

# Computing Returns

Package version 0.10-0

Enrico Schumann  
es@enricoschumann.net

In the simplest case, the function returns takes a numeric vector of prices and evaluates to a numeric vector of returns.

```
> library("PMWR")  
> returns(100:105)
```

```
[1] 0.01000 0.00990 0.00980 0.00971 0.00962
```

The function will recognise when the input argument has several columns, i.e. is a matrix or a data frame. In such a case, it computes returns for each column.

In fact, returns is a generic function, and also understands time-series such as zoo objects. To demonstrate this functionality, we use the datