Package 'avfintools'

October 12, 2022

Type Package

Title Financial Analysis Tools Using Data from 'Alpha Vantager'

Version 0.1.0

Maintainer Edward Wei <edwwei2020@gmail.com>

Description To pull data from 'ALPHA VANTAGE' <https://www.alphavantage.co/>, use the av_api_key() function from 'alphavantager' for inserting your API key. This is a complement to the 'alphavantager' package from CRAN. Contains commonly used quantitative finance tools. 'avfintools' stands for 'ALPHA VANTAGE' Finance Tools, as it depends on sourcing financial data from the 'ALPHA VANTAGE' <https://www.alphavantage.co/documentation/> API.

Language en-US

License CC0

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Depends R(>= 4.0)

Imports methods(>= 4.2.0), alphavantager(>= 0.1.2), dplyr(>= 1.0.9), lubridate(>= 1.8.0), plotly(>= 4.10.0), ggplot2(>= 3.3.6), tibble(>= 3.1.7)

NeedsCompilation no

Author Edward Wei [aut, cre, cph]

Repository CRAN

Date/Publication 2022-10-06 16:10:08 UTC

R topics documented:

`addreturns` *Adds various returns as well as additional statistics.*

Description

Adds percentage returns, interday returns, total returns, cumulative returns, multiplicative returns, range, and dollar returns to the default dataframe pulled from alphavantager

Usage

```
addreturns(df)
```

Arguments

`df`	Dataframe pulled from alphavantager

Value

A more detailed dataframe with additional return metrics and summary statistics.

Examples

```
addreturns(SPYdaily)
```

ATR *Average True Range*

Description

Returns the average true range as well as the relative price based on the ATR as a reference

Usage

```
ATR(df, period, current = FALSE, mrprice = NULL, hideprints = TRUE)
```

Arguments

df	Dataframe with price data.
period	Calculation period in day for the true range
current	If one wants to input the latest price point before data update
mrprice	Most recent price;
hideprints	if TRUE hides the print outs regarding the percentile within an ATR

Value

Returns a vector of ATR calculations in dataframe format. If current = TRUE, returns the most recent ATR as well as where price is in the context of the ATR

Examples

```
ATR(SPY15, 14)
ATR(SPY15, 14, current = TRUE, mrprice = tail(SPYdaily$close, 1) + 2)
```

---

candles *Candlestick chart*

Description

Returns plot_ly candlestick chart

Usage

```
candles(df)
```

Arguments

df	Dataframe with price data.

Value

A candlestick chart

Examples

```
candles(SPYdaily)
```

---

compare_returns *Compare returns visually between two securities*

Description

Prints a plotly graph comparing returns, cumulative nominal returns, and cumulative multiplicative returns

Usage

```
compare_returns(a, b, a_name, b_name)
```

Arguments

a	The first dataframe filled with data from the function getdaily()
b	The second dataframe filled with data from the function getdaily()
a_name	Character string name for first dataframe
b_name	Character string name for second dataframe

Value

Returns plotly graph comparing returns, cumulative nominal returns, and cumulative multiplicative returns. Click graph name to un-toggle for better visibility.

Examples

```
compare_returns(GMEdaily, SPYdaily, "GME", "SPY")
```

crypto60 *Get Cryptocurrency Data at the hourly level localized to current time zone*

Description

Summary statistics for the movements on the hourly level

Usage

```
crypto60(coin_name)
```

Arguments

coin_name	The ticker symbol for the concurrency as a string

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment. Adjusted to local time zone.

Examples

```
## Not run:
crypto60("BTC")

## End(Not run)
```

cryptodaily *Get Cryptocurrency Data at the Daily Level*

Description

Daily, as in the summary statistics for the daily movement

Usage

```
cryptodaily(coin_name)
```

Arguments

coin_name	The ticker symbol for the concurrency as a string

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment.

Examples

```
## Not run:
cryptodaily("WTI")

## End(Not run)
```

`fastzoom` *Fast Zoom*

Description

An Easy way to get to where you want to on a candlestick chart or other plots

Usage

```
fastzoom(plot, x)
```

Arguments

`plot`	The plot to be zoomed into
`x`	A character string in the format("YYYYMMDD") for where to zoom in

Value

The graph but zoomed in where you want to, mostly for daily data

Examples

```
fastzoom(candles(SPYdaily), "20220202")
```

`fibs` *Show Fibonacci bands*

Description

Displays Fibonacci bands

Usage

```
fibs(df, showgraph = TRUE, title = NULL, hideprints = FALSE)
```

Arguments

`df`	Dataframe with price data, works with various intervals
`showgraph`	Whether or not you want the function to pop out the visual
`title`	A character string for the Title of your graph
`hideprints`	if set to FALSE, prints out the support and resistance levels

Value

Returns graph with various levels as well as a vector

Examples

```
fibs(tail(SPYdaily, 200))
SPYdailyfibs <- fibs(tail(SPYdaily, 200))
fibs(SPY15)
```

`findcor` *Calculate the correlation between a column shared within two dataframes.*

Description

Two dataframes from one of the "get" functions recommended, but works with any dataframe that shares columns as well as column lengths.

