
## Not run:
# Dummy example
setChartCursorSettings(.Object = amStockChart(), oneBallOnly = TRUE)

## End(Not run)
## Not run:
# Dummy example
amchart <- setChartScrollbarSettings(.Object = amStockChart(), enabled = TRUE)
print(amchart)

# equivalent to:
chartScrollbarSettings_obj <- chartScrollbarSettings()
setChartScrollbarSettings(.Object = amStockChart(),
                          chartScrollbarSettings = chartScrollbarSettings_obj)

## End(Not run)
## Not run:
# Dummy example
comparedDataSets_ls <- list(dataSet(compared = TRUE), dataSet(compared = TRUE))
setComparedDataSets(.Object = amStockChart(), comparedDataSets = comparedDataSets_ls)

## End(Not run)
## Not run:
# Dummy example
addComparedDataSet(.Object = amStockChart(), compared = TRUE)

## End(Not run)
## Not run:

```

```

# Dummy example
dataSets_ls <- list(dataSet(compared = FALSE), dataSet(compared = FALSE))
setDataSets(.Object = amStockChart(), dataSet = dataSets_ls)

## End(Not run)
## Not run:
# Dummy example
addDataSet(.Object = amStockChart(), compared = FALSE)
# equivalent to:
dataSet_obj <- dataSet(compared = FALSE)
addDataSet(.Object = amStockChart(), dataSet = dataSet_obj)

## End(Not run)
## Not run:
# Dummy example
print(setDataSetSelector(.Object = amStockChart(), width = 180))

# equivalent to:
dataSetSelector_obj <- dataSetSelector(width = 180)
print(setDataSetSelector(.Object = amStockChart(),
                        dataSetSelector = dataSetSelector_obj))

## End(Not run)
## Not run:
# Dummy example
setLegendSettings(.Object = amStockChart(), equalWidths = TRUE)

## End(Not run)
## Not run:
# Dummy example
setMainDataSet(.Object = amStockChart(), showInCompare = TRUE)

## End(Not run)
## Not run:
# Dummy example
panels_ls <- list(stockPanel(compared = TRUE), stockPanel(compared = TRUE))
setPanels(.Object = amStockChart(), panels = panels_ls)

## End(Not run)
## Not run:
# Dummy example
chart <- addPanel(.Object = amStockChart(), allowTurningOff = TRUE); print(chart)
# equivalent to:
panel_obj <- panel(allowTurningOff = TRUE)
addPanel(.Object = amStockChart(), panel = panel_obj)

## End(Not run)
## Not run:
# Dummy example
setPanelsSettings(.Object = amStockChart(), backgroundAlpha = 0)

## End(Not run)
## Not run:

```

```

# Dummy example
setPeriodSelector(.Object = amStockChart(), dateFormat = "DD-MM-YYYY")

## End(Not run)
## Not run:
# Dummy example
setStockEventsSettings(.Object = amStockChart(), backgroundAlpha = 1)

## End(Not run)
## Not run:
# Dummy example
setValueAxesSettings(.Object = amStockChart(), autoGridCount = "TRUE")

## End(Not run)

```

---

```

initialize,CategoryAxis-method
Initializes a CategoryAxis

```

---

## Description

Initializes or update a [CategoryAxis](#).

## Usage

```

## S4 method for signature 'CategoryAxis'
initialize(.Object, gridPosition, guides, ...)

categoryAxis(gridPosition, ...)

setGridPosition(.Object, gridPosition)

## S4 method for signature 'CategoryAxis,character'
setGridPosition(.Object, gridPosition)

```

## Arguments

.Object	<a href="#">CategoryAxis</a> .
gridPosition	character, specifies if a grid line is placed on the center of a cell or on the beginning of a cell. Possible values are: "start" and "middle" This setting doesn't work if parseDates is set to TRUE.
guides	list of <a href="#">Guide</a> .
...	Other properties.

**Examples**

```

guides <- list(guide(fillAlpha = .4, adjustBorderColor = TRUE),
              guide(fillAlpha = .4, adjustBorderColor = TRUE))
new("CategoryAxis", gridPosition = "start", gridThickness = 1, guides = guides)

## Not run:
new("CategoryAxis")
new("CategoryAxis", gridPosition = "start", 1) # 1 is not take into account

## End(Not run)

categoryAxis(gridPosition = "start", adjustBorderColor = TRUE)

setGridPosition(.Object = categoryAxis(), gridPosition = "start")

```

---

```

initialize,ChartCursor-method
      Initializes a ChartCursor

```

---

**Description**

Initializes or updates a [ChartCursor](#).

**Usage**

```

## S4 method for signature 'ChartCursor'
initialize(.Object, oneBalloonOnly, valueLineAxis, ...)

chartCursor(animationDuration = 0.3, oneBalloonOnly, valueLineAxis, ...)

setOneBalloonOnly(.Object, oneBalloonOnly)

## S4 method for signature 'ChartCursor,logical'
setOneBalloonOnly(.Object, oneBalloonOnly)

setValueLineAxis(.Object, valueLineAxis = NULL, ...)

## S4 method for signature 'ChartCursor,ValueAxisOrCharacterOrMissing'
setValueLineAxis(.Object, valueLineAxis = NULL, ...)

```

**Arguments**

.Object	<a href="#">ChartCursor</a> .
oneBalloonOnly	logical, if TRUE, border color will be changed when user rolls-over the slice, graph, etc, instead of background color.
valueLineAxis	<a href="#">ValueAxis</a> . If you set valueLineBalloonEnabled to TRUE, but you have more than one axis, you can use this property to indicate which axis should display balloon.

... other properties of ChartCursor. Run : api("ChartCursor") for more information.  
 animationDuration  
     numeric, duration of animation of a line, in seconds.

### Value

(updated) .Object of class [ChartCursor](#).

### Examples

```
new("ChartCursor", oneBalloonOnly = TRUE)

chartCursor()
chartCursor(oneBalloonOnly = TRUE)

setOneBalloonOnly(.Object = chartCursor(), oneBalloonOnly = TRUE)
setValueLineAxis(.Object = chartCursor(), id = "valueAxis1",
                 title = "Hello !", axisTitleOffset = 12)
# equivalent to:
valueLineAxis_obj <- valueAxis(id = "valueAxis1", title = "Hello !", axisTitleOffset = 12)
setValueLineAxis(.Object = chartCursor(), valueLineAxis = valueLineAxis_obj)
# or iff 'valueLineAxis_obj' has already been added to the chart:
setValueLineAxis(.Object = chartCursor(), valueLineAxis = "valueAxis1")
```

---

initialize,ChartScrollbar-method

*Initializes a ChartScrollbar*

---

### Description

ChartScrollbarSettings sets settings for chart scrollbar. If you change a property after the chart is initialized, you should call stockChart.validateNow() method. If there is no default value specified, default value of ChartScrollbar class will be used. Run api("ChartScrollbarSettings") for more informations.

### Usage

```
## S4 method for signature 'ChartScrollbar'
initialize(.Object, graph, enabled, ...)

chartScrollbar(graph, enabled = TRUE, ...)

chartScrollbarSettings(graph, enabled = TRUE, ...)

## S4 method for signature 'ChartScrollbar,AmGraphOrCharacterOrMissing'
setGraph(.Object, graph = NULL, ...)

setEnabled(.Object, enabled)
```

```
## S4 method for signature 'ChartScrollbar,logical'
setEnabled(.Object, enabled)
```

### Arguments

.Object	<a href="#">ChartScrollbar</a> .
graph	<a href="#">AmGraph</a> . Specifies which graph will be displayed in the scrollbar.
enabled	logical, specifies if the chart should be updated while dragging/resizing the scrollbar or only when user releases mouse button.
...	other properties of ChartScrollbar. Run : <code>api("ChartScrollbar")</code> for more information.

### Examples

```
new("ChartScrollbar", graph = "g1")
new("ChartScrollbar", graph = amGraph(test = 1))
chartScrollbar()
chartScrollbar(enabled = TRUE)
chartScrollbar()
chartScrollbar(enabled = TRUE)
# chartScrollbar with default graph
setGraph(.Object = chartScrollbar())

# example with arguments
setGraph(.Object = chartScrollbar(), id = "graph1", balloonText = "performance")
# equivalent to:
graph_obj <- amGraph(id = "graph1", balloonText = "performance")
(chartScrollbar_obj <- setGraph(.Object = chartScrollbar(), graph = graph_obj))
# or, iff graph_obj has already been added to the chart:
setGraph(.Object = chartScrollbar(), graph = "graph1")

# ---
setEnabled(.Object = chartScrollbar(), enabled = TRUE)
```

---

initialize,DataSet-method

*Creates or updates a DataSet*

---

### Description

Uses the constructors to create the object with its properties or updates an existing one with the setters.



**Usage**

```

## S4 method for signature 'DataSet'
initialize(
  .Object,
  compared = FALSE,
  dataProvider,
  fieldMappings,
  stockEvents,
  ...
)

dataSet(compared = FALSE, dataProvider, fieldMappings, stockEvents, ...)

## S4 method for signature 'DataSet,ANY,ANY'
setDataProvider(.Object, dataProvider, keepNA = TRUE)

setFieldMappings(.Object, fieldMappings)

## S4 method for signature 'DataSet,list'
setFieldMappings(.Object, fieldMappings)

addFieldMapping(.Object, ...)

## S4 method for signature 'DataSet'
addFieldMapping(.Object, ...)

setStockEvents(.Object, stockEvents)

## S4 method for signature 'DataSet,list'
setStockEvents(.Object, stockEvents)

addStockEvent(.Object, stockEvent = NULL, ...)

## S4 method for signature 'DataSet,StockEventOrMissing'
addStockEvent(.Object, stockEvent = NULL, ...)

```

**Arguments**

.Object	<a href="#">DataSet</a> .
compared	logical.
dataProvider	data.frame, the data set data. Important: the data sets need to come pre-ordered in ascending order. Data with incorrect order might result in visual and functional glitches on the chart.
fieldMappings	list of field mappings. Field mapping is an object with fromField and toField properties. fromField is a name of your value field in dataProvider. toField might be chosen freely, it will be used to set value/open/close/high/low fields for the StockGraph. Example: list(fromField = "val1", toField = "value").

stockEvents     [StockEvent](#).  
 ...             other properties of DataSet.  
 keepNA          logical, TRUE to keep NA values.  
 stockEvent      [StockEvent](#). Argument for method addStockEvent.

**Value**

(updated) [DataSet](#) object

**Examples**

```

new("DataSet")

dataSet(categoryField = "categoryField")

setDataProvider(.Object = dataSet(), data.frame(key = c("FR", "US"), value = c(20,10)))

dataset <- addFieldMapping(.Object = dataSet(),
                           fieldMappings = list(fromField = "val1", toField = "value"))
print(dataset)
dataset <- addFieldMapping(.Object = dataSet(), fromField = "val1", toField = "value")
print(dataset)
addStockEvent(.Object = dataSet(), backgroundAlpha = 1, backgroundColor = "#DADADA")
# equivalent to:
stockEvent_obj <- stockEvent(backgroundAlpha = 1, backgroundColor = "#DADADA")
chart <- addStockEvent(.Object = dataSet(), stockEvent = stockEvent_obj); print(chart)

```

---

initialize,DataSetSelector-method

*Creates or updates a DataSetSelector*

---

**Description**

Use the constructors to create the object with its properties or update an existing one with the setters.

**Usage**

```

## S4 method for signature 'DataSetSelector'
initialize(.Object, position, ...)

dataSetSelector(position, ...)

setPosition(.Object, position)

## S4 method for signature 'DataSetSelector,character'
setPosition(.Object, position)

```

**Arguments**

.Object	<a href="#">DataSetSelector</a> .
position	character. Possible values: "right", "left", "top", "bottom". "top" and "bottom" positions has a limitation - only one data set can be selected for comparison.
...	other properties of DataSetSelector.

**Value**

(updated) [DataSetSelector](#).

**Examples**

```
new("DataSetSelector", size = 10)

dataSetSelector(position = "left")
setPosition(.Object = dataSetSelector(), position = "left")
```

---

```
initialize,GaugeArrow-method
Initializes a GaugeArrow
```

---

**Description**

Uses the constructor to create the object with its properties or update an existing one with the setters.

**Usage**

```
## S4 method for signature 'GaugeArrow'
initialize(.Object, alpha = 1, axis, ...)

gaugeArrow(alpha = 1, axis, ...)

setAxis(.Object, axis = NULL, ...)

## S4 method for signature 'GaugeArrow,GaugeAxisOrCharacterOrMissing'
setAxis(.Object, axis = NULL, ...)
```

**Arguments**

.Object	<a href="#">GaugeArrow</a> .
alpha	numeric.
axis	<a href="#">GaugeAxis</a> . Axis of the arrow. You can use reference to the axis or id of the axis. If you don't set any axis, the first axis of the chart will be used.
...	other properties of GaugeArrow.

**Value**

(updated) .Object of class [GaugeArrow](#).

**Examples**

```
# --- method initialize
new("GaugeArrow", alpha = 2)

# --- constructor
gaugeArrow(value = 10)

# -- update 'axis' property
setAxis(.Object = gaugeArrow(), id = "axis1", startValue = 0,
        endValue = 100, valueInterval = 10)
# equivalent to:
axis_obj <- gaugeAxis(id = "axis1", startValue = 0, endValue = 100, valueInterval = 10)
setAxis(.Object = gaugeArrow(), axis = axis_obj)
# or, iff, 'axis_obj' has already been added to the chart
setAxis(.Object = gaugeArrow(), axis = "axis1")
```

---

initialize,GaugeAxis-method

*Initializes a GaugeAxis*

---

**Description**

Uses the constructor to create the object or update an existing one with the setters.

**Usage**

```
## S4 method for signature 'GaugeAxis'
initialize(.Object, axisAlpha = 1, bands, ...)

gaugeAxis(axisAlpha = 1, bands, ...)

setBands(.Object, bands)

## S4 method for signature 'GaugeAxis,list'
setBands(.Object, bands)

addBand(.Object, band = NULL, ...)

## S4 method for signature 'GaugeAxis,GaugeBandOrMissing'
addBand(.Object, band = NULL, ...)
```

**Arguments**

.Object	<a href="#">GaugeAxis</a> .
axisAlpha	numeric.
bands	list of <a href="#">GaugeBand</a> . Bands are used to draw color fills between specified values.
...	other properties of GaugeAxis.
band	<a href="#">GaugeBand</a> . Argument for method addBand.

**Examples**

```
# --- method initialize
new("GaugeAxis", alpha = 1)

# -- constructor
gaugeAxis()

# -- update 'bands' at once
bands <- list(gaugeBand(startValue = 70, endValue = 90),
             gaugeBand(startValue = 40, endValue = 60))
gaugeAxis(bands = bands)

# --- add 'band' one by one one
addBand(.Object = gaugeAxis(), startValue = 0, endValue = 100)
# equivalent to
gaugeBand_obj <- gaugeBand(startValue = 0, endValue = 100)
addBand(.Object = gaugeAxis(), band = gaugeBand_obj)
```

---

```
initialize,GaugeBand-method
```

*Initializes a GaugeBand*

---

**Description**

Uses the constructor to create the object or update an existing one with the setters.

**Usage**

```
## S4 method for signature 'GaugeBand'
initialize(.Object, alpha = 1, id, ...)

gaugeBand(alpha = 1, id, ...)

setID(.Object, id)

## S4 method for signature 'GaugeBand'
setID(.Object, id)
```

**Arguments**

.Object	<a href="#">GaugeBand</a> (or "GaugeBand" for initialize).
alpha	numeric.
id	character.
...	other properties of GaugeBand.

**Value**

(updated) .Object of class [GaugeBand](#).

**Examples**

```
# --- method 'initialize'
new("GaugeBand")

# --- constructor
gaugeBand(alpha = 2, id = "band2")

# --- set the 'id'
setID(.Object = gaugeBand(), id = "1")
```

---

initialize,Guide-method

*Initializes a Guide*

---

**Description**

Uses the constructor to create the object or update an existing one with the setters.

**Usage**

```
## S4 method for signature 'Guide'
initialize(.Object, fillAlpha, valueAxis, value, ...)

guide(fillAlpha, valueAxis, value, ...)

setFillAlpha(.Object, fillAlpha)

## S4 method for signature 'Guide,numeric'
setFillAlpha(.Object, fillAlpha)

## S4 method for signature 'Guide,ValueAxisOrCharacterOrMissing'
setValueAxis(.Object, valueAxis = NULL, ...)
```

**Arguments**

.Object	<a href="#">Guide</a>
fillAlpha	numeric, specifies if a grid line is placed on the center of a cell or on the beginning of a cell. Possible values are: "start" and "middle" This setting doesn't work if parseDates is set to TRUE.
valueAxis	<a href="#">ValueAxis</a> class. As you can add guides directly to the chart, you might need to specify which value axis should be used.
value	numeric.
...	other properties of Guide.

**Examples**

```
# --- method initialize
new("Guide", fillAlpha = 0.1, gridThickness = 1, value = 1)

# --- constructor
guide(fillAlpha = .4, value = 1)
guide(fillAlpha = .4, adjustBorderColor = TRUE, gridThickness = 1)

setFillAlpha(.Object = guide(), fillAlpha = 1)
valueAxis_obj <- valueAxis(test = "foo")
setValueAxis(.Object = guide(), valueAxis = valueAxis_obj)
```

---

```
initialize,Label-method
```

*Initializes Label*

---

**Description**

Uses the constructor to create the object or update an existing one with the setters.

**Usage**

```
## S4 method for signature 'Label'
initialize(.Object, text, bold, x, y, ...)

label(text, bold, x, y, ...)

setBold(.Object, bold)

## S4 method for signature 'Label,logical'
setBold(.Object, bold)

## S4 method for signature 'Label,character'
setText(.Object, text)
```

```
setX(.Object, x)

## S4 method for signature 'Label,numericOrCharacter'
setX(.Object, x)

setY(.Object, y)

## S4 method for signature 'Label,numericOrCharacter'
setY(.Object, y)
```

### Arguments

.Object	<a href="#">Label</a> .
text	character, text of a title.
bold	character, specifies if label is bold or not.
x	numeric, label's x position.
y	numeric, label's y position.
...	other properties of <a href="#">Label</a> .

### Value

(updated) .Object of class [Label](#).

### Examples

```
# --- method initialize
new("Label", x = 10)

# --- constructor
label(text = "bonjour")
label(text = "Male", x = "28%", y = "97%")

# --- update property 'bold'
setBold(.Object = label(), bold = TRUE)

# --- update 'text'
setText(.Object = label(), text = "Bonjour")

# --- update 'x'
setX(.Object = label(), x = 16)

# --- update 'y'
setY(.Object = label(), y = 16)
```



---

initialize,PeriodSelector-method  
*Initializes a PeriodSelector*

---

### Description

Uses the constructors to create the object with its properties or update an existing one with the setters.

### Usage

```
## S4 method for signature 'PeriodSelector'
initialize(.Object, periods, ...)

periodSelector(periods, ...)

addPeriod(.Object, ...)

## S4 method for signature 'PeriodSelector'
addPeriod(.Object, ...)
```

### Arguments

.Object	<a href="#">PeriodSelector</a> .
periods	list. Period object has 4 properties - period, count, label and selected. Possible period values are: "ss" - seconds, "mm" - minutes, "hh" - hours, "DD" - days, "MM" - months and "YYYY" - years. property "count" specifies how many periods this button will select. "label" will be displayed on a button and "selected" is a logical. which specifies if this button is selected when chart is initialized or not.
...	other properties of PeriodSelector.

### Value

(updated) .Object of class [PeriodSelector](#).

### Examples

```
new( "PeriodSelector")
periodSelector(fillAlpha = .4, value = 1)
periodSelector(fillAlpha = .4, adjustBorderColor = TRUE, gridThickness = 1)
addPeriod(.Object = periodSelector(), period = "MM", selected = TRUE,
          count = 1, label= "1 month")
```

---

```
initialize,StockEvent-method
      Initialize a StockEvent
```

---

### Description

Use the constructor to create the object or update an existing one with the setters.

### Usage

```
## S4 method for signature 'StockEvent'
initialize(.Object, backgroundAlpha = 1, stockGraph, ...)

stockEvent(backgroundAlpha = 1, stockGraph, ...)

setStockGraph(.Object, stockGraph = NULL, ...)

## S4 method for signature 'StockEvent,AmGraphOrCharacterOrMissing'
setStockGraph(.Object, stockGraph = NULL, ...)
```

### Arguments

.Object	<a href="#">StockEvent</a> .
backgroundAlpha	numeric.
stockGraph	<a href="#">AmGraph</a> created with <code>stockGraph(*)</code> . This is the graph on which event will be displayed. You can use a reference to the stock graph object or id of the graph.
...	other properties of <code>StockEvent</code> .

### Value

(updated) argument .Object of class [StockEvent](#).

### Examples

```
new("StockEvent")
stockEvent()
setStockGraph(.Object = stockEvent(), id = "stockGraph1", balloonText = "balloonText")
# equivalent to:
stockGraph_obj <- stockGraph(id = "stockGraph1", balloonText = "balloonText")
setStockGraph(.Object = stockEvent(), stockGraph = stockGraph_obj)
# if stockGraph_obj has already been added to the chart:
setStockGraph(.Object = stockEvent(), stockGraph = "stockGraph1")
```

---

initialize,StockPanel-method  
*Initialize a StockPanel*

---

### Description

Use the constructor to create the object or update an existing one with the setters.

### Usage

```
## S4 method for signature 'StockPanel'
initialize(
  .Object,
  allLabels,
  axes,
  balloon,
  categoryAxis,
  categoryField,
  chartCursor,
  chartScrollbar,
  creditsPosition,
  dataProvider,
  graphs,
  graph,
  guides,
  legend,
  theme,
  title,
  titles,
  trendLines,
  type,
  valueAxes,
  valueScrollbar,
  drawOnAxis,
  stockGraphs,
  stockLegend,
  ...
)

stockPanel(...)

panel(...)

setDrawOnAxis(.Object, valueAxis = NULL, ...)

