# Package: ti (via r-universe)

November 2, 2026

**Title** Time Intelligence and Customer Segmentation for Financial Analysis

**Description** Calculate time intelligence metrics for financial planning and analysis. 'ti' provides functions for period-over-period comparisons (year-over-year, month-over-month), period-to-date calculations (YTD, MTD, QTD), and customer segmentation (ABC analysis, cohorts). Supports standard and retail calendars (4-4-5, 4-5-4, 5-4-4) with both in-memory and database backends via 'dbplyr'.

**Version** 4.3.0

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** cli, DBI, dbplyr (>= 2.3.0), dplyr, duckdb, glue, janitor, lubridate, purrr, rlang, S7, tibble, tidyr

**Suggests** testthat (>= 3.0.0), quarto, devtools, gt, fansi, contoso (>= 2.1.0), withr

**Config/testthat/edition** 3

**Depends** R (>= 4.2.0)

**URL** https://codeberg.org/usrbinr/ti

**Config/roxygen2/version** 8.1.1

**NeedsCompilation** no

**Author** Alejandro Hagan [aut, cre]

**Maintainer** Alejandro Hagan <alejandro.hagan@outlook.com>

**Config/pak/sysreqs** libicu-dev

**Repository** https://usrbinr.r-universe.dev

**Date/Publication** 2026-11-02 17:22:02 UTC

**RemoteUrl** https://github.com/cran/ti

**RemoteRef** HEAD

**RemoteSha** 61c5dcdb1ab5857c867331de483b9b7de85ffe14

# Contents

abc . . . 2
atd . . . 3
augment_standard_calendar . . . 5
calculate . . . 6
cohort . . . 7
dod . . . 8
mom . . . 10
momtd . . . 12
mtd . . . 14
mtdopm . . . 15
pmtd . . . 17
pqtd . . . 19
pwtd . . . 21
pytd . . . 23
qoq . . . 25
qoqtd . . . 26
qtd . . . 28
qtdopq . . . 29
wow . . . 31
wowtd . . . 33
wtd . . . 35
wtdopw . . . 37
yoy . . . 39
yoytd . . . 41
ytd . . . 43
ytdopy . . . 45

**Index** **47**

---

abc *ABC classification function*

---

## Description

- For your group variable, `abc()` will categorize which groups make up what proportion of the totals according to the category_values that you have entered
- The function returns a segment object which prints out the execution steps and actions that it will take to categorize your data
- Use calculate to return the results

## Usage

```
abc(.data, category_values, .value)
```

### Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `category_values` | vector of break points between 0 and 1 |
| `.value` | optional: if left blank,`abc()` will use the number of rows per group to categorize, alternatively you can pass a column name to categorize |

### Details

- This function is helpful to understand which groups of make up what proportion of the cumulative contribution
- If you do not provide a `.value` then it will count the transactions per group, if you provide `.value` then it will `sum()` the `.value` per group
- The function creates a `segment` object, which pre-processes the data into its components

### Value

abc object

### Examples

```
contoso::sales |>
  dplyr::group_by(product_key) |>
  abc(c(.1,.5,.7,.96,1), .value = gross_margin)
```

---

## `atd` *All period-to-date*

---

### Description

This calculates the all-time cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

### Usage

```
atd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: dod(), mom(), momtd(), mtd(), mtdopm(), pmtd(), pqtd(), pwtd(), pytd(), qoq(), qoqtd(), qtd(), qtdopq(), wow(), wowtd(), wtd(), wtdopw(), yoy(), yoytd(), ytd(), ytdopy()

## Examples

```
atd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard")
```

---

# augment_standard_calendar

*Add Comprehensive Date-Based Attributes to a DBI lazy frame or tibble object*

---

## Description

This function takes a data frame and a date column and generates a wide set of derived date attributes. These include start/end dates for year, quarter, month, and week; day-of-week indicators; completed and remaining days in each time unit; and additional convenience variables such as weekend indicators.

It is designed for time-based feature engineering and is useful in reporting, forecasting, time-series modeling, and data enrichment workflows.

## Usage

```
augment_standard_calendar(.data, .date)
```

## Arguments

| | |
|---|---|
| .data | A data frame or tibble containing at least one date column. |
| .date | A column containing Date values, passed using tidy-eval (e.g., {{ date_col }}). |

## Details

The function creates the following groups of attributes:

### 1. Start and end dates

- year_start_date, year_end_date
- quarter_start_date, quarter_end_date
- month_start_date, month_end_date

- week_start_date, week_end_date

**2. Day-of-week fields**

- day_of_week – numeric day of the week (1–7)
- day_of_week_label – ordered factor label (e.g., Mon, Tue, ...)

**3. Duration fields**

- days_in_year – total days in the year interval
- days_in_quarter – total days in the quarter interval
- days_in_month – total days in the month

**4. Completed/remaining days**

- days_complete_in_week, days_remaining_in_week
- days_complete_in_month, days_remaining_in_month
- days_complete_in_quarter, days_remaining_in_quarter
- days_complete_in_year, days_remaining_in_year

**5. Miscellaneous**

- weekend_indicator – equals 1 if Saturday or Sunday; otherwise 0

All date-derived fields ending in _date are coerced to class Date.

## Value

A dbi or tibble containing the original data along with all generated date-based attributes.

---

`calculate` *Execute time-intelligence or segments class objects to return the underlying transformed table*

---

## Description

The `calculate()` function takes an object created by a time function (like `ytd()`, `mtd()`, or `qtd()`) or a segment function (like `cohort()` or `abc()`) and executes the underlying transformation logic. It translates the function blueprint into an actionable query, returning the final data table.

## Usage

```
calculate(x, ...)

## S7 method for class <ti::ti>
calculate(x, ...)

## S7 method for class <ti::segment_cohort>
calculate(x, ...)

## S7 method for class <ti::segment_abc>
calculate(x, ...)
```

### Arguments

| | |
|---|---|
| x | ti object |
| ... | Not used. |

### Details

The TI and segment functions in **ti**—such as `ytd()` or `cohort()` and others—are designed to be **lazy and database-friendly**. They do not perform the heavy data transformation immediately. Instead, they return a blueprint object (of class `ti`,`segment_abc` or `segment_cohort`) that contains all the parameters and logic needed for the calculation.

`calculate()` serves as the **execution engine**.

When called, it interprets the blueprint and generates optimized R code or SQL code (using the `dbplyr` package) that is then executed efficiently on the data source, whether it's an in-memory `tibble` or a 'DuckDB' table. This approach minimizes data transfer and improves performance for large datasets.

The resulting table will be sorted by the relevant date column to ensure the correct temporal ordering of the calculated metrics.

### Value

dbi object

### Examples

```
x <- ytd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard")
calculate(x)
```

---

## `cohort` *Cohort Analysis*

---

### Description

Database-friendly cohort analysis function. A remake of https://github.com/PeerChristensen/cohorts, combining `cohort_table_month`, `cohort_table_year`, and `cohort_table_day` into a single package. Rewritten for database compatibility; runs entirely in SQL without writing tables.

### Usage

```
cohort(.data, .date, .value, time_unit = "month", period_label = FALSE)
```

### Arguments

| | |
|---|---|
| .data | tibble or dbi object |
| .date | date column |
| .value | id column |
| time_unit | do you want summarize the date column to 'day', 'week', 'month','quarter' or 'year' |
| period_label | do you want period labels or the dates c(TRUE , FALSE) |

### Details

- Groups your .value column by shared time attributes from the .date column.
- Assigns each member to a cohort based on their first entry in .date.
- Aggregates the cohort by the time_unit argument (day, week, month, quarter, or year).
- Computes the distinct count of each cohort member over time.

### Value

segment object

---

## dod *Current period day over previous period day*

---

### Description

This calculates the daily value compared to the previous day's value using a standard or 5-5-4 calendar respecting any groups that are passed through with dplyr::group_by()

Use calculate to return the results

### Usage

```
dod(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: atd(), mom(), momtd(), mtd(), mtdopm(), pmtd(), pqtd(), pwtd(), pytd(), qoq(), qoqtd(), qtd(), qtdopq(), wow(), wowtd(), wtd(), wtdopw(), yoy(), yoytd(), ytd(), ytdopy()

## Examples

```
dod(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

---

mom *Current full period month over previous full period month*

---

## Description

This calculates the monthly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with dplyr::group_by()

Use calculate to return the results

## Usage

```
mom(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| .data | tibble or dbi object (either grouped or ungrouped) |
| .date | the date column to group by. Must be a Date or POSIXt type. |
| .value | the value column to summarize. Must be numeric. |
| calendar_type | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| lag_n | the number of periods to lag |

`fiscal_year_start`
: integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there.

`week_start`
: integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect.

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
mom(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

---

`momtd` *Current period month to date compared to previous period month-to-date*

---

## Description

This calculates the monthly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
momtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
momtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard", lag_n=1)
```

`mtd` *Current period month-to-date*

## Description

This calculates the monthly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
mtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group

- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

**NA Handling**

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
mtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard")
```

---

# mtdopm *Current month-to-date over full previous period month*

---

## Description

This calculates the monthly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
mtdopm(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| .data | tibble or dbi object (either grouped or ungrouped) |
| .date | the date column to group by. Must be a Date or POSIXt type. |
| .value | the value column to summarize. Must be numeric. |
| calendar_type | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| lag_n | the number of periods to lag |
| fiscal_year_start | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| week_start | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results

- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
mtdopm(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

---

# pmtd *Previous period month-to-date*

---

## Description

This calculates the monthly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
pmtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: atd(), dod(), mom(), momtd(), mtd(), mtdopm(), pqtd(), pwtd(), pytd(), qoq(), qoqtd(), qtd(), qtdopq(), wow(), wowtd(), wtd(), wtdopw(), yoy(), yoytd(), ytd(), ytdopy()

## Examples

```
pmtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

---

pqtd *Prior period quarter-to-date*

---

## Description

- This calculates the quarterly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with dplyr::group_by()

use calculate to return the results

## Usage

```
pqtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| .data | tibble or dbi object (either grouped or ungrouped) |
| .date | the date column to group by. Must be a Date or POSIXt type. |
| .value | the value column to summarize. Must be numeric. |
| calendar_type | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| lag_n | the number of periods to lag |

`fiscal_year_start`
: integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there.

`week_start`
: integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect.

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
pqtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

---

pwtd *Previous period week-to-date*

---

## Description

This calculates the weekly cumulative sum of targeted value for the previous week using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
pwtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
pwtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

pytd *Previous period year-to-date*

## Description

- For each group, `pytd()` will create the running annual sum of a value based on the calendar type for the previous year compared to the current year calendar date
- If no period exists, it will return `NA`
- The function returns a ti object which prints out the summary of steps and actions that will take to create the calendar table and calculations
- Use calculate to return the results

## Usage

```
pytd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
pytd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

qoq *Current full period quarter over previous full period quarter*

## Description

- This calculates the full quarter value compared to the previous quarter value respecting any groups that are passed through with `dplyr::group_by()`
- Use calculate to return the results

## Usage

```
qoq(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
qoq(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

---

# qoqtd *Current period quarter-to-date compared to previous period quarter-to-date*

---

## Description

- This calculates the annual cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`
- Use calculate to return the results

## Usage

```
qoqtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |

week_start — integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (wtd(), pwtd(), wow(), wowtd(), wtdopw()) and the week column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect.

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with dplyr::group_by(), it will generate a complete calendar for each group
- The function creates a ti object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the .value column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the .date column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: atd(), dod(), mom(), momtd(), mtd(), mtdopm(), pmtd(), pqtd(), pwtd(), pytd(), qoq(), qtd(), qtdopq(), wow(), wowtd(), wtd(), wtdopw(), yoy(), yoytd(), ytd(), ytdopy()

## Examples

```
qoqtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

qtd *Current period quarter-to-date*

## Description

This calculates the quarterly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
qtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group

- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

**NA Handling**

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
qtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard")
```

---

# qtdopq *Current period quarter-to-date over previous period quarter*

---

## Description

- This calculates the quarterly cumulative sum of a targeted value and compares it with the previous full quarter's total value respecting any groups that are passed through with `dplyr::group_by()`
- Use calculate to return the results

## Usage

```
qtdopq(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| .data | tibble or dbi object (either grouped or ungrouped) |
| .date | the date column to group by. Must be a Date or POSIXt type. |
| .value | the value column to summarize. Must be numeric. |
| calendar_type | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| lag_n | the number of periods to lag |
| fiscal_year_start | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| week_start | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (wtd(), pwtd(), wow(), wowtd(), wtdopw()) and the week column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with dplyr::group_by(), it will generate a complete calendar for each group
- The function creates a ti object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the .value column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the .date column are excluded from the calendar join and will not appear in results

- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
qtdopq(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

---

# wow *Current full period week over full previous period week*

---

## Description

This calculates the full week value compared to the previous week value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
wow(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: atd(), dod(), mom(), momtd(), mtd(), mtdopm(), pmtd(), pqtd(), pwtd(), pytd(), qoq(), qoqtd(), qtd(), qtdopq(), wowtd(), wtd(), wtdopw(), yoy(), yoytd(), ytd(), ytdopy()

## Examples

```
wow(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

---

# wowtd *Current period Week-to-date over previous period week-to-date*

---

## Description

This calculates the weekly cumulative sum of targeted value and compares it with the previous week's cumulative sum using a standard or 5-5-4 calendar respecting any groups that are passed through with dplyr::group_by()

Use calculate to return the results

## Usage

```
wowtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| .data | tibble or dbi object (either grouped or ungrouped) |
| .date | the date column to group by. Must be a Date or POSIXt type. |
| .value | the value column to summarize. Must be numeric. |
| calendar_type | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| lag_n | the number of periods to lag |

`fiscal_year_start`
: integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there.

`week_start`
: integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect.

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
wowtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

---

`wtd` *Current period week-to-date*

---

## Description

This calculates the weekly cumulative sum of targeted value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
wtd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
wtd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard")
```

wtdopw *Current period week-to-date over full previous period week*

## Description

This calculates the weekly cumulative sum of targeted value and compares it with the full previous week's total using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`

Use calculate to return the results

## Usage

```
wtdopw(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `yoy()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
wtdopw(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

yoy *Current full period year over previous full period year*

## Description

- This calculates the full year value compared to the previous year value respecting any groups that are passed through with `dplyr::group_by()`
- Use calculate to return the results

## Usage

```
yoy(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years

- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

**NA Handling**

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoytd()`, `ytd()`, `ytdopy()`

## Examples

```
yoy(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

yoytd *Current period year-to-date compared to previous period year-to-date*

## Description

- This calculates the annual cumulative sum of targeted value and compares it with the previous period's annual cumulative to date sum using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`
- Use calculate to return the results

## Usage

```
yoytd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years
- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

### NA Handling

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

### standard calendar

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

### 5-5-4 calendar

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `ytd()`, `ytdopy()`

## Examples

```
yoytd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard",lag_n=1)
```

`ytd` *Current period year-to-date*

## Description

- For each group, `ytd()` will create the running annual sum of a value based on the calendar type specified
- The function returns a ti object which prints out the summary of steps and actions that will take to create the calendar table and calculations
- Use calculate to return the results

## Usage

```
ytd(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years

- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

**NA Handling**

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytdopy()`

## Examples

```
ytd(contoso::sales,.date=order_date,.value=quantity,calendar_type="standard")
```

`ytdopy` *Current period year-to-date compared to full previous period*

## Description

- This calculates the full year value compared to the previous year value using a standard or 5-5-4 calendar respecting any groups that are passed through with `dplyr::group_by()`
- Use calculate to return the results

## Usage

```
ytdopy(
  .data,
  .date,
  .value,
  calendar_type = "standard",
  lag_n = 1,
  fiscal_year_start = 1,
  week_start = 7
)
```

## Arguments

| | |
|---|---|
| `.data` | tibble or dbi object (either grouped or ungrouped) |
| `.date` | the date column to group by. Must be a Date or POSIXt type. |
| `.value` | the value column to summarize. Must be numeric. |
| `calendar_type` | select either 'standard', '445', '454', or '544' calendar, see 'Details' for additional information |
| `lag_n` | the number of periods to lag |
| `fiscal_year_start` | integer 1-12, the month the fiscal year starts nearest to (default 1 = January). Only used with the retail calendars ('445', '454', '544'); the standard calendar always uses calendar years, and a message says so if you change it there. |
| `week_start` | integer 1-7, the day weeks start on (1 = Monday, 7 = Sunday, the default). On the standard calendar it affects only the week functions (`wtd()`, `pwtd()`, `wow()`, `wowtd()`, `wtdopw()`) and the `week` column of daily results; week 1 is the week containing 1 January, so the weeks either side of 1 January can be partial. On the retail calendars it affects every function, because fiscal weeks, months, quarters and years are built from whole weeks starting on this day. A message tells you if you change it where it has no effect. |