Usage

```
findcor(a, b, sdata)
```

Arguments

`a`	First dataframe
`b`	Second dataframe
`sdata`	A string of the column name to compare

Value

The correlation as a single numeric.

Examples

```
findcor(SPYdaily, GMEdaily, "returns")
```

genMA *Generate moving averages*

Description

Description

Usage

```
genMA(df, ma)
```

Arguments

df	Dataframe with price data
ma	# of periods for the moving average to calculate

Value

a vector with the same number of columns as df showing the moving averages. Periods before moving average should be not considered for use. Output is kept same columns for compatibility.

Examples

```
SPYDMA200 <- genMA(SPYdaily, 14)
```

get15 *Get Stock Data at the 15 minute level localized to current time zone*

Description

Summary statistics for the movements on the 15 minute level

Usage

```
get15(ticker, truncated = TRUE)
```

Arguments

ticker	The ticker symbol as a string
truncated	Option to limit output to hours closer to market open hours. Default is true.

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment. Default is truncated to show data more relevant to active trading hours. Adjusted to local time zone.

Examples

```
## Not run:
get15("WTI")
get15("SPY", truncated = FALSE)

## End(Not run)
```

---

get5 *Get Stock Data at the 5 minute level localized to current time zone*

Description

Summary statistics for the movements on the 5 minute level

Usage

```
get5(ticker, truncated = TRUE)
```

Arguments

ticker	The ticker symbol as a string
truncated	Option to limit output to hours closer to market open hours. Default is true.

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment. Default is truncated to show data more relevant to active trading hours. Adjusted to local time zone.

Examples

```
## Not run:
get5("WTI")
get5("SPY", truncated = FALSE)

## End(Not run)
```

get60 *Get Stock Data at the hourly level localized to current time zone*

Description

Summary statistics for the movements on the hourly level

Usage

```
get60(ticker, truncated = TRUE)
```

Arguments

ticker	The ticker symbol as a string
truncated	Option to limit output to hours closer to market open hours. Default is true.

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment. Default is truncated to show data more relevant to active trading hours. Adjusted to local time zone.

Examples

```
## Not run:
get60("WTI")
get60("SPY", truncated = FALSE)

## End(Not run)
```

getdaily *Get Stock Data at the Daily Level*

Description

Daily, as in the summary statistics for the daily movement

Usage

```
getdaily(ticker)
```

Arguments

ticker	The ticker symbol as a string

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment.

Examples

```
## Not run:
getdaily("SPY")

## End(Not run)
```

`getweekly` *Get Stock Data at the Weekly level*

Description

Summary statistics for the stock movements on the weekly

Usage

```
getweekly(ticker)
```

Arguments

`ticker`	The ticker symbol as a string

Value

A data frame with daily data such as the high, low, open, close, and associated returns. Available in the global environment.

Examples

```
## Not run:
getweekly("WTI")

## End(Not run)
```

GMEdaily *This is data to be included in my package*

Description

This is data to be included in my package

Usage

```
GMEdaily
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 360 rows and 13 columns.

Source

Alpha Vantager API

References

https://www.alphavantage.co/documentation/

idret *After-hour and Pre-market Returns*

Description

Show returns between the close of the last trading day and the open of the current trading day

Usage

```
idret(df)
```

Arguments

df	Dataframe with daily data

Value

A vector in dataframe format of combined after-hour, overnight, and pre-market returns in percentage

Examples

```
idret(SPYdaily)
```

nvi *Negative Volume Index*

Description

Calculates the negative volume index, uses closing price of the time period

Usage

```
nvi(df)
```

Arguments

df	Dataframe with price data.

Value

Returns a 1 x # of columns in df dataframe

Examples

```
nvi(tail(SPYdaily, 200))
```

project_price *Projects future prices based on regression*

Description

Works with multiple time intervals. Do not oversupply with data, regression is expensive.

Usage

```
project_price(df, tickername)
```

Arguments

df	Dataframe with price data. The opening price is used for projection purposes, works for all security types
tickername	The ticker or the security you are putting in.

Value

What comes out of this function

Examples

```
project_price(tail(SPYdaily,200), "SPY")
project_price(SPY15, "SPY")
```

`pvi` *Positive Volume Index*

Description

Calculates the positive volume index, uses closing price of the time period

Usage

```
pvi(df)
```

Arguments

`df`	Dataframe with price data

Value

Returns a 1 x # of columns in df dataframe

Examples

```
pvi(tail(SPYdaily, 200))
```

`ret_to_cr` *Total return to cumulative return*

Description

The cumulative percentage is the addition of subsequent total daily returns.

Usage

```
ret_to_cr(list_of_returns)
```

Arguments

`list_of_returns`	Vector in dataframe showing returns

Value

A vector in dataframe format cumulative percentage returns

Examples