## S4 method for signature 'StockPanel,ValueAxisOrCharacterOrMissing'
setDrawOnAxis(.Object, valueAxis = NULL, ...)
```

```

setStockGraphs(.Object, stockGraphs)

## S4 method for signature 'StockPanel,list'
setStockGraphs(.Object, stockGraphs)

addStockGraph(.Object, stockGraph = NULL, ...)

## S4 method for signature 'StockPanel,AmGraphOrMissing'
addStockGraph(.Object, stockGraph = NULL, ...)

setStockLegend(.Object, stockLegend = NULL, ...)

## S4 method for signature 'StockPanel,AmLegendOrMissing'
setStockLegend(.Object, stockLegend = NULL, ...)

```

### Arguments

.Object	<a href="#">StockPanel</a> .
allLabels	list of <a href="#">Label</a> . Example of a label object, with all possible properties: label(x = 20, y = 20, text = "this is a label", align = "left", size = 12, color = "#CC0000", alpha = 1, rotation = 0, bold = TRUE, url = "http://www.amcharts.com"). Run <code>api("Label")</code> for more informations.
axes	list of <a href="#">GaugeAxis</a> properties. Only valid for gauge charts. Run <code>api("GaugeAxis")</code> for more informations.
balloon	<a href="#">AmBalloon</a> . Creates the balloons (tooltips) of the chart. It follows the mouse cursor when you roll-over the data items. The framework automatically generates the instances you just have to adjust the appearance to your needs. Run <code>api("AmBalloon")</code> for more informations.
categoryAxis	<a href="#">CategoryAxis</a> . Read-only. Chart creates category axis itself. If you want to change some properties, you should get this axis from the chart and set properties to this object. Run <code>api("CategoryAxis")</code> for more informations.
categoryField	character, category field name indicates the name of the field in your dataProvider object which will be used for category axis values.
chartCursor	<a href="#">ChartCursor</a> . Chart's cursor. Run <code>api("ChartCursor")</code> for more informations.
chartScrollbar	<a href="#">ChartScrollbar</a> . Chart's scrollbar. Run <code>api("ChartScrollbar")</code> for more informations.
creditsPosition	character, specifies position of the amCharts' website link. Allowed values are: "top-left", "top-right", "bottom-left" and "bottom-right".
dataProvider	data.frame, containing the data.
graphs	list of <a href="#">AmGraph</a> . Creates the visualization of the data in following types: line, column, step line, smoothed line, olhc and candlestick. Run <code>api("AmGraph")</code> for more informations.
graph	<a href="#">AmGraph</a> . Only valid for Gantt charts. Gant chart actually creates multiple graphs (separate for each segment). Properties of this graph are passed to each

	of the created graphs - this allows you to control the look of segments. Run <code>api("AmGraph")</code> for more informations.
<code>guides</code>	list of <a href="#">Guide</a> . Instead of adding guides to the axes, you can push all of them to this array. In case guide has category or date defined, it will automatically be assigned to the category axis, otherwise to the first value axis, unless you specify a different valueAxes for the guide. Run <code>api("Guide")</code> for more informations.
<code>legend</code>	<a href="#">AmLegend</a> . Legend of a chart. Run <code>api("AmLegend")</code> for more informations.
<code>theme</code>	character, theme of a chart. Config files of themes can be found in <code>amcharts/themes/</code> folder. See <a href="https://www.amcharts.com/docs/v3/tutorials/working-with-themes/">https://www.amcharts.com/docs/v3/tutorials/working-with-themes/</a> .
<code>title</code>	A title of a panel. Note, <code>StockLegend</code> should be added in order title to be displayed.
<code>titles</code>	list of <a href="#">Title</a> . Run <code>api("Title")</code> for more informations.
<code>trendLines</code>	list of <a href="#">TrendLine</a> objects added to the chart. You can add trend lines to a chart using this list or access already existing trend lines. Run <code>api("TrendLine")</code> for more informations.
<code>type</code>	character, possible types are: "serial", "pie", "radar", "xy", "radar", "funnel", "gauge", "stock". See details about using argument type. (type map is in development).
<code>valueAxes</code>	list of <a href="#">ValueAxis</a> . Chart creates one value axis automatically, so if you need only one value axis, you don't need to create it. Run <code>api("ValueAxis")</code> for more informations.
<code>valueScrollbar</code>	<a href="#">ChartScrollbar</a> . Value scrollbar, enables scrolling value axes.
<code>drawOnAxis</code>	<a href="#">ValueAxis</a> . Specifies on which value axis user can draw trend lines. Set <code>drawingIconsEnabled</code> to true if you want drawing icons to be visible. First value axis will be used if not set here. You can use a reference to the value axis object or id of value axis.
<code>stockGraphs</code>	list of <a href="#">AmGraph</a> . Each element must be have been created with <code>stockGraph(*)</code>
<code>stockLegend</code>	list of <a href="#">AmLegend</a> . Each element must be have been created with <code>stockLegend(*)</code>
<code>...</code>	other properties of <code>StockPanel</code> .
<code>valueAxis</code>	A <a href="#">ValueAxis</a> for the property 'drawnOnAxis'.
<code>stockGraph</code>	<a href="#">AmGraph</a> , created with <code>stockGraph(...)</code> . Argument for method <code>addStockGraph</code> .

**Value**

(updated) [StockPanel](#) with given properties.

**Examples**

```
new("StockPanel", title = "Volume")

stockPanel(stockLegend = amLegend(useGraphSettings = TRUE))
panel(creditsPosition = "top-right")
panel(title = "top-right")
```

```

valueAxis_obj <- valueAxis(id = "valueAxis1")
setDrawOnAxis(.Object = stockPanel(), valueAxis = valueAxis_obj)
setDrawOnAxis(.Object = stockPanel(), valueAxis = "valueAxis1")
# ---
stockGraphs <- list(stockGraph(comparable = TRUE), stockGraph(comparable = FALSE))
setStockGraphs(.Object = stockPanel(), stockGraphs = stockGraphs)
stockPanel(stockGraphs = stockGraphs)
# ---
stock_panel <- addStockGraph(.Object = stockPanel(), comparable = FALSE); print(stock_panel)
# or...
stock_panel <- addStockGraph(.Object = stockPanel(), stockGraph = stockGraph(comparable = FALSE))
# ---
setStockLegend(.Object = stockPanel(), valueTextRegular = "[[value]]")
# equivalent to:
stockLegend_obj <- stockLegend(valueTextRegular = "[[value]]")
setStockLegend(.Object = stockPanel(), stockLegend = stockLegend_obj)
# ---

```

---

```
initialize,Title-method
```

*Initializes A Title*

---

## Description

Uses the constructor to create the object or update an existing one with the setters.

## Usage

```

## S4 method for signature 'Title'
initialize(.Object, text, size, ...)

title(text, size, ...)

amTitle(text, size, ...)

## S4 method for signature 'Title,character'
setText(.Object, text)

setSize(.Object, size)

## S4 method for signature 'Title,numeric'
setSize(.Object, size)

```

## Arguments

.Object	<b>Title</b>
text	character, title text.
size	numeric, title size.
...	other properties of Title.

**Value**

(updated) [Title](#)

**Examples**

```
new("Title", size = 10)
rAmCharts:::title(text = "Main", size = 10)
rAmCharts:::title(text = "Main", bold = TRUE)
amTitle(text = "Main", size = 10)
amTitle(text = "Main", bold = TRUE)
setText(.Object = amTitle(), text = "Bonjour")
setSize(amTitle(), 16)
```

---

initialize,TrendLine-method

*Initializes a TrendLine*

---

**Description**

Uses the constructor to create the object or update an existing one with the setters.

**Usage**

```
## S4 method for signature 'TrendLine'
initialize(
  .Object,
  initialValue,
  initialXValue,
  finalValue,
  finalXValue,
  valueAxis,
  valueAxisX,
  ...
)

trendLine(
  .Object,
  initialValue,
  initialXValue,
  finalValue,
  finalXValue,
  valueAxis,
  valueAxisX,
  ...
)

setInitialValue(.Object, initialValue)
```

```

## S4 method for signature 'TrendLine,numeric'
setInitialValue(.Object, initialValue)

setInitialXValue(.Object, initialXValue)

## S4 method for signature 'TrendLine,numeric'
setInitialXValue(.Object, initialXValue)

setFinalValue(.Object, finalValue)

## S4 method for signature 'TrendLine,numeric'
setFinalValue(.Object, finalValue)

setFinalXValue(.Object, finalXValue)

## S4 method for signature 'TrendLine,numeric'
setFinalXValue(.Object, finalXValue)

## S4 method for signature 'TrendLine,ValueAxisOrCharacterOrMissing'
setValueAxis(.Object, valueAxis = NULL, ...)

setValueAxisX(.Object, valueAxisX = NULL, ...)

## S4 method for signature 'TrendLine,ValueAxisOrCharacterOrMissing'
setValueAxisX(.Object, valueAxisX = NULL, ...)

```

### Arguments

.Object	<a href="#">TrendLine</a> .
initialValue	numeric, value from which trend line should start.
initialXValue	numeric, used by XY chart only. X value from which trend line should start.
finalValue	numeric, value at which trend line should end.
finalXValue	numeric, used by XY chart only. X value at which trend line should end.
valueAxis	<a href="#">ValueAxis</a> . Value axis of the trend line. Will use first value axis of the chart if not set any. You can use a reference to the value axis object or id of value axis.
valueAxisX	<a href="#">ValueAxis</a> . Used by XY chart only. X axis of trend line. Will use first X axis of the chart if not set any. You can use a reference to the value axis object or id of value axis.
...	other properties of TrendLine.

### Value

(possibly updated) .Object of class [TrendLine](#).



**Examples**

```

new("TrendLine", initialValue = 1, finalValue = 11)

# Other example
valueAxis <- valueAxis(title = "Hello !", axisTitleOffset = 12)
new("TrendLine", valueAxis = valueAxis)

trendLine(initialValue = 1, finalValue = 11)
setInitialValue(.Object = trendLine(), initialValue = 16)
setInitialXValue(.Object = trendLine(), initialXValue = 16)
setFinalValue(.Object = trendLine(), finalValue = 16)
setFinalXValue(.Object = trendLine(), finalXValue = 16)
setValueAxis(.Object = trendLine(), id = "valueAxis-1",
             title = "Hello !", axisTitleOffset = 12)

# equival to:
valueAxis_obj <- valueAxis(id = "valueAxis-1", title = "Hello !", axisTitleOffset = 12)
trendLine(valueAxis = valueAxis_obj)
# or...
trendLine(valueAxis = "valueAxis-1")
# valid if and only if 'valueAxis_obj' has already been added to the chart

setValueAxisX(.Object = trendLine(), id = "valueAxisX-1",
              title = "Hello !", axisTitleOffset = 12)

# equival to:
valueAxisX_obj <- valueAxis(id = "valueAxisX-1", title = "Hello !", axisTitleOffset = 12)
trendLine(valueAxisX = valueAxisX_obj)
# or...
trendLine(valueAxisX = "valueAxisX-1")
# valid if and only if 'valueAxisX_obj' has already been added to the chart

```

---

initialize, ValueAxis-method

*Initializes ValueAxis*


---

**Description**

Creates a ValueAxis or updates its properties.

**Usage**

```

## S4 method for signature 'ValueAxis'
initialize(.Object, title, guides, ...)

valueAxis(...)

