

dfidx

Yves Croissant

June 8, 2025

In some situations, series from a data frame have a natural two-dimensional (tabular) representation because each observation can be uniquely characterized by a combination of two indexes. Two major cases of these situations in applied econometrics are:

- panel data, where the same individuals are observed for several time periods,
- random utility models, where each observation describes the features of an alternative among a set of alternatives for a given choice situation.

The idea of **dfidx** is to keep in the same object the data and the information about its structure. A **dfidx** object is a data frame with an **idx** column, which is a data frame that contains the series that define the indexes.

From version 0.1-2, **dfidx** doesn't depend anymore on some of **tidyverse** packages. If you want to use **dfidx** along with **tidyverse** in order to use tibbles instead of ordinary data frames and **dplyr**'s verbs, you should use the new **tidydfidx** package instead of **dfidx**.

1 Basic use of the dfidx function

The **dfidx** package is loaded using:

```
library(dfidx)
```

To illustrate the features of **dfidx**, we'll use the **munnell** data set (Munnell 1990) that is used in Baltagi (2013)'s famous book and is part of the **plm** package as **Produc**. It contains several economic series for American states from 1970 to 1986. We've added to the initial data set a **president** series which indicates the name of the American president in power for the given year.

```
head(munnell, 3)
```

	state	year	region	president	publiccap	highway	water
1	Alabama	1970	East-South Central	Nixon	15032.67	7325.80	1655.68
2	Alabama	1971	East-South Central	Nixon	15501.94	7525.94	1721.02
3	Alabama	1972	East-South Central	Nixon	15972.41	7765.42	1764.75

	utilities	privatecap	gsp	labor	unemp
1	6051.20	35793.80	28418	1010.5	4.7
2	6254.98	37299.91	29375	1021.9	5.2
3	6442.23	38670.30	31303	1072.3	4.7

The two indexes are `state` and `year` and both are nested in another variable: `state` in `region` and `year` in `president`. A `dfidx` object is created with the `dfidx` function: the first argument should be a data frame (or a tibble) and the second argument `idx` is used to indicate the indexes. As, in the `munnell` data set, the first two columns contain the two indexes, the `idx` argument is not mandatory and a `dfidx` can be obtained from the `munnell` tibble simply by using:

```
dfidx(munnell) |> print(n = 3)
```

```
~~~~~
first 3 observations out of 816
~~~~~
```

	region	president	publiccap	highway	water	utilities
1	East-South Central	Nixon	15032.67	7325.80	1655.68	6051.20
2	East-South Central	Nixon	15501.94	7525.94	1721.02	6254.98
3	East-South Central	Nixon	15972.41	7765.42	1764.75	6442.23

	privatecap	gsp	labor	unemp	idx
1	35793.80	28418	1010.5	4.7	Alab:1970
2	37299.91	29375	1021.9	5.2	Alab:1971
3	38670.30	31303	1072.3	4.7	Alab:1972


```
~~~ indexes ~~~~
state year
1 Alabama 1970
2 Alabama 1971
3 Alabama 1972
indexes: 1, 2
```

The resulting object is of class `dfidx` and is a tibble with an `idx` column, which is a tibble containing the two indexes. Note that the two indexes are now longer standalone series in the resulting tibble, because the default value of the `drop.index` argument is `TRUE`. The header of the tibble indicates the names and the cardinal of the two indexes. It also indicated whether

the data set is balanced ie, in this panel data context, whether all the states are observed for the same set of years (which is the case for the `munnell` data set). The `idx` column can be retrieved using the `idx` function:

```
munnell |> dfidx() |> idx() |> print(n = 3)
```

```
~~~ indexes ~~~~
      state year
1 Alabama 1970
2 Alabama 1971
3 Alabama 1972
indexes: 1, 2
```

If the first two columns don't contain the indexes, the `idx` argument should be set. If the observations are ordered first by the first index and then by the second one and if the data set is *balanced*, `idx` can be an integer, the number of distinct values of the first index:

```
munnell |> dfidx(48) |> print(n = 3)
```

```
~~~~~
first 3 observations out of 816
~~~~~
```

	state	year	region	president	publiccap	highway	water
1	Alabama	1970	East-South Central	Nixon	15032.67	7325.80	1655.68
2	Alabama	1971	East-South Central	Nixon	15501.94	7525.94	1721.02
3	Alabama	1972	East-South Central	Nixon	15972.41	7765.42	1764.75

```

      utilities privatecap   gsp   labor unemp idx
1    6051.20    35793.80 28418 1010.5   4.7 1:1
2    6254.98    37299.91 29375 1021.9   5.2 1:2
3    6442.23    38670.30 31303 1072.3   4.7 1:3