```
ret_to_cr(SPYdaily$returns)
```

RSI *Relative Strength Index*

Description

Returns the Relative Strength Index with adjustable periods

Usage

```
RSI(df, periods, current = FALSE, pricechange = NULL, hideprints = TRUE)
```

Arguments

df	Dataframe with price data
periods	Calculation Period
current	If one wants to input the latest price point before data updates, RSI uses the percentage return at the end of the market hours
pricechange	Input in percentage
hideprints	If TRUE, hides printouts from the current message

Value

Returns a vector of RSI calculations in dataframe format. If current = TRUE, returns the most recent RSI.

Examples

```
RSI(SPY15, 14)
RSI(tail(SPYdaily,200), 14, current = TRUE, pricechange = 1.3)
```

showidreturns *Frequency plot based on Absolute percentage movements*

Description

Shows the cumulative probabilities of each percentage movement

Usage

```
showidreturns(df, name_in_string)
```

Arguments

df	Dataframe from the "get"
name_in_string	Name of security associated with dataframe

Value

A plot_ly graph showing the frequency of absolute returns. "All Data" - no data omitted "Cumulative Probability" Cumulative frequency graph of returns "No Extremes" filter out skewed data

Examples

```
showidreturns(SPYdaily, "SPY")
```

SPY15 *This is data to be included in my package*

Description

This is data to be included in my package

Usage

```
SPY15
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 960 rows and 13 columns.

Source

Alpha Vantager API

References

https://www.alphavantage.co/documentation/

SPYdaily *This is data to be included in my package*

Description

This is data to be included in my package

Usage

```
SPYdaily
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 360 rows and 13 columns.

Source

Alpha Vantager API

References

https://www.alphavantage.co/documentation/

streak *Streak*

Description

Counts the number of days the open price has moved consecutively Negative and positive streaks are represented by their sign

Usage

```
streak(df)
```

Arguments

df	Dataframe with price data.

Value

Returns a 1 x # of columns in df dataframe

Examples

```
streak(tail(SPYdaily, 200))
```

streak_var *Streak (Multiple Variables)*

Description

Counts the number of days the open price has moved consecutively Negative and positive streaks are represented by their sign; only works with various types of returns or mutated vector created by diff()

Usage

```
streak_var(df, var)
```

Arguments

df	Dataframe with price data.
var	String of column name you wish to see streak in

Value

Returns a 1 x # of columns in df dataframe

Examples

```
streak_var(tail(SPYdaily,200), "tot_ret")
```

thedayafter *Frequency plot of Subsequent Returns After a Percentage Input*

Description

Returns a frequency plot drawn from historical data based on a percentage change that occurred

Usage

```
thedayafter(dataset, price_input, hideprints = TRUE)
```

Arguments

dataset	Dataframe with daily data
price_input	the price movement, in percent, of the most recent (or whatever you are interested in) trading day
hideprints	if set to FALSE, returns summary statistics

Value

percentage frequency plot of the following day based on historical data

Examples

```
thedayafter(SPYdaily, -1.35)
```

`trtomr` *Total return to multiplicative return*

Description

Multiplicative returns are always comparative to the earliest return

Usage

```
trtomr(df)
```

Arguments

`df`	Dataframe with daily data

Value

A vector in dataframe format cumulative multiplicative returns

Examples

```
trtomr(SPYdaily)
```

`volatility_freq` *Frequency plot of Range, as well as maximum Upward and Downward Movement*

Description

a plot_ly plot

Usage

```
volatility_freq(df, tick_name, cumulative = FALSE, hideprints = FALSE)
```

Arguments

df	Dataframe with daily data
tick_name	The ticker so the graph is correct
cumulative	Default is FALSE, turn to TRUE for a cumulative plot.
hideprints	if set to FALSE, shows summary statistics

Value

Frequency plot where you can find intraday volatility (range), maximum upside (Upward Movement), maximum downside (Downward Movement) on a cumulative percentile basis

Examples

```
volatility_freq (SPYdaily, "SPY")
volatility_freq (SPYdaily, "SPY", cumulative = TRUE)
volatility_freq (SPYdaily, "SPY", hideprints = TRUE)
```

---

volp *Relative percentage from the Maximum Value*

Description

Returns the percent of the maximum volume movement from data

Usage

```
volp(df)
```

Arguments

df	Dataframe with price data.

Value

Returns a 1 x # of columns in df dataframe in percentage

Examples

```
volp(tail(SPYdaily, 200))
```

volume_analysis *Graph of Volume Indicators*

Description

Follows Open, PVI, NVI, and

Usage

```
volume_analysis(df, name)
```

Arguments

df	Dataframe with price data.
name	A character string to add to the title "____ Volume Analysis"

Value

Returns plot_ly graph with PVI, NVI, Open and

Examples

```
volume_analysis(SPY15, "SPY")
```

Index