## S4 method for signature 'ValueAxis,character'
setTitle(.Object, title)

```

**Arguments**

.Object      [ValueAxis](#).  
 title        character.  
 guides      list of [Guide](#).  
 ...         Other properties (depend of call function)

**Examples**

```
guides <- list(guide(fillAlpha = .4), guide(fillAlpha = .5))
new("ValueAxis", title = "Hello !", gridThickness = 1, guides = guides)

valueAxis(title = "Hello !", axisTitleOffset = 12)

setTitle(.Object = valueAxis(), title = "Hello !")
```

---

 Label-class

*Label class*


---

**Description**

Creates a label on the chart which can be placed anywhere, multiple can be assigned.

**Details**

Run `api("Label")` for more information and all available properties.

**Slots**

`bold` character. Specifies if label is bold or not.

`text` character. Text of a title.

`x` numeric. X position of a label.

`y` numeric. Y position of a label.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

`otherProperties` list containing other available properties not yet implemented in the package.

`value` numeric.

**Author(s)**

datastorm-open

---

listProperties	<i>List properties of an S4 object</i>
----------------	--

---

**Description**

Each S4 class implements the method to list its properties (usefull to update complex properties).

**Usage**

```
listProperties(.Object)

## S4 method for signature 'AmObject'
listProperties(.Object)
```

**Arguments**

.Object            any class object of the package

**Value**

A list containing all the chart's properties.

**Examples**

```
amChart(type = "serial")
```

---

PeriodSelector-class	<i>PeriodSelector</i>
----------------------	-----------------------

---

**Description**

Defines the PeriodSelector properties.

**Slots**

**periods** list. Period object has 4 properties - period, count, label and selected. Possible period values are: "ss" - seconds, "mm" - minutes, "hh" - hours, "DD" - days, "MM" - months and "YYYY" - years. property "count" specifies how many periods this button will select. "label" will be displayed on a button and "selected" is logical. which specifies if this button is selected when chart is initialized or not.

**listeners** list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. See examples for details.

**otherProperties** list containing other available properties not yet implemented in the package.  
**value** Object of class numeric.

**Author(s)**

datastorm-open

---

plot,AmCharts-method *PLOTTING METHOD*

---

**Description**

Basic method to plot an AmChart

**Usage**

```
## S4 method for signature 'AmCharts'
plot(x, y, width = "100%", height = NULL, background = "#ffffff", ...)
```

**Arguments**

x	<a href="#">AmChart</a>
y	unused.
width	character.
height	character.B
background	character.
...	Other properties.

**Details**

Plots an object of class [AmChart](#)

---

print,AmObject-method *Visualize with print*

---

**Description**

Display the object in the console.

**Usage**

```
## S4 method for signature 'AmObject'
print(x, withDetail = TRUE, ...)
```

**Arguments**

x	<a href="#">AmChart</a> .
withDetail	logical, TRUE to display details.
...	Other properties.

**Details**

If the object possess a 'dataProvider' property, it will be hidden in the console. To see if it's correctly registered use '@dataProvider'.

**Examples**

```
print(new("AmChart", categoryField = "variables", type = "serial"))
print(new("AmChart", categoryField = "variables", type = "serial"), withDetail = FALSE)
```

---

rAmCharts-shinymodules

*Shiny module to export rAmCharts graphics on server-side*

---

**Description**

This function need the base64enc package to save image.

**Usage**

```
rAmChartsExportServerUI(id)

rAmChartsExportServer(
  input,
  output,
  session,
  list_am_graph,
  path = shiny::reactive(tempdir()),
  mode = "single",
  progress = T,
  message = "Calculation in progress",
  detail = "This may take a while..."
)
```

**Arguments**

id	character, used to specify namesapce, see shiny::NS
input	standard, shiny input
output	standard, shiny output
session	standard, shiny session
list_am_graph	named list, reactive expression with all amCharts to export <ul style="list-style-type: none"> <li>• "graph" rAmCharts object to export</li> <li>• "name" character, name of file, with ".jpg" extension</li> <li>• "width" Optionnal, character. Linked to <a href="#">amChartsOutput</a></li> <li>• "height" Optionnal, character. Linked to <a href="#">amChartsOutput</a></li> <li>• "type" Optionnal, character. Linked to <a href="#">amChartsOutput</a></li> </ul>

path	character, directory. tmpdir() by Default
mode	character, 'single' : graphics are rendered and saved one by one. 'multiple' all at same time
progress	boolean, set a progress bar or not ?
message	character, if progress, message. Default to "Calculation in progress"
detail	character, if progress, detail. Default to "This may take a while...'

**Value**

a reactive expression

**Examples**

```
## Not run:

# ui
rAmChartsExportServerUI("export_server_graphs")

# server

mult_amgraph <- reactive({
  if(input$goSave > 0){
    isolate({
      list(
        list(graph = amPie(data = data_pie), name = "pie.jpg", height = "200px", width = "300px"),
        list(graph = amBarplot(x = "country", y = "visits", data = data_bar, main = "example") %>%
          setExport(), name = "bar.jpg", height = "600px")
      )
    })
  } else {
    NULL
  }
})

callModule(rAmChartsExportServer, "export_server_graphs", mult_amgraph,
  reactive("/home/benoit/amchart_export"))

## End(Not run)
```

---

rAmCharts-shinymodules-ts

*Shiny module to render large time-series data with live server-client aggregation*

---

**Description**

Shiny module to render large time-series data with live server-client aggregation

**Usage**

```

rAmChartsTimeSeriesUI(id, width = "100%", height = "400px")

rAmChartsTimeSeriesServer(
  input,
  output,
  session,
  data,
  col_date,
  col_series,
  maxPoints = shiny::reactive(600),
  tz = shiny::reactive("UTC"),
  ts = shiny::reactive(c("5 min", "10 min", "30 min", "hour", "3 hour", "12 hour", "day",
    "week", "month", "year")),
  fun_aggr = shiny::reactive("mean"),
  treat_missing = shiny::reactive(FALSE),
  maxgap = shiny::reactive(Inf),
  type_aggr = shiny::reactive("first"),
  na.rm = shiny::reactive(TRUE),
  main = shiny::reactive(""),
  ylab = shiny::reactive(""),
  color = shiny::reactive(c("#2E2EFE", "#31B404", "#FF4000", "#AEB404")),
  type = shiny::reactive(c("line")),
  bullet = shiny::reactive(NULL),
  bulletSize = shiny::reactive(2),
  linetype = shiny::reactive(c(0, 5, 10, 15, 20)),
  linewidth = shiny::reactive(c(1, 1, 1, 1, 1, 1)),
  fillAlphas = shiny::reactive(0),
  precision = shiny::reactive(1),
  connect = shiny::reactive(FALSE),
  export = shiny::reactive(FALSE),
  legend = shiny::reactive(TRUE),
  legendPosition = shiny::reactive("bottom"),
  legendHidden = shiny::reactive(FALSE),
  ZoomButton = shiny::reactive(data.frame(Unit = "MAX", multiple = 1, label = "All")),
  ZoomButtonPosition = shiny::reactive("bottom"),
  periodFieldsSelection = shiny::reactive(FALSE),
  scrollbar = shiny::reactive(TRUE),
  scrollbarPosition = shiny::reactive("bottom"),
  scrollbarHeight = shiny::reactive(40),
  scrollbarGraph = shiny::reactive(NULL),
  cursor = shiny::reactive(TRUE),
  cursorValueBalloonsEnabled = shiny::reactive(TRUE),
  creditsPosition = shiny::reactive("top-right"),
  group = shiny::reactive(NULL),
  dataDateFormat = shiny::reactive("YYYY-MM-DD JJ:NN:ss"),
  categoryBalloonDateFormats = shiny::reactive(list(list(period = "YYYY", format =
    "YYYY"), list(period = "MM", format = "YYYY-MM"), list(period = "WW", format =

```

```

"YYYY-MM-DD"), list(period = "DD", format = "YYYY-MM-DD"), list(period = "hh", format
= "YYYY-MM-DD JJ:NN"), list(period = "mm", format = "YYYY-MM-DD JJ:NN"), list(period
= "ss", format = "YYYY-MM-DD JJ:NN:ss"), list(period = "fff", format =
"YYYY-MM-DD JJ:NN:ss))),
dateFormats = shiny::reactive(list(list(period = "YYYY", format = "YYYY"), list(period
= "MM", format = "MMM"), list(period = "WW", format = "MMM DD"), list(period = "DD",
format = "MMM DD"), list(period = "hh", format = "JJ:NN"), list(period = "mm", format
= "JJ:NN"), list(period = "ss", format = "JJ:NN:ss"), list(period = "fff", format =
"JJ:NN:ss"))),
thousandsSeparator = shiny::reactive(" "),
decimalSeparator = shiny::reactive("."),
balloonFontSize = shiny::reactive(10),
balloonMaxWidth = shiny::reactive(400)
)

```

### Arguments

id	character, used to specify namespace, see shiny::NS
width	character, the width of the chart container. For amChartsOutput.
height	character, the height of the chart container. For amChartsOutput.
input	standard, shiny input
output	standard, shiny output
session	standard, shiny session
data	: data.frame to transform.
col_date	Date column name, default to "date". Must be "POSIXct"
col_series	Column name of quantitative variable(s) to be transformed. Default to setdiff(colnames(data), "date")
maxPoints	: Maximal number of rows in results
tz	: Timezone of result. Default to "UTC".
ts	All enabled aggregation. Default to c("5 min", "10 min", "30 min", "hour", "3 hour", "12 hour", "day", "week", "month", "year"). Can be a number, in seconds, or a character string containing one of "min", "hour", "day".... This can optionally be preceded by a positive integer and a space
fun_aggr	: Aggregation function to use ("min", "max", "sum", "mean", "first", "last", "minabs", "maxabs"). Default to "mean".
treat_missing	: Boolean. Default to FALSE Whether or not to interpolate missing values ? see na.approx
maxgap	When interpolate missing values with na.approx. Maximum number of consecutive NAs to fill. Default to Inf.
type_aggr	character Type of aggregation <ul style="list-style-type: none"> <li>• "first" : Date/Time result is equal to minimum of sequence, and this minimum is included in aggregation</li> <li>• "last" : Date/Time result is equal to maximum of sequence, and this maximum is included in aggregation</li> </ul>



na.rm	: aggregation only. a logical value indicating whether NA values should be stripped before the computation proceeds.
main	character, title.
ylab	character, value axis label.
color	character, color of series (in hexadecimal).
type	character, Type of graph. Possible values are : "line" (default), "column", "step", "smoothedLine"
bullet	character, point shape. Possible values are : "diamond", "square", "bubble", "yError", "xError", "round", "triangleLeft", "triangleRight", "triangleUp"
bulletSize	numeric, size of bullet.
linetype	numeric, line type, 0 : solid, number : dashed length
linewidth	numeric, line width.
fillAlphas	numeric, fill. Between 0 (no fill) to 1.
precision	numeric, default set to 1.
connect	logical, default set to FALSE. Specifies whether to connect data points if data is missing.
export	logical, default set to FALSE. TRUE to display export feature.
legend	logical, enabled or not legend ? Defaut to TRUE.
legendPosition	character, legend position. Possible values are : "left", "right", "bottom", "top"
legendHidden	logical hide some series on rendering ? Defaut to FALSE
ZoomButton	data.frame, 3 or 4 columns : <ul style="list-style-type: none"> <li>• "Unit" : Character. Times unit. 'ss', 'mm', 'hh', 'DD', 'MM', 'YYYY'</li> <li>• "multiple" : Numeric. multiple*unit</li> <li>• "label" : Character. button's label</li> <li>• "selected" : Boolean. Optional. To set initial selection. (One TRUE, others FALSE)</li> </ul>
ZoomButtonPosition	character, zoom button position. Possible values are : "left", "right", "bottom", "top"
periodFieldsSelection	boolean, using zoom button, add also two fields to select period ?
scrollbar	boolean, enabled or not scrollbar ? Defaut to TRUE.
scrollbarPosition	character, scrollbar position. Possible values are : "left", "right", "bottom", "top"
scrollbarHeight	numeric, height of scroll bar. Default : 40.
scrollbarGraph	character, name of serie (column) to print in scrollbar. Defaut to NULL.
cursor	boolean, enabled or not cursor ? Defaut to TRUE.
cursorValueBalloonsEnabled	boolean, if cursor, enabled or not balloons on cursor ? Defaut to TRUE.

creditsPosition	character, credits position. Possible values are : "top-right", "top-left", "bottom-right", "bottom-left"
group	character, like in dygraphs, for synchronization in shiny or rmarkdown.
dataDateFormat	character Data date format. Default to 'YYYY-MM-DD JJ:NN:ss'. See <a href="#">amTimeSeries</a> .
categoryBalloonDateFormats	list Date format objects for chart cursor. See <a href="#">amTimeSeries</a> .
dateFormats	list Date format objects for x-axis. See <a href="#">amTimeSeries</a> .
thousandsSeparator	character, default set to " "
decimalSeparator	character, default set to ".",
balloonFontSize	numeric, text font size on balloon. Default : 10.
balloonMaxWidth	numeric. Default : 400.

### Value

a reactive expression with aggregate data and ts

### Examples

```
## Not run:

library(shiny)
library(rAmCharts)
library(data.table)

# number of points
n <- 1000000
data <- data.frame(date = seq(c(ISOdate(1999,12,31)), by = "5 min", length.out = n),
  value = rnorm(n, 100, 50), check.names = FALSE)

# maximun of points in javascript
max_points <- 1000

# Call module in UI
ui <- fluidPage(
  rAmChartsTimeSeriesUI("ts_1", height = "600px"),
  h4(textOutput("ts"))
)

# Define server
server <- function(input, output) {

  # Call module in server
  res <- callModule(rAmChartsTimeSeriesServer, "ts_1", reactive(data), reactive("date"),
    reactive("value"), maxPoints = shiny::reactive(max_points),
    main = reactive("Example of rAmChartsTimeSeries module"),
```

```

    color = reactive("red"), periodFieldsSelection = reactive(TRUE)
  )

  # show module return and print ts
  output$ts <- renderText({
    print(res())
    paste0("Current ts : ", res()$ts)
  })

}

# Run the application
shinyApp(ui = ui, server = server)

## End(Not run)

```

---

renderAmCharts	<i>SHINY</i>
----------------	--------------

---

### Description

Widget output function to use in Shiny.

### Usage

```
renderAmCharts(expr, env = parent.frame(), quoted = FALSE)
```

### Arguments

expr	an expression that generates an HTML widget.
env	the environment in which expr must be evaluated.
quoted	is expr a quoted expression (with quote()). This is useful if you want to save an expression into variable.

---

runExamples	<i>Run example with shiny</i>
-------------	-------------------------------

---

### Description

See some examples in a shiny web application. Both 'am' functions and basic functions are illustrated.

### Usage

```
runExamples()
```

**Examples**

```
## Not run:
if (interactive()) runExamples()

## End(Not run)
```

---

setExport

*Setters for AmChart and AmStockChart.*


---

**Description**

These methods can be used both for AmChart and AmStockChart. There are general for some first-level properties.

**Usage**

```
setExport(.Object, enabled = TRUE, ...)

## S4 method for signature 'AmCharts,logicalOrMissing'
setExport(.Object, enabled = TRUE, ...)

setResponsive(.Object, enabled = TRUE, ...)

## S4 method for signature 'AmCharts,logicalOrMissing'
setResponsive(.Object, enabled = TRUE, ...)
```

**Arguments**

.Object	<a href="#">AmChart</a> or <a href="#">AmStockChart</a> .
enabled	logical, TRUE to display the export button.
...	Other properties that can be used depending on the setter.

**Examples**

```
## Not run:
# Dummy examples
setExport(amPlot(1:10))
setExport(amStockChart())

## End(Not run)
## Not run:
# Dummy examples
setResponsive(amSerialChart())
setResponsive(amStockChart())

## End(Not run)
```

---

show, AmChart-method    *Visualize AmStockChart with show*

---

**Description**

Display the object in the console.

**Usage**

```
## S4 method for signature 'AmChart'
show(object)
```

**Arguments**

object            [AmChart](#).

**Value**

If the object has a valid type, it will plot the chart. If not the method will trigger the method 'print'.

---

show, AmObject-method    *Visualize with show*

---

**Description**

Display the object in the console.

**Usage**

```
## S4 method for signature 'AmObject'
show(object)
```

**Arguments**

object            [AmObject](#).

**Examples**

```
library(pipeR)
amPieChart(valueField = "value", titleField = "key", backgroundColor = "#7870E8") %>>%
  setDataProvider(data.frame(key = c("FR", "US"), value = c(20,10))) %>>%
  setExport(position = "bottom-left")
```

---

show, AmStockChart-method

*Visualize AmStockChart with show*

---

### Description

Display the object in the console.

### Usage

```
## S4 method for signature 'AmStockChart'
show(object)
```

### Arguments

object            [AmStockChart](#).

### Value

If the object has a valid type, it will plot the chart. If not the method will trigger the method 'print'.

---

StockEvent-class

*StockEvent class*

---

### Description

StockEvent is an object which holds information about event (bullet). Values from StockEventsSettings will be used if not set. Stock event bullet's size depends on it's graphs fontSize. When user rolls-over, clicks or rolls-out of the event bullet, AmStockChart dispatches events.

### Details

Run `api("StockEvent")` for more informations.

### Slots

stockGraph [AmGraph](#) containing properties of stockGraph. This is the graph on which event will be displayed. You can use a reference to the stock graph object or id of the graph.

listeners list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string.

otherProperties list containing other available properties not yet implemented in the package.  
value numeric.

### Author(s)

datastorm-open

---

stockGraph	<i>Constructor for a stockGraph (class AmGraph)</i>
------------	---

---

**Description**

Constructor used for [AmStockChart](#)

**Usage**

```
stockGraph(animationPlayed = FALSE, balloonText, title, type, valueField, ...)
```

**Arguments**

animationPlayed	logical.
balloonText	character. Balloon text. You can use tags like <code>[[value]]</code> , <code>[[description]]</code> , <code>[[percents]]</code> , <code>[[open]]</code> , <code>[[category]]</code> or any other field name from your data provider. HTML tags can also be used.
title	character. Graph title.
type	character. Type of the graph. Possible values are: "line", "column", "step", "smoothedLine", "candlestick", "ohlc". XY and Radar charts can only display "line" otherArguments graphs.
valueField	character. Name of the value field in your dataProvider.
...	Other properties

**Value**

An object of class [AmGraph](#).

**Examples**

```
# --- constructor
stockGraph(balloonText = "balloonText", valueField = "value", animationPlayed = TRUE)
```

---

stockLegend	<i>Constructor for StockLegend.</i>
-------------	-------------------------------------

---

**Description**

This method is used for [AmStockChart](#).

**Usage**

```
stockLegend(useGraphSettings, valueTextComparing = "[[percents.value]]%", ...)
```

**Arguments**

- useGraphSettings      logical Legend markers can mirror graph's settings, displaying a line and a real bullet as in the graph itself. Set this property to TRUE if you want to enable this feature.
- valueTextComparing    character
- ...                      Properties of AmLegend. See <https://docs.amcharts.com/3/javascriptstockchart/StockLegend>

**Value**

An [AmLegend](#) object

**Examples**

