# Converting numeric values to class "Date"

Mark Eisler and Ana Rabaza

February 11, 2026

## 1 Introduction

For each observation of a subject in a longitudinal study data set, the main **Transition** package functions `add_prev_date()`, `add_prev_result()` and `add_transitions()` all need to identify the previous observation for that same subject, if any. For compatibility with these **Transition** package functions, the timings of observations in a dataset, each referred to as a *timepoint*, should be coded within the data frame as R objects of class `"Date"`, representing calendar dates.

This vignette explains how timepoints represented by numeric values in data may be easily converted to class `"Date"`, using the R **base** package function `as.Date()`.

## 2 Convert numeric values representing year to class "Date"

We start by creating an example data frame of longitudinal data containing years 2018 to 2025 as numeric values for three subjects with observations having one of three possible ordinal values: -

```
> (df <- data.frame(
          subject = rep(1001:1003),
          timepoint = rep(2018:2025, each = 3),
          result = gl(3, 4, lab = c("good", "bad", "ugly"), ordered = TRUE)
      ))
```

```
   subject timepoint result
1     1001      2018   good
2     1002      2018   good
3     1003      2018   good
4     1001      2019   good
5     1002      2019    bad
6     1003      2019    bad
7     1001      2020    bad
8     1002      2020    bad
9     1003      2020   ugly
10    1001      2021   ugly
11    1002      2021   ugly
12    1003      2021   ugly
```

```
13    1001      2022   good
14    1002      2022   good
15    1003      2022   good
16    1001      2023   good
17    1002      2023    bad
18    1003      2023    bad
19    1001      2024    bad
20    1002      2024    bad
21    1003      2024   ugly
22    1001      2025   ugly
23    1002      2025   ugly
24    1003      2025   ugly
```

We convert the numeric values for year in the *timepoint* column to class "Date", using as.Date() with consistent arbitrary values of January 1st for month and day: -

```
> (df <- transform(
          df,
          timepoint = as.Date(paste(timepoint, "01", "01", sep = "-"))
      ))
```

```
   subject  timepoint result
1     1001 2018-01-01   good
2     1002 2018-01-01   good
3     1003 2018-01-01   good
4     1001 2019-01-01   good
5     1002 2019-01-01    bad
6     1003 2019-01-01    bad
7     1001 2020-01-01    bad
8     1002 2020-01-01    bad
9     1003 2020-01-01   ugly
10    1001 2021-01-01   ugly
11    1002 2021-01-01   ugly
12    1003 2021-01-01   ugly
13    1001 2022-01-01   good
14    1002 2022-01-01   good
15    1003 2022-01-01   good
16    1001 2023-01-01   good
17    1002 2023-01-01    bad
18    1003 2023-01-01    bad
19    1001 2024-01-01    bad
20    1002 2024-01-01    bad
21    1003 2024-01-01   ugly
22    1001 2025-01-01   ugly
23    1002 2025-01-01   ugly
24    1003 2025-01-01   ugly
```

We can now use the add_prev_result() function with default values for all but the first argument to add a column of results from the previous observation: -

```
> (df <- add_prev_result(df))
```

```
   subject  timepoint result prev_result
1     1001 2018-01-01   good        <NA>
2     1002 2018-01-01   good        <NA>
3     1003 2018-01-01   good        <NA>
4     1001 2019-01-01   good        good
5     1002 2019-01-01    bad        good
6     1003 2019-01-01    bad        good
7     1001 2020-01-01    bad        good
8     1002 2020-01-01    bad         bad
9     1003 2020-01-01   ugly         bad
10    1001 2021-01-01   ugly         bad
11    1002 2021-01-01   ugly         bad
12    1003 2021-01-01   ugly        ugly
13    1001 2022-01-01   good        ugly
14    1002 2022-01-01   good        ugly
15    1003 2022-01-01   good        ugly
16    1001 2023-01-01   good        good
17    1002 2023-01-01    bad        good
18    1003 2023-01-01    bad        good
19    1001 2024-01-01    bad        good
20    1002 2024-01-01    bad         bad
21    1003 2024-01-01   ugly         bad
22    1001 2025-01-01   ugly         bad
23    1002 2025-01-01   ugly         bad
24    1003 2025-01-01   ugly        ugly
```

Finally, we can format the class `"Date"` *timepoint* column to show just the year, as in the original data: -

```
> transform(df, timepoint = format(timepoint, "%Y"))
```

```
   subject timepoint result prev_result
1     1001      2018   good        <NA>
2     1002      2018   good        <NA>
3     1003      2018   good        <NA>
4     1001      2019   good        good
5     1002      2019    bad        good
6     1003      2019    bad        good
7     1001      2020    bad        good
8     1002      2020    bad         bad
9     1003      2020   ugly         bad
10    1001      2021   ugly         bad
11    1002      2021   ugly         bad
12    1003      2021   ugly        ugly
13    1001      2022   good        ugly
14    1002      2022   good        ugly
15    1003      2022   good        ugly
16    1001      2023   good        good
17    1002      2023    bad        good
18    1003      2023    bad        good
19    1001      2024    bad        good
20    1002      2024    bad         bad
```

```
21    1003      2024   ugly          bad
22    1001      2025   ugly          bad
23    1002      2025   ugly          bad
24    1003      2025   ugly         ugly
```