## Details

- This function creates a complete calendar object that fills in any missing days, weeks, months, quarters, or years

- If you provide a grouped object with `dplyr::group_by()`, it will generate a complete calendar for each group
- The function creates a `ti` object, which pre-processes the data and arguments for further downstream functions

**NA Handling**

- NA values in the `.value` column propagate through cumulative sums (cumsum returns NA once an NA is encountered)
- NA values in the `.date` column are excluded from the calendar join and will not appear in results
- Missing dates (gaps in your data) are filled with 0 values, not NA

**standard calendar**

- The standard calendar splits the year into 12 months (with 28–31 days each) and uses a 7-day week
- It automatically accounts for leap years every four years to match the Gregorian calendar

**5-5-4 calendar**

- The 5-5-4 calendar divides the fiscal year into 52 weeks (occasionally 53), organizing each quarter into two 5-week periods and one 4-week period.
- This system is commonly used in retail and financial reporting

## Value

ti object

## See Also

Other time_intelligence: `atd()`, `dod()`, `mom()`, `momtd()`, `mtd()`, `mtdopm()`, `pmtd()`, `pqtd()`, `pwtd()`, `pytd()`, `qoq()`, `qoqtd()`, `qtd()`, `qtdopq()`, `wow()`, `wowtd()`, `wtd()`, `wtdopw()`, `yoy()`, `yoytd()`, `ytd()`

## Examples

```
ytdopy(contoso::sales,.date=order_date,.value=quantity,calendar_type='standard',lag_n=1)
```

# Index


∗ **time_intelligence**
atd, 3
dod, 8
mom, 10
momtd, 12
mtd, 14
mtdopm, 15
pmtd, 17
pqtd, 19
pwtd, 21
pytd, 23
qoq, 25
qoqtd, 26
qtd, 28
qtdopq, 29
wow, 31
wowtd, 33
wtd, 35
wtdopw, 37
yoy, 39
yoytd, 41
ytd, 43
ytdopy, 45

abc, 2
abc(), *2, 3*
atd, 3
atd(), *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
augment_standard_calendar, 5

calculate, *2, 3*, 6, *8*, *10*, *12*, *14, 15*, *17*, *19*, *21*, *23*, *25, 26*, *28, 29*, *31*, *33*, *35*, *37*, *39*, *41*, *43*, *45*
calculate,ti::segment_abc-method *(*calculate*)*, 6
calculate,ti::segment_cohort-method *(*calculate*)*, 6
calculate,ti::ti-method *(*calculate*)*, 6

cohort, 7

dod, 8
dod(), *5*, *11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
dplyr::group_by(), *3, 4*, *8–22*, *24–42*, *44–46*

mom, 10
mom(), *5*, *10*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
momtd, 12
momtd(), *5*, *10, 11*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
mtd, 14
mtd(), *5*, *10, 11*, *13*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
mtdopm, 15
mtdopm(), *5*, *10, 11*, *13*, *15*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*

pmtd, 17
pmtd(), *5*, *10, 11*, *13*, *15*, *17*, *20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
pqtd, 19
pqtd(), *5*, *10, 11*, *13*, *15*, *17*, *19*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
pwtd, 21
pwtd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
pytd, 23
pytd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22, 23*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*

qoq, 25

qoq(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
qoqtd, 26
qoqtd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
qtd, 28
qtd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*
qtdopq, 29
qtdopq(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *33, 34*, *36*, *38*, *40*, *42*, *44*, *46*

sum(), *3*

wow, 31
wow(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *34*, *36*, *38*, *40*, *42*, *44*, *46*
wowtd, 33
wowtd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33*, *36*, *38*, *40*, *42*, *44*, *46*
wtd, 35
wtd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *38*, *40*, *42*, *44*, *46*
wtdopw, 37
wtdopw(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *40*, *42*, *44*, *46*

yoy, 39
yoy(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *42*, *44*, *46*
yoytd, 41
yoytd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *44*, *46*
ytd, 43
ytd(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42, 43*, *46*
ytdopy, 45
ytdopy(), *5*, *10, 11*, *13*, *15*, *17*, *19, 20*, *22*, *24*, *26, 27*, *29*, *31*, *33, 34*, *36*, *38*, *40*, *42*, *44*