```

```

~~~ indexes ~~~~
      id1 id2
1      1   1
2      1   2
3      1   3
indexes: 1, 2

```

Then the two indexes are created with the default names `id1` and `id2`. More relevant names can be indicated using the `idnames` argument and the values of the second index can be indicated, using the `levels` argument.

```
munnell |> dfidx(48, idnames = c("state", "year"),
               levels = 1970:1986) |> print(n = 3)
```

~~~~~

first 3 observations out of 816

~~~~~

	region	president	publiccap	highway	water	utilities
1	East-South Central	Nixon	15032.67	7325.80	1655.68	6051.20
2	East-South Central	Nixon	15501.94	7525.94	1721.02	6254.98
3	East-South Central	Nixon	15972.41	7765.42	1764.75	6442.23

	privatecap	gsp	labor	unemp	idx
1	35793.80	28418	1010.5	4.7	1:1970
2	37299.91	29375	1021.9	5.2	1:1971
3	38670.30	31303	1072.3	4.7	1:1972

~~~ indexes ~~~~

|   | state | year |
|---|-------|------|
| 1 | 1     | 1970 |
| 2 | 1     | 1971 |
| 3 | 1     | 1972 |

indexes: 1, 2

The `idx` argument can also be a character of length one or two. In the first case, only the first index is indicated:

```
munnell |> dfidx("state", idnames = c(NA, "date"),
               levels = 1970:1986) |> print(n = 3)
```

~~~~~

first 3 observations out of 816

~~~~~

|   | year | region             | president | publiccap | highway | water   |
|---|------|--------------------|-----------|-----------|---------|---------|
| 1 | 1970 | East-South Central | Nixon     | 15032.67  | 7325.80 | 1655.68 |
| 2 | 1971 | East-South Central | Nixon     | 15501.94  | 7525.94 | 1721.02 |
| 3 | 1972 | East-South Central | Nixon     | 15972.41  | 7765.42 | 1764.75 |

  

|   | utilities | privatecap | gsp   | labor  | unemp | idx       |
|---|-----------|------------|-------|--------|-------|-----------|
| 1 | 6051.20   | 35793.80   | 28418 | 1010.5 | 4.7   | Alab:1970 |
| 2 | 6254.98   | 37299.91   | 29375 | 1021.9 | 5.2   | Alab:1971 |
| 3 | 6442.23   | 38670.30   | 31303 | 1072.3 | 4.7   | Alab:1972 |

```

~~~ indexes ~~~~
 state date
1 Alabama 1970
2 Alabama 1971
3 Alabama 1972
indexes: 1, 2

```

Note that we've only provided a name for the second index, the NA in the first position of the `idnames` argument meaning that we want to keep the original name for the first index. Finally, if the `idx` argument is a character of length 2, it should contain the name of the two indexes.

```
munnell |> dfidx(c("state", "year")) |> print(n = 3)
```

```

~~~~~
first 3 observations out of 816
~~~~~
 region president publiccap highway water utilities
1 East-South Central Nixon 15032.67 7325.80 1655.68 6051.20
2 East-South Central Nixon 15501.94 7525.94 1721.02 6254.98
3 East-South Central Nixon 15972.41 7765.42 1764.75 6442.23
 privatecap gsp labor unemp idx
1 35793.80 28418 1010.5 4.7 Alab:1970
2 37299.91 29375 1021.9 5.2 Alab:1971
3 38670.30 31303 1072.3 4.7 Alab:1972

~~~ indexes ~~~~
      state year
1 Alabama 1970
2 Alabama 1971
3 Alabama 1972
indexes: 1, 2