## 3 Convert numeric values representing year and month to class "Date"

We create another example data frame of longitudinal data containing year and month July 2024 to June 2025 as numeric values for two subjects with observations having one of two possible ordinal values: -

```
> (df <- data.frame(
          subject = 1001:1002,
          year = rep(2024:2025, each = 12),
          month = rep(c(7:12, 1:6), each = 2),
          result = gl(2, 3, lab = c("low", "high"), ordered = TRUE)
      ))
```

```
   subject year month result
1     1001 2024     7    low
2     1002 2024     7    low
3     1001 2024     8    low
4     1002 2024     8   high
5     1001 2024     9   high
6     1002 2024     9   high
7     1001 2024    10    low
8     1002 2024    10    low
9     1001 2024    11    low
10    1002 2024    11   high
11    1001 2024    12   high
12    1002 2024    12   high
13    1001 2025     1    low
14    1002 2025     1    low
15    1001 2025     2    low
16    1002 2025     2   high
17    1001 2025     3   high
18    1002 2025     3   high
19    1001 2025     4    low
20    1002 2025     4    low
21    1001 2025     5    low
22    1002 2025     5   high
23    1001 2025     6   high
24    1002 2025     6   high
```

We convert numeric values for year and month to class "Date", using as.Date() with a consistent arbitrary value of 1st for day of the month: -

```
> (df <- transform(
          df,
          timepoint = as.Date(paste(year, month, "01", sep = "-")),
          year = NULL,
          month = NULL
      ))
```

```
   subject result  timepoint
1     1001    low 2024-07-01
2     1002    low 2024-07-01
3     1001    low 2024-08-01
4     1002   high 2024-08-01
5     1001   high 2024-09-01
6     1002   high 2024-09-01
7     1001    low 2024-10-01
8     1002    low 2024-10-01
9     1001    low 2024-11-01
10    1002   high 2024-11-01
11    1001   high 2024-12-01
12    1002   high 2024-12-01
13    1001    low 2025-01-01
14    1002    low 2025-01-01
15    1001    low 2025-02-01
16    1002   high 2025-02-01
17    1001   high 2025-03-01
18    1002   high 2025-03-01
19    1001    low 2025-04-01
20    1002    low 2025-04-01
21    1001    low 2025-05-01
22    1002   high 2025-05-01
23    1001   high 2025-06-01
24    1002   high 2025-06-01
```

We can now use the `add_transitions()` function with default values for all but the first argument to add a column of transitions: -

```
> (df <- add_transitions(df))
```

```
   subject result  timepoint transition
1     1001    low 2024-07-01         NA
2     1002    low 2024-07-01         NA
3     1001    low 2024-08-01          0
4     1002   high 2024-08-01          1
5     1001   high 2024-09-01          1
6     1002   high 2024-09-01          0
7     1001    low 2024-10-01         -1
8     1002    low 2024-10-01         -1
9     1001    low 2024-11-01          0
10    1002   high 2024-11-01          1
11    1001   high 2024-12-01          1
12    1002   high 2024-12-01          0
13    1001    low 2025-01-01         -1
14    1002    low 2025-01-01         -1
```

```
15     1001    low 2025-02-01          0
16     1002   high 2025-02-01          1
17     1001   high 2025-03-01          1
18     1002   high 2025-03-01          0
19     1001    low 2025-04-01         -1
20     1002    low 2025-04-01         -1
21     1001    low 2025-05-01          0
22     1002   high 2025-05-01          1
23     1001   high 2025-06-01          1
24     1002   high 2025-06-01          0
```

Finally, we can format the class `"Date"` *timepoint* column to show just the month and year, as in the original data: -

```
> transform(df, timepoint = format(timepoint, "%b-%Y"))
```

```
   subject result timepoint transition
1     1001    low  Jul-2024         NA
2     1002    low  Jul-2024         NA
3     1001    low  Aug-2024          0
4     1002   high  Aug-2024          1
5     1001   high  Sep-2024          1
6     1002   high  Sep-2024          0
7     1001    low  Oct-2024         -1
8     1002    low  Oct-2024         -1
9     1001    low  Nov-2024          0
10    1002   high  Nov-2024          1
11    1001   high  Dec-2024          1
12    1002   high  Dec-2024          0
13    1001    low  Jan-2025         -1
14    1002    low  Jan-2025         -1
15    1001    low  Feb-2025          0
16    1002   high  Feb-2025          1
17    1001   high  Mar-2025          1
18    1002   high  Mar-2025          0
19    1001    low  Apr-2025         -1
20    1002    low  Apr-2025         -1
21    1001    low  May-2025          0
22    1002   high  May-2025          1
23    1001   high  Jun-2025          1
24    1002   high  Jun-2025          0
```