```
stockLegend(useGraphSettings = TRUE)
```

---

StockPanel-class

*StockPanel class*

---

**Description**

StockPanel class creates stock panels (charts). AmStockChart can have multiple Stock panels.

**Details**

Run `api("StockPanel")` for more information and all available properties.

**Fields**

- `drawOnAxis` [ValueAxis](#). Specifies on which value axis user can draw trend lines. Set `drawingIconsEnabled` to TRUE if you want icons to be visible. First value axis will be used if not set here. You can use a reference to the value axis object or id of value axis.
- `stockGraphs` list. Each element must be have been created with `stockGraph(*)`
- `stockLegend` list. Each element must be have been created with `stockLegend(*)`
- `allLabels` list of [Label](#). Example of label object, with all possible properties: `label(x = 20, y = 20, text = "this is label", align = "left", size = 12, color = "#CC0000", alpha = 1, rotation = 0, bold = TRUE, url = "http://www.amcharts.com")`. Run `api("Label")` for more informations.
- `arrows` list of [GaugeArrow](#). Only valid for gauge charts. Run `api("GaugeArrow")` for more informations.
- `axes` list of [GaugeAxis](#) properties. Only valid for gauge charts. Run `api("GaugeAxis")` for more informations.



**balloon** [AmBalloon](#) Creates the balloons (tooltips) of the chart. It follows the mouse cursor when you roll-over the data items. The framework automatically generates the instances you just have to adjust the appearance to your needs. Run `api("AmBalloon")` for more informations.

**categoryAxis** [CategoryAxis](#). Read-only. Chart creates category axis itself. If you want to change some properties, you should get this axis from the chart and set properties to this object.

**categoryField** character. Category field name indicates the chart the name of the field in your `dataProvider` object which will be used for category axis values.

**ChartCursor** [ChartCursor](#). Cursor of a chart. Run `api("ChartCursor")` for more informations.

**ChartScrollbar** [ChartScrollbar](#). Chart's scrollbar. Run `api("ChartScrollbar")` for more informations.

**creditsPosition** character, specifies position of the amCharts' website link. Allowed values are: "top-left", "top-right", "bottom-left" and "bottom-right".

**dataProvider** `data.frame`, containing the data.

**graphs** list of [AmGraph](#). Creates the visualization of the data in following types: line, column, step line, smoothed line, olhc and candlestick.

**graph** [AmGraph](#). Only valid for Gantt charts. Gantt chart actually creates multiple graphs (separate for each segment). Properties of this graph are passed to each of the created graphs - this allows you to control the look of segments. Run `api("AmGraph")` for more informations.

**guides** list of [Guide](#). Instead of adding guides to the axes, you can push all of them to this array. In case guide has category or date defined, it automatically will be assigned to the category axis. Otherwise to first value axis, unless you specify a different `valueAxes` for the guide. Run `api("Guide")` for more informations.

**legend** [AmLegend](#). Chart's legend. Run `api("AmLegend")` for more informations.

**segmentsField** character. Segments field in your data provider. Only valid for Gantt Charts.

**subChartProperties** list. Only valid for Drilldown charts.

**theme** character. Chart's theme. Config files of themes can be found in `amcharts/themes/` folder. See <https://www.amcharts.com/docs/v3/tutorials/working-with-themes/>.

**titles** list of [Title](#). Run `api("Title")` for more informations.

**trendLines** list of [TrendLine](#) objects added to a chart. You can add trend lines to a chart using this list or access already existing trend lines.

**type** character. Possible types are: "serial", "pie", "radar", "xy", "radar", "funnel", "gauge", "stock". See details about using argument type. (type map is in development).

**valueAxes** list of [ValueAxis](#). Chart creates one value axis automatically, so if you need only one value axis, you don't need to create it. Run `api("ValueAxis")` for more informations.

**valueAxis** [ValueAxis](#). Only valid for Gantt Charts. Set it's type to "date" if your data is date or time based.

**listeners** list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. Run `runShinyExamples()` for examples.

**otherProperties** list containing other available properties not yet implemented in the package.

**value** numeric.

**Author(s)**

datastorm-open

---

Title-class	<i>Title class</i>
-------------	--------------------

---

**Description**

Creates a title on above the chart, multiple can be assigned.

**Details**

Run `api("Title")` for more informations and all available properties.

**Slots**

`text` character, title's text.

`size` numeric, title's size.

`listeners` list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. See examples for details.

`otherProperties` list containing other available properties not yet implemented in the package.  
`value` numeric.

**Author(s)**

datastorm-open

---

TrendLine-class	<i>TrendLine class</i>
-----------------	------------------------

---

**Description**

Creates a trendline for `amSerialChart` and `amXYChart` charts which indicates the trend of your data or covers some different purposes. Multiple can be assigned.

**Details**

Run `api("TrendLine")` for more information and all available properties.

**Slots**

`finalValue` numeric. Value at which trend line should end.

`finalXValue` numeric. Used by XY chart only. X value at which trend line should end.

`initialValue` numeric. Value from which trend line should start.

`initialXValue` numeric. Used by XY chart only. X value from which trend line should start.

`valueAxis` [ValueAxis](#). Value axis of the trend line. Will use first value axis of the chart if not set any. You can use a reference to the value axis object or id of value axis.

valueAxisX [ValueAxis](#). Used by XY chart only. X axis of trend line. Will use first X axis of the chart if not set any. You can use a reference to the value axis object or id of value axis.

listeners list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. See examples for details.

otherProperties list, containing other available properties.

value numeric.

#### Author(s)

datastorm-open

---

ValueAxis-class

*ValueAxis class*

---

#### Description

Extension for ValueAxis to create an axis for amSerialChart, amRadarChart, amXYChart charts, multiple can be assigned. Gets automatically populated, one for amSerialChart and two for amXY-Chart charts, if none has been specified.

#### Details

Run `api("ValueAxis")` for more information and all available properties.

#### Slots

title character. Title of the axis.

guides list.

listeners list containing the listeners to add to the object. The list must be named as in the official API. Each element must be a character string. See examples for details.

otherProperties list containing other available properties not yet implemented in the package.

value numeric. Guides belonging to this axis. Use `addGuide` method

#### Author(s)

datastorm-open

# Index

## \* datasets

- data\_AirPassengers, [63](#)
- data\_bar, [63](#)
- data\_candleStick1, [64](#)
- data\_candleStick2, [64](#)
- data\_fbar, [65](#)
- data\_funnel, [65](#)
- data\_gantt, [66](#)
- data\_gbar, [66](#)
- data\_gdp, [67](#)
- data\_mekko, [67](#)
- data\_pie, [68](#)
- data\_radar, [68](#)
- data\_stock1, [69](#)
- data\_stock\_2, [69](#)
- data\_stock\_3, [70](#)
- data\_waterfall, [70](#)
- data\_wind, [71](#)
  
- add\_animate\_dependency, [7](#)
- add\_dataloader\_dependency, [8](#)
- add\_export\_dependency, [8](#)
- add\_responsive\_dependency, [9](#)
- add\_theme\_dependency, [9](#)
- addArrow (initialize, AmChart-method), [79](#)
- addArrow, AmChart, GaugeArrowOrMissing-method (initialize, AmChart-method), [79](#)
- addAxe (initialize, AmChart-method), [79](#)
- addAxe, AmChart, GaugeAxisOrMissing-method (initialize, AmChart-method), [79](#)
- addAxis (initialize, AmChart-method), [79](#)
- addAxis, AmChart, GaugeAxisOrMissing-method (initialize, AmChart-method), [79](#)
- addBand (initialize, GaugeAxis-method), [108](#)
- addBand, GaugeAxis, GaugeBandOrMissing-method (initialize, GaugeAxis-method), [108](#)
- addComparedDataSet (initialize, AmStockChart-method), [95](#)
- addComparedDataSet, AmStockChart, DataSetOrMissing-method (initialize, AmStockChart-method), [95](#)
- addDataSet (initialize, AmStockChart-method), [95](#)
- addDataSet, AmStockChart, DataSetOrMissing-method (initialize, AmStockChart-method), [95](#)
- addFieldMapping (initialize, DataSet-method), [104](#)
- addFieldMapping, DataSet-method (initialize, DataSet-method), [104](#)
- addGraph (initialize, AmChart-method), [79](#)
- addGraph, AmChart, AmGraphOrMissing-method (initialize, AmChart-method), [79](#)
- addGuide (Generics functions), [73](#)
- addGuide, AmChart, GuideOrMissing-method (initialize, AmChart-method), [79](#)
- addGuide, AxisBase, GuideOrMissing-method, [5](#)
- addLabel (initialize, AmChart-method), [79](#)
- addLabel, AmChart, LabelOrMissing-method (initialize, AmChart-method), [79](#)
- addListener, [5](#)
- addListener, AmObject, character, character-method (addListener), [5](#)
- addPanel (initialize, AmStockChart-method), [95](#)
- addPanel, AmStockChart, StockPanelOrMissing-method (initialize, AmStockChart-method), [95](#)
- addPeriod (initialize, PeriodSelector-method), [113](#)

- addPeriod,PeriodSelector-method  
(initialize,PeriodSelector-method),  
113
- addSegment(initialize,AmChart-method),  
79
- addSegment,AmChart,numeric-method  
(initialize,AmChart-method), 79
- addStockEvent  
(initialize,DataSet-method),  
104
- addStockEvent,DataSet,StockEventOrMissing-method  
(initialize,DataSet-method),  
104
- addStockGraph  