```

## 2 More advanced use of `dfidx`

### 2.1 Nesting structure

One or both of the indexes may be nested in another series. In this case, the `idx` argument is still a character of length two, but the nesting series is indicated as the name of the corresponding index:

```
mn <- munnell |> dfidx(c(region = "state", "year"))
mn <- munnell |> dfidx(c(region = "state", president = "year"))
mn |> print(n = 3)
```

~~~~~

first 3 observations out of 816

~~~~~

|   | publiccap | highway  | water   | utilities | privatecap | gsp    | labor  | unemp |
|---|-----------|----------|---------|-----------|------------|--------|--------|-------|
| 1 | 62200.83  | 26835.52 | 7669.60 | 27695.71  | 146285.6   | 168627 | 4655.5 | 6.2   |
| 2 | 63096.48  | 27300.22 | 8004.98 | 27791.28  | 150855.5   | 173767 | 4788.8 | 6.0   |
| 3 | 63642.81  | 27247.22 | 8491.35 | 27904.24  | 156752.0   | 173817 | 4880.0 | 5.5   |

idx

1 Illi:1977  
2 Illi:1978  
3 Illi:1979

~~~ indexes ~~~~

| | state | region | year | president |
|---|----------|--------------------|------|-----------|
| 1 | Illinois | East-North Central | 1977 | Carter |
| 2 | Illinois | East-North Central | 1978 | Carter |
| 3 | Illinois | East-North Central | 1979 | Carter |

indexes: 1, 1, 2, 2

The `idx` column is now a tibble containing the two indexes and the nesting variables.

```
idx(mn) |> print(n = 3)
```

~~~ indexes ~~~~

|   | state    | region             | year | president |
|---|----------|--------------------|------|-----------|
| 1 | Illinois | East-North Central | 1977 | Carter    |
| 2 | Illinois | East-North Central | 1978 | Carter    |
| 3 | Illinois | East-North Central | 1979 | Carter    |

indexes: 1, 1, 2, 2

## 2.2 Customized the name and the position of the `idx` column

By default, the column that contains the indexes is called `idx` and is the first column of the returned data frame. The position and the name of this column can be set using the `position` and `name` arguments:

```
dfidx(munnell, idx = c(region = "state", president = "year"),
      name = "index", position = 4) |> print(n = 3)
```

~~~~~

first 3 observations out of 816

~~~~~

|   | publiccap | highway  | water   | index     | utilities | privatecap | gsp    |
|---|-----------|----------|---------|-----------|-----------|------------|--------|
| 1 | 62200.83  | 26835.52 | 7669.60 | Illi:1977 | 27695.71  | 146285.6   | 168627 |
| 2 | 63096.48  | 27300.22 | 8004.98 | Illi:1978 | 27791.28  | 150855.5   | 173767 |
| 3 | 63642.81  | 27247.22 | 8491.35 | Illi:1979 | 27904.24  | 156752.0   | 173817 |

  

|   | labor  | unemp |
|---|--------|-------|
| 1 | 4655.5 | 6.2   |
| 2 | 4788.8 | 6.0   |
| 3 | 4880.0 | 5.5   |

~~~ indexes ~~~~

| | state | region | year | president |
|---|----------|--------------------|------|-----------|
| 1 | Illinois | East-North Central | 1977 | Carter |
| 2 | Illinois | East-North Central | 1978 | Carter |
| 3 | Illinois | East-North Central | 1979 | Carter |

indexes: 1, 1, 2, 2

2.3 Data frames in wide format

`dfidx` can deal with data frames in wide format, i.e., for which each series for a given value of the second index is a column of the data frame. This is the case of the `munnell_wide` tibble that contains two series of the original data set (`gsp` and `unemp`).

```
head(munnell_wide, 3)
```

| | state | region | gsp_1970 | gsp_1971 | gsp_1972 | gsp_1973 |
|---|----------|--------------------|----------|----------|----------|----------|
| 1 | Alabama | East-South Central | 28418 | 29375 | 31303 | 33430 |
| 2 | Arizona | Mountain | 19288 | 21040 | 23289 | 25244 |
| 3 | Arkansas | West-South Central | 15392 | 16177 | 17702 | 18825 |

| | gsp_1974 | gsp_1975 | gsp_1976 | gsp_1977 | gsp_1978 | gsp_1979 | gsp_1980 |
|---|----------|----------|----------|----------|----------|----------|----------|
| 1 | 33749 | 33604 | 35764 | 37463 | 39964 | 40979 | 40380 |
| 2 | 25698 | 24915 | 26041 | 28110 | 31062 | 33943 | 34708 |
| 3 | 19287 | 19024 | 20277 | 21410 | 23063 | 23501 | 23210 |

| | gsp_1981 | gsp_1982 | gsp_1983 | gsp_1984 | gsp_1985 | gsp_1986 | unemp_1970 |
|---|----------|----------|----------|----------|----------|----------|------------|
| 1 | 41105 | 40328 | 42245 | 45118 | 46849 | 48409 | 4.7 |

| | | | | | | | |
|---|------------|------------|------------|------------|------------|------------|-----|
| 2 | 35244 | 33603 | 35963 | 40010 | 43350 | 46058 | 4.4 |
| 3 | 24134 | 23462 | 24415 | 26512 | 27159 | 28168 | 5.0 |
| | unemp_1971 | unemp_1972 | unemp_1973 | unemp_1974 | unemp_1975 | unemp_1976 | |
| 1 | 5.2 | 4.7 | 3.9 | 5.5 | 7.7 | 6.8 | |
| 2 | 4.7 | 4.2 | 4.1 | 5.6 | 12.0 | 9.8 | |
| 3 | 5.4 | 4.6 | 4.1 | 4.8 | 9.5 | 7.1 | |
| | unemp_1977 | unemp_1978 | unemp_1979 | unemp_1980 | unemp_1981 | unemp_1982 | |
| 1 | 7.4 | 6.3 | 7.1 | 8.8 | 11.0 | 14.0 | |
| 2 | 8.2 | 6.1 | 5.1 | 6.7 | 6.1 | 9.9 | |
| 3 | 6.6 | 6.4 | 6.2 | 7.6 | 9.1 | 9.8 | |
| | unemp_1983 | unemp_1984 | unemp_1985 | unemp_1986 | | | |
| 1 | 14.0 | 11.0 | 8.9 | 9.8 | | | |
| 2 | 9.1 | 5.0 | 6.5 | 6.9 | | | |
| 3 | 10.0 | 8.9 | 8.7 | 8.7 | | | |

Each line is now an American state and, apart the indexes, there are now 34 series with names obtained by the concatenation of the name of the series and the year (for example `gsp_1988`). In this case a supplementary argument called `varying` should be provided. It is a vector of integers indicating the position of the columns that should be merged in the resulting long formatted data frame. The `stats::reshape` function is then used and the `sep` argument can be also provided to indicate the separating character in the names of the series (the default value being `"."`).

```
munnell_wide |> dfidx(varying = 3:36, sep = "_") |> print(n = 3)

~~~~~
first 3 observations out of 816
~~~~~
   state      region  gsp unemp  idx
1 Alabama East-South Central 28418  4.7 1:1970
2 Alabama East-South Central 29375  5.2 1:1971
3 Alabama East-South Central 31303  4.7 1:1972

~~~ indexes ~~~
  id1 id2
1   1 1970
2   1 1971
3   1 1972
indexes: 1, 2
```

Better results can be obtained using the `idx` and `idnames` previously described:


```
munnell_wide |> dfidx(idx = c(region = "state"), varying = 3:36,
  sep = "_", idnames = c(NA, "year")) |>
  print(n = 3)
```

```
~~~~~
first 3 observations out of 816
~~~~~
```

```
      gsp unemp      idx
1 145792   4.1 Illi:1970
2 148503   5.1 Illi:1971
3 154413   5.1 Illi:1972
```

```
~~~ indexes ~~~~
```

```
      state      region year
1 Illinois East-North Central 1970
2 Illinois East-North Central 1971
3 Illinois East-North Central 1972
indexes: 1, 1, 2
```

3 Getting the indexes or their names

The name (and the position) of the `idx` column can be obtained as a named integer (the integer being the position of the column and the name its name) using the `idx_name` function:

```
idx_name(mn)
## idx
## 9
```

To get the name of one of the indexes, the second argument, `n`, is set either to 1 or 2 to get the first or the second index, ignoring the nesting variables:

```
idx_name(mn, 2)
## [1] "year"
idx_name(idx(mn), 2)
## [1] "year"
```

Not that `idx_name` can be in this case applied to a `dfidx` or to a `idx` object. To get a nesting variable, the third argument, called `m`, is set to 2:

```
idx_name(mn, 1, 1)
## [1] "state"
idx_name(mn, 1, 2)
## [1] "region"
```

To extract one or all the indexes, the `idx` function is used. This function has already been encountered when one wants to extract the `idx` column of a `dfidx` object. The same `n` and `m` arguments as for the `idx_name` function can be used in order to extract a specific series. For example, to extract the region index, which nests the state index:

```
id_index1 <- idx(mn, n = 1, m = 2)
id_index2 <- idx(idx(mn), n = 1, m = 2)
head(id_index1)
## [1] "East-North Central" "East-North Central" "East-North Central"
## [4] "East-North Central" "East-North Central" "East-North Central"
identical(id_index1, id_index2)
## [1] TRUE
```

4 Data frames subsetting

Subsets of data frames are obtained using the `[` and the `[[` operators. The former returns most of the time a data frame as the second one always returns a series.

4.1 Commands that return a data frame

Consider first the use of `[`. If one argument is provided, it indicates the columns that should be selected. The result is always a data frame, even if a single column is selected. If two arguments are provided, the first one indicates the subset of lines and the second one the subset of columns that should be returned. If only one column is selected, the result depends on the value of the `drop` argument. If `TRUE`, a series is returned and if `FALSE`, a one series data frame is returned. An important difference between tibbles and ordinary data frames is that the default value of `drop` is `FALSE` for the former and `TRUE` for the later. Therefore, with tibbles, the use of `[` will always by default return a data frame.

A specific `dfidx` method is provided for one reason: the column that contains the indexes should be “sticky” (we borrow this idea from the `sf` package¹), which means that it should be always returned while using the extractor operator, even if it is not explicitly selected.

¹Pebesma and Bivand (2023) and Pebesma (2018).

```
mn[mn$unemp > 10, ] |> print(n = 3)
```

```
~~~~~
```

```
first 3 observations out of 46
```

```
~~~~~
```

```
publiccap highway    water utilities privatecap    gsp labor
1  65064.20 27568.50 10218.00 27277.69 154806.13 159778 4593.3
2  64751.99 27483.00 10436.04 26832.94 157095.80 160856 4530.6
3  25108.79 10618.71 3297.40 11192.68 82361.44 64042 2028.0
unemp      idx
1    11 Illi:1982
2    11 Illi:1983
3    12 Indi:1982
```

```
~~~ indexes ~~~~
```

```
state          region year president
13 Illinois East-North Central 1982 Reagan
14 Illinois East-North Central 1983 Reagan
30 Indiana East-North Central 1982 Reagan
indexes: 1, 1, 2, 2
```

```
mn[mn$unemp > 10, c("highway", "utilities")] |> print(n = 3)
```

```
~~~~~
```

```
first 3 observations out of 46
```

```
~~~~~
```

```
highway utilities      idx
1 27568.50 27277.69 Illi:1982
2 27483.00 26832.94 Illi:1983
3 10618.71 11192.68 Indi:1982
```

```
~~~ indexes ~~~~
```

```
state          region year president
13 Illinois East-North Central 1982 Reagan
14 Illinois East-North Central 1983 Reagan
30 Indiana East-North Central 1982 Reagan
indexes: 1, 1, 2, 2
```

```
mn[mn$unemp > 10, "highway"] |> print(n = 3)
```

```
[1] 27568.50 27483.00 10618.71
~~~ indexes ~~~~
      state      region year president
1 Illinois East-North Central 1977   Carter
2 Illinois East-North Central 1978   Carter
3 Illinois East-North Central 1979   Carter
indexes:  1, 1, 2, 2
```

All the previous commands extract the observations where the unemployment rate is greater than 10% and, in the first case all the series, in the second case two of them and in the third case only one series.

4.2 Commands that return a series

A series can be extracted using any of the following commands:

```
mn1 <- mn[, "highway", drop = TRUE]
mn2 <- mn[["highway"]]
mn3 <- mn$highway
c(identical(mn1, mn2), identical(mn1, mn3))
## [1] TRUE TRUE
```

The result is a **xseries** which inherits the **idx** column from the data frame it has been extracted from as an attribute :

```
mn1 |> print(n = 3)
```

```
[1] 26835.52 27300.22 27247.22
~~~ indexes ~~~~
      state      region year president
1 Illinois East-North Central 1977   Carter
2 Illinois East-North Central 1978   Carter
3 Illinois East-North Central 1979   Carter
indexes:  1, 1, 2, 2
```

```
class(mn1)
```

```
[1] "xseries" "numeric"
```

```
idx(mn1) |> print(n = 3)
```

```
~~~ indexes ~~~~
      state      region year president
1 Illinois East-North Central 1977   Carter
2 Illinois East-North Central 1978   Carter
3 Illinois East-North Central 1979   Carter
indexes:  1, 1, 2, 2
```

Note that, except when `dfidx` hasn't been used with `drop.index = FALSE`, a series which defines the indexes is dropped from the data frame (but is one of the column of the `idx` column of the data frame). It can be therefore retrieved using:

```
head(mn$idx$president)
```

```
[1] Carter Carter Carter Carter Ford   Ford
Levels: Carter Ford Nixon Reagan
```

or

```
idx(mn)$president |> head()
```

```
[1] Carter Carter Carter Carter Ford   Ford
Levels: Carter Ford Nixon Reagan
```

or more simply by applying the `$` operator as if the series were a stand-alone series in the data frame :

```
mn$president |> print(n = 3)
```

```
[1] Carter Carter Carter
Levels: Carter Ford Nixon Reagan
~~~ indexes ~~~~
      state      region year president
1 Illinois East-North Central 1977   Carter
2 Illinois East-North Central 1978   Carter
3 Illinois East-North Central 1979   Carter
indexes:  1, 1, 2, 2
```

In this last case, the resulting series is a `xseries`, *ie* it inherits the index data frame as an attribute.

4.3 User defined class for extracted series

While creating the `dfidx`, a `pkg` argument can be indicated, so that the resulting `dfidx` object and its series are respectively of class `c("dfidx_pkg", "dfidx")` and `c("xseries_pkg", "xseries")` which enables the definition of special methods for `dfidx` and `xseries` objects. For example, consider the hypothetical `pnl` package for panel data:

```
mn <- dfidx(munnell, idx = c(region = "state", president = "year"),
           pkg = "pnl")

mn1 <- mn$gsp
class(mn)
## [1] "dfidx_pnl" "dfidx"      "data.frame"
class(mn1)
## [1] "xseries_pnl" "xseries"    "integer"
```

For example, we want to define a `lag` method for `xseries_pnl` objects. While lagging there should be a NA not only on the first position of the resulting vector like for time-series, but each time we encounter a new individual. A minimal `lag` method could therefore be written as:

```
lag.xseries_pnl <- function(x, ...){
  .idx <- idx(x)
  class <- class(x)
  x <- unclass(x)
  id <- .idx[[1]]
  lgt <- length(id)
  lagid <- c("", id[- lgt])
  sameid <- lagid == id
  x <- c(NA, x[- lgt])
  x[! sameid] <- NA
  structure(x, class = class, idx = .idx)
}

lmn1 <- stats::lag(mn1)
lmn1 |> print(n = 3)
```

```
[1]      NA 168627 173767
~~~ indexes ~~~~
      state      region year president
1 Illinois East-North Central 1977   Carter
2 Illinois East-North Central 1978   Carter
3 Illinois East-North Central 1979   Carter
indexes:  1, 1, 2, 2
```

```
class(lmn1)
```

```
[1] "xseries_pnl" "xseries"      "integer"
```

```
rbind(mn1, lmn1)[, 1:20]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
mn1 168627 173767 173817 165722 157366 163112 145792 148503 154413
lmn1    NA 168627 173767 173817 165722 157366 163112 145792 148503
      [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18]
mn1 163125 161725 166029 159778 160856 173602 178493 183849 68832
lmn1 154413 163125 161725 166029 159778 160856 173602 178493    NA
      [,19] [,20]
mn1  71717 72047
lmn1 68832 71717
```

Note the use of `stats::lag` instead of `lag` which ensures that the `stats::lag` function is used, even if the **dplyr** (or **tidyverse**) package is attached.

5 Model building

The two main steps in **R** in order to estimate a model are to use the `model.frame` function to construct a data frame, using a formula and a data frame and then to extract from it the matrix of covariates using the `model.matrix` function.

5.1 Model frame

The default method of `model.frame` has as first two arguments `formula` and `data`. It returns a data frame with a `terms` attribute. Some other methods exist in the **stats** package, for example for `lm` and `glm` object with a first and main argument called `formula`. This is quite unusual and misleading as for most of the generic functions in **R**, the first argument is called either `x` or `object`.

Another noticeable method for `model.frame` is provided by the **Formula** package and, in this case, the first argument is a `Formula` object, which is an extended formula which can contain several parts on the left and/or on the right hand side of the formula.

We provide a `model.frame` method for `dfidx` objects, mainly because the `idx` column should be returned in the resulting data frame. This leads to an unusual order of the arguments, the

data frame first and then the formula. The method then first extract (and subset if necessary the `idx` column), call the `formula/Formula` method and then add to the resulting data frame the `idx` column. The resulting data frame is a `dfidx` object.

```
mf_mn <- mn |> model.frame(gsp ~ utilities + highway | unemp | labor,
                           subset = unemp > 10)
mf_mn |> print(n = 3)
```

```
~~~~~
first 3 observations out of 46
~~~~~
```

| | gsp | utilities | highway | unemp | labor | idx |
|---|--------|-----------|----------|-------|--------|-----------|
| 1 | 159778 | 27277.69 | 27568.50 | 11 | 4593.3 | Illi:1982 |
| 2 | 160856 | 26832.94 | 27483.00 | 11 | 4530.6 | Illi:1983 |
| 3 | 64042 | 11192.68 | 10618.71 | 12 | 2028.0 | Indi:1982 |

```

~~~ indexes ~~~~
state          region year president
1 Illinois East-North Central 1982   Reagan
2 Illinois East-North Central 1983   Reagan
3 Indiana East-North Central 1982   Reagan
indexes:  1, 1, 2, 2
```

```
formula(mf_mn)
```

```
gsp ~ utilities + highway + unemp + labor + (state + region +
year + president)
```

Note that the column that contains the indexes is at the end and not at the beginning of the returned data frame. This is because the `stats::model.response` function, which is used to extract the response of a model and is not generic consider that the first column of the model frame is the response.

5.2 Model matrix

`model.matrix` is a generic function and for the default method, the first two arguments are a `terms` object and a data frame. In `lm`, the `terms` attribute is extracted from the `model.frame` internally constructed using the `model.frame` function. This means that, at least in this context, `model.matrix` doesn't need a `formula/term` argument and a `data.frame`, but only a data frame returned by the model frame method, i.e., a data frame with a `terms` attribute.

We use this idea for the `model.matrix` method for `dfidx` object; the only required argument is a `dfidx` returned by the `model.frame` function. The formula is then extracted from the `dfidx` and the `Formula` or default method is then called. The result is a matrix of class `dfidx_matrix`, with a printing method that allows the use of the `n` argument:

```
mf_mn |> model.matrix(rhs = 1) |> print(n = 5)

# [46 x 3]
  (Intercept) utilities highway
1           1  27277.69 27568.50
2           1  26832.94 27483.00
3           1  11192.68 10618.71
4           1  11141.74 10558.11
5           1  21281.74 19996.38
```

```
mf_mn |> model.matrix(rhs = 2:3) |> print(n = 5)

# [46 x 3]
  (Intercept) unemp labor
1           1    11 4593.3
2           1    11 4530.6
3           1    12 2028.0
4           1    11 2029.5
5           1    12 3442.8
```

References

- Baltagi, B. H. 2013. *Econometric Analysis of Panel Data*. 5th ed. John Wiley; Sons ltd.
- Munnell, A. 1990. “Why Has Productivity Growth Declined? Productivity and Public Investment.” *New England Economic Review*, 3–22.
- Pebesma, Edzer. 2018. “Simple Features for R: Standardized Support for Spatial Vector Data.” *The R Journal* 10 (1): 439–46. <https://doi.org/10.32614/RJ-2018-009>.
- Pebesma, Edzer, and Roger Bivand. 2023. *Spatial Data Science: With applications in R*. Chapman and Hall/CRC. <https://doi.org/10.1201/9780429459016>.