(initialize,StockPanel-method),  
115
- addStockGraph,StockPanel,AmGraphOrMissing-method  
(initialize,StockPanel-method),  
115
- addSubData(initialize,AmChart-method),  
79
- addSubData,AmChart,numeric-method  
(initialize,AmChart-method), 79
- addTitle(initialize,AmChart-method), 79
- addTitle,AmChart,TitleOrMissing-method  
(initialize,AmChart-method), 79
- addTrendLine  
(initialize,AmChart-method), 79
- addTrendLine,AmChart,TrendLineOrMissing-method  
(initialize,AmChart-method), 79
- addValueAxes  
(initialize,AmChart-method), 79
- addValueAxes,AmChart,ValueAxisOrMissing-method  
(initialize,AmChart-method), 79
- addValueAxis  
(initialize,AmChart-method), 79
- addValueAxis,AmChart,ValueAxisOrMissing-method  
(initialize,AmChart-method), 79
- amAngularGauge, 10, 11, 14, 17, 19, 21, 26,  
28, 31, 34, 37, 39, 40, 44, 46, 47, 50,  
54, 57, 58
- amAngularGaugeChart  
(initialize,AmChart-method), 79
- AmBalloon, 22, 48, 73, 77, 78, 85, 86, 98, 116,  
137
- amBalloon  
(initialize,AmBalloon-method),  
77
- AmBalloon-class, 11
- amBarplot, 11, 12, 14, 17, 19, 21, 26, 28, 31,  
34, 37, 39, 40, 44, 46, 47, 50, 54, 57,  
58
- amBoxplot, 11, 14, 15, 17, 19, 21, 26, 28, 31,  
34, 37, 39, 40, 44, 46, 47, 50, 54, 57,  
58
- amBullet, 11, 14, 17, 18, 19, 21, 26, 28, 31,  
34, 37, 39, 40, 44, 46, 47, 50, 54, 57,  
58
- amCandlestick, 11, 14, 17, 19, 20, 21, 26, 28,  
31, 34, 37, 39, 40, 44, 46, 47, 50, 54,  
57, 58
- AmChart, 13, 17, 26, 31, 33, 38, 84, 86, 87,  
124, 132, 133
- amChart(initialize,AmChart-method), 79
- AmChart-class, 22
- amChartsAPI, 11, 14, 17, 19, 21, 24, 26, 28,  
31, 34, 36, 39, 40, 44, 45, 47, 50, 54,  
57, 58
- amChartsOutput, 24, 125
- amFloatingBar, 11, 14, 17, 19, 21, 25, 26, 28,  
31, 34, 37, 39, 40, 44, 46, 47, 50, 54,  
57, 58
- amFunnel, 11, 14, 17, 19, 21, 26, 27, 28, 31,  
34, 37, 39, 40, 44, 46, 47, 50, 54, 57,  
58
- amFunnelChart  
(initialize,AmChart-method), 79
- amGanttChart  
(initialize,AmChart-method), 79
- AmGraph, 23, 73, 85, 86, 93, 94, 104, 114, 116,  
117, 134, 135, 137
- amGraph(initialize,AmGraph-method), 92
- AmGraph-class, 29
- amHist, 11, 14, 17, 19, 21, 26, 28, 30, 31, 34,  
37, 39, 40, 44, 46, 47, 50, 54, 57, 58
- AmLegend, 23, 85, 86, 95, 117, 136, 137
- amLegend(initialize,AmLegend-method),  
94
- AmLegend-class, 32
- amLines, 32
- amMekko, 11, 14, 17, 19, 21, 26, 28, 31, 34, 34,  
37, 39, 40, 44, 46, 47, 50, 54, 57, 58
- AmObject, 6, 12, 73, 133
- AmObject-class, 35
- amOHLC, 11, 14, 17, 19, 21, 26, 28, 31, 34, 35,  
37, 39, 40, 44, 46, 47, 50, 54, 57, 58

- amOptions, [11](#), [13](#), [14](#), [16](#), [17](#), [19](#), [21](#), [26](#), [28](#),  
[30](#), [31](#), [34](#), [36](#), [37](#), [37](#), [39](#), [40](#), [43–47](#),  
[50](#), [54](#), [57](#), [58](#)
- amPie, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#), [34](#), [37](#),  
[39](#), [40](#), [40](#), [44](#), [46](#), [47](#), [50](#), [54](#), [57](#), [58](#)
- amPieChart (initialize, AmChart-method),  
[79](#)
- amPlot, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#), [34](#), [37](#),  
[39](#), [40](#), [41](#), [44](#), [46](#), [47](#), [50](#), [54](#), [57](#), [58](#)
- amRadar, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#), [34](#), [37](#),  
[39](#), [40](#), [44](#), [45](#), [46](#), [47](#), [50](#), [54](#), [57](#), [58](#)
- amRadarChart  
(initialize, AmChart-method), [79](#)
- amSerialChart  
(initialize, AmChart-method), [79](#)
- amSolidGauge, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#),  
[34](#), [37](#), [39](#), [40](#), [44](#), [46](#), [46](#), [47](#), [50](#), [54](#),  
[57](#), [58](#)
- AmStockChart, [62](#), [98](#), [132](#), [134](#), [135](#)
- amStockChart  
(initialize, AmStockChart-method),  
[95](#)
- AmStockChart-class, [48](#)
- amStockMultiSet, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#),  
[31](#), [34](#), [37](#), [39](#), [40](#), [44](#), [46](#), [47](#), [49](#), [50](#),  
[54](#), [57](#), [58](#)
- amTimeSeries, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#),  
[34](#), [37](#), [39](#), [40](#), [44](#), [46](#), [47](#), [50](#), [51](#), [54](#),  
[57](#), [58](#), [130](#)
- amTitle (initialize, Title-method), [118](#)
- amWaterfall, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#),  
[34](#), [37](#), [39](#), [40](#), [44](#), [46](#), [47](#), [50](#), [54](#), [56](#),  
[57](#), [58](#)
- amWind, [11](#), [14](#), [17](#), [19](#), [21](#), [26](#), [28](#), [31](#), [34](#), [37](#),  
[39](#), [40](#), [44](#), [46](#), [47](#), [50](#), [54](#), [57](#), [57](#), [58](#)
- amXYChart (initialize, AmChart-method),  
[79](#)
- api, [58](#)
- AxisBase, [5](#)
- AxisBase-class, [59](#)
- CategoryAxis, [22](#), [85](#), [101](#), [116](#), [137](#)
- categoryAxis  
(initialize, CategoryAxis-method),  
[101](#)
- CategoryAxis-class, [59](#)
- ChartCursor, [22](#), [85](#), [102](#), [103](#), [116](#), [137](#)
- chartCursor  
(initialize, ChartCursor-method),  
[102](#)
- ChartCursor-class, [60](#)
- ChartScrollbar, [22](#), [23](#), [85](#), [86](#), [98](#), [104](#), [116](#),  
[117](#), [137](#)
- chartScrollbar  
(initialize, ChartScrollbar-method),  
[103](#)
- ChartScrollbar-class, [61](#)
- chartScrollbarSettings  
(initialize, ChartScrollbar-method),  
[103](#)
- controlShinyPlot, [61](#)
- data\_AirPassengers, [63](#)
- data\_bar, [13](#), [63](#)
- data\_candleStick1, [20](#), [64](#)
- data\_candleStick2, [20](#), [64](#)
- data\_fbar, [25](#), [65](#)
- data\_funnel, [28](#), [65](#)
- data\_gantt, [66](#)
- data\_gbar, [13](#), [66](#)
- data\_gdp, [67](#)
- data\_mekko, [34](#), [67](#)
- data\_pie, [40](#), [68](#)
- data\_radar, [45](#), [68](#)
- data\_stock1, [69](#)
- data\_stock\_2, [69](#)
- data\_stock\_3, [70](#)
- data\_waterfall, [57](#), [70](#)
- data\_wind, [58](#), [71](#)
- DataSet, [48](#), [98](#), [105](#), [106](#)
- dataSet (initialize, DataSet-method), [104](#)
- DataSet-class, [62](#)
- DataSetSelector, [48](#), [98](#), [107](#)
- dataSetSelector  
(initialize, DataSetSelector-method),  
[106](#)
- DataSetSelector-class, [62](#)
- GaugeArrow, [22](#), [85](#), [86](#), [107](#), [108](#), [136](#)
- gaugeArrow  
(initialize, GaugeArrow-method),  
[107](#)
- GaugeArrow-class, [71](#)
- GaugeAxis, [22](#), [71](#), [85](#), [86](#), [107](#), [109](#), [116](#), [136](#)
- gaugeAxis  
(initialize, GaugeAxis-method),  
[108](#)
- GaugeAxis-class, [72](#)

- GaugeBand, [72](#), [109](#), [110](#)
- gaugeBand
  - (initialize, GaugeBand-method), [109](#)
- GaugeBand-class, [72](#)
- Generics functions, [73](#)
- getCurrentStockData, [74](#)
- getTransformTS, [75](#)
- graph (initialize, AmGraph-method), [92](#)
- Guide, [5](#), [23](#), [73](#), [85](#), [86](#), [101](#), [111](#), [117](#), [122](#), [137](#)
- guide (initialize, Guide-method), [110](#)
- Guide-class, [77](#)
- initialize, AmBalloon-method, [77](#)
- initialize, AmChart-method, [79](#)
- initialize, AmGraph-method, [92](#)
- initialize, AmLegend-method, [94](#)
- initialize, AmStockChart-method, [95](#)
- initialize, CategoryAxis-method, [101](#)
- initialize, ChartCursor-method, [102](#)
- initialize, ChartScrollbar-method, [103](#)
- initialize, DataSet-method, [104](#)
- initialize, DataSetSelector-method, [106](#)
- initialize, GaugeArrow-method, [107](#)
- initialize, GaugeAxis-method, [108](#)
- initialize, GaugeBand-method, [109](#)
- initialize, Guide-method, [110](#)
- initialize, Label-method, [111](#)
- initialize, PeriodSelector-method, [113](#)
- initialize, StockEvent-method, [114](#)
- initialize, StockPanel-method, [115](#)
- initialize, Title-method, [118](#)
- initialize, TrendLine-method, [119](#)
- initialize, ValueAxis-method, [121](#)
- Label, [22](#), [84](#), [86](#), [112](#), [116](#), [136](#)
- label (initialize, Label-method), [111](#)
- Label-class, [122](#)
- legend (initialize, AmLegend-method), [94](#)
- listProperties, [123](#)
- listProperties, AmObject-method
  - (listProperties), [123](#)
- NS, [125](#), [128](#)
- panel (initialize, StockPanel-method), [115](#)
- PeriodSelector, [48](#), [98](#), [113](#)
- periodSelector
  - (initialize, PeriodSelector-method), [113](#)
- PeriodSelector-class, [123](#)
- plot, AmCharts-method, [124](#)
- print, AmObject-method, [124](#)
- rAmCharts-shinymodules, [125](#)
- rAmCharts-shinymodules-ts, [126](#)
- rAmChartsExportServer
  - (rAmCharts-shinymodules), [125](#)
- rAmChartsExportServerUI
  - (rAmCharts-shinymodules), [125](#)
- rAmChartsTimeSeriesServer
  - (rAmCharts-shinymodules-ts), [126](#)
- rAmChartsTimeSeriesUI, [49](#), [53](#), [54](#), [98](#)
- rAmChartsTimeSeriesUI
  - (rAmCharts-shinymodules-ts), [126](#)
- renderAmCharts, [131](#)
- resetProperties (addListener), [5](#)
- resetProperties, AmObject-method
  - (addListener), [5](#)
- runExamples, [131](#)
- setAdjustBorderColor
  - (initialize, AmBalloon-method), [77](#)
- setAdjustBorderColor, AmBalloon, logical-method
  - (initialize, AmBalloon-method), [77](#)
- setAllLabels
  - (initialize, AmChart-method), [79](#)
- setAllLabels, AmChart, list-method
  - (initialize, AmChart-method), [79](#)
- setArrows (initialize, AmChart-method), [79](#)
- setArrows, AmChart-method
  - (initialize, AmChart-method), [79](#)
- setAxes (initialize, AmChart-method), [79](#)
- setAxes, AmChart, list-method
  - (initialize, AmChart-method), [79](#)
- setAxis (initialize, GaugeArrow-method), [107](#)
- setAxis, GaugeArrow, GaugeAxisOrCharacterOrMissing-method
  - (initialize, GaugeArrow-method), [107](#)
- setBalloon (Generics functions), [73](#)

- setBalloon, AmChart, AmBalloonOrMissing-method (initialize, AmChart-method), 79
- setBalloon, AmStockChart, AmBalloonOrMissing-method (initialize, AmStockChart-method), 95
- setBalloonText (initialize, AmGraph-method), 92
- setBalloonText, AmGraph, character-method (initialize, AmGraph-method), 92
- setBands (initialize, GaugeAxis-method), 108
- setBands, GaugeAxis, list-method (initialize, GaugeAxis-method), 108
- setBold (initialize, Label-method), 111
- setBold, Label, logical-method (initialize, Label-method), 111
- setCategoryAxesSettings (initialize, AmStockChart-method), 95
- setCategoryAxesSettings, AmStockChart-method (initialize, AmStockChart-method), 95
- setCategoryAxis (initialize, AmChart-method), 79
- setCategoryAxis, AmChart-method (initialize, AmChart-method), 79
- setCategoryField (initialize, AmChart-method), 79
- setCategoryField, AmChart, character-method (initialize, AmChart-method), 79
- setChartCursor (initialize, AmChart-method), 79
- setChartCursor, AmChart, ChartCursorOrMissing-method (initialize, AmChart-method), 79
- setChartCursorSettings (initialize, AmStockChart-method), 95
- setChartCursorSettings, AmStockChart-method (initialize, AmStockChart-method), 95
- setChartScrollbar (initialize, AmChart-method), 79
- setChartScrollbar, AmChart, ChartScrollbarOrMissing-method (initialize, AmChart-method), 79
- setChartScrollbarSettings (initialize, AmStockChart-method), 95
- setChartScrollbarSettings, AmStockChart-method (initialize, AmStockChart-method), 95
- setChartScrollbarSettings, AmStockChart, ChartScrollbarOrMissing-method (initialize, AmStockChart-method), 95
- setColor (initialize, AmBalloon-method), 77
- setColor, AmBalloon, character-method (initialize, AmBalloon-method), 77
- setComparedDataSets (initialize, AmStockChart-method), 95
- setComparedDataSets, AmStockChart-method (initialize, AmStockChart-method), 95
- setCornerRadius (initialize, AmBalloon-method), 77
- setCornerRadius, AmBalloon, numeric-method (initialize, AmBalloon-method), 77
- setCreditsPosition (initialize, AmChart-method), 79
- setCreditsPosition, AmChart, character-method (initialize, AmChart-method), 79
- setDataLoader (initialize, AmChart-method), 79
- setDataLoader, AmChart, character, character-method (initialize, AmChart-method), 79
- setDataProvider (Generics functions), 73
- setDataProvider, AmChart, ANY, logicalOrMissing-method (initialize, AmChart-method), 79
- setDataProvider, DataSet, ANY, ANY-method (initialize, DataSet-method), 79
- setDataSets (initialize, AmStockChart-method), 95
- setDataSets, AmStockChart-method (initialize, AmStockChart-method), 95
- setDataSetSelector (initialize, AmStockChart-method), 95
- setDataSetSelector, AmStockChart-method (initialize, AmStockChart-method), 95
- setDrawOnAxis (initialize, StockPanel-method), 79



- 115
- setDrawOnAxis, StockPanel, ValueAxisOrCharacterOrMissing-method, list-method (initialize, StockPanel-method), 115
- setEnabled (initialize, ChartScrollbar-method), 103
- setEnabled, ChartScrollbar, logical-method (initialize, ChartScrollbar-method), 103
- setExport, 132
- setExport, AmCharts, logicalOrMissing-method (setExport), 132
- setFieldMappings (initialize, DataSet-method), 104
- setFieldMappings, DataSet, list-method (initialize, DataSet-method), 104
- setFillAlpha (initialize, Guide-method), 110
- setFillAlpha, Guide, numeric-method (initialize, Guide-method), 110
- setFillColor (initialize, AmBalloon-method), 77
- setFillColor, AmBalloon, character-method (initialize, AmBalloon-method), 77
- setFinalValue (initialize, TrendLine-method), 119
- setFinalValue, TrendLine, numeric-method (initialize, TrendLine-method), 119
- setFinalXValue (initialize, TrendLine-method), 119
- setFinalXValue, TrendLine, numeric-method (initialize, TrendLine-method), 119
- setGraph (Generics functions), 73
- setGraph, AmChart, AmGraphOrMissing-method (initialize, AmChart-method), 79
- setGraph, ChartScrollbar, AmGraphOrCharacterOrMissing-method (initialize, ChartScrollbar-method), 103
- setGraphs (initialize, AmChart-method), 79
- setGridPosition (initialize, AmChart-method), list-method (initialize, AmChart-method), 79
- setGridPosition (initialize, CategoryAxis-method), 101
- setGridPosition, CategoryAxis, character-method (initialize, CategoryAxis-method), 101
- setGuides (initialize, AmChart-method), 79
- setGuides, AmChart, list-method (initialize, AmChart-method), 79
- setID (initialize, GaugeBand-method), 109
- setID, GaugeBand-method (initialize, GaugeBand-method), 109
- setInitialValue (initialize, TrendLine-method), 119
- setInitialValue, TrendLine, numeric-method (initialize, TrendLine-method), 119
- setInitialXValue (initialize, TrendLine-method), 119
- setInitialXValue, TrendLine, numeric-method (initialize, TrendLine-method), 119
- setLegend (initialize, AmChart-method), 79
- setLegend, AmChart, AmLegendOrMissing-method (initialize, AmChart-method), 79
- setLegendSettings (initialize, AmStockChart-method), 95
- setLegendSettings, AmStockChart-method (initialize, AmStockChart-method), 95
- setMainDataSet (initialize, AmStockChart-method), 95
- setMainDataSet, AmStockChart, DataSetOrMissing-method (initialize, AmStockChart-method), 95
- setOneBalloonOnly (initialize, ChartCursor-method), 102

- setOneBalloonOnly, ChartCursor, logical-method (initialize, ChartCursor-method), 102
- setPanels (initialize, AmStockChart-method), 95
- setPanels, AmStockChart, list-method (initialize, AmStockChart-method), 95
- setPanelsSettings (initialize, AmStockChart-method), 95
- setPanelsSettings, AmStockChart-method (initialize, AmStockChart-method), 95
- setPeriodSelector (initialize, AmStockChart-method), 95
- setPeriodSelector, AmStockChart, PeriodSelectorOrMissing-method (initialize, AmStockChart-method), 95
- setPosition (initialize, DataSetSelector-method), 106
- setPosition, DataSetSelector, character-method (initialize, DataSetSelector-method), 106
- setProperties, 37
- setProperties (addListener), 5
- setProperties, AmObject-method (addListener), 5
- setResponsive (setExport), 132
- setResponsive, AmCharts, logicalOrMissing-method (setExport), 132
- setSize (initialize, Title-method), 118
- setSize, Title, numeric-method (initialize, Title-method), 118
- setStockEvents (initialize, DataSet-method), 104
- setStockEvents, DataSet, list-method (initialize, DataSet-method), 104
- setStockEventsSettings (initialize, AmStockChart-method), 95
- setStockEventsSettings, AmStockChart-method (initialize, AmStockChart-method), 95
- setStockGraph (initialize, StockEvent-method), 114
- setStockGraph, StockEvent, AmGraphOrCharacterOrMissing-method (initialize, StockEvent-method), 114
- setStockGraphs (initialize, StockPanel-method), 115
- setStockGraphs, StockPanel, list-method (initialize, StockPanel-method), 115
- setStockLegend (initialize, StockPanel-method), 115
- setStockLegend, StockPanel, AmLegendOrMissing-method (initialize, StockPanel-method), 115
- setSubChartProperties (initialize, AmChart-method), 79
- setSubChartProperties, AmChart-method (initialize, AmChart-method), 79
- setText (Generics functions), 73
- setText, Label, character-method (initialize, Label-method), 111
- setText, Title, character-method (initialize, Title-method), 118
- setTheme (initialize, AmChart-method), 79
- setTheme, AmChart, character-method (initialize, AmChart-method), 79
- setTitle (Generics functions), 73
- setTitle, AmGraph, character-method (initialize, AmGraph-method), 92
- setTitle, ValueAxis, character-method (initialize, ValueAxis-method), 121
- setTitles (initialize, AmChart-method), 79
- setTitles, AmChart, list-method (initialize, AmChart-method), 79
- setTrendLines (initialize, AmChart-method), 79
- setTrendLines, AmChart, list-method (initialize, AmChart-method), 79
- setType (Generics functions), 73
- setType, AmChart, character-method (initialize, AmChart-method), 79

- setType, AmGraph, character-method  
(initialize, AmGraph-method), 92
- setUseGraphSettings  
(initialize, AmLegend-method),  
94
- setUseGraphSettings, AmLegend, logical-method  
(initialize, AmLegend-method),  
94
- setValueAxes  
(initialize, AmChart-method), 79
- setValueAxes, AmChart, list-method  
(initialize, AmChart-method), 79
- setValueAxesSettings  
(initialize, AmStockChart-method),  
95
- setValueAxesSettings, AmStockChart-method  
(initialize, AmStockChart-method),  
95
- setValueAxis (Generics functions), 73
- setValueAxis, AmChart, ValueAxisOrMissing-method  
(initialize, AmChart-method), 79
- setValueAxis, Guide, ValueAxisOrCharacterOrMissing-method  
(initialize, Guide-method), 110
- setValueAxis, TrendLine, ValueAxisOrCharacterOrMissing-method  
(initialize, TrendLine-method),  
119
- setValueAxisX  
(initialize, TrendLine-method),  
119
- setValueAxisX, TrendLine, ValueAxisOrCharacterOrMissing-method  
(initialize, TrendLine-method),  
119
- setValueField  
(initialize, AmGraph-method), 92
- setValueField, AmGraph, character-method  
(initialize, AmGraph-method), 92
- setValueLineAxis  
(initialize, ChartCursor-method),  
102
- setValueLineAxis, ChartCursor, ValueAxisOrCharacterOrMissing-method  
(initialize, ChartCursor-method),  
102
- setValueScrollbar  
(initialize, AmChart-method), 79
- setValueScrollbar, AmChart, ChartScrollbarOrMissing-method  
(initialize, AmChart-method), 79
- setX (initialize, Label-method), 111
- setX, Label, numericOrCharacter-method  
(initialize, Label-method), 111
- setY (initialize, Label-method), 111
- setY, Label, numericOrCharacter-method  
(initialize, Label-method), 111
- show, AmChart-method, 133
- show, AmObject-method, 133
- show, AmStockChart-method, 134
- StockEvent, 62, 106, 114
- stockEvent  
(initialize, StockEvent-method),  
114
- StockEvent-class, 134
- stockGraph, 135
- stockLegend, 135
- StockPanel, 48, 98, 116, 117
- stockPanel  
(initialize, StockPanel-method),  
115
- StockPanel-class, 136
- Title, 23, 85, 86, 117–119, 137
- title (initialize, Title-method), 118
- Title-class, 138
- TrendLine, 23, 85, 86, 117, 120, 137
- trendLine  
(initialize, TrendLine-method),  
119
- TrendLine-class, 138
- ValueAxis, 23, 73, 77, 86, 102, 111, 117, 120,  
122, 136–139
- valueAxis  
(initialize, ValueAxis-method),  
121
- ValueAxis-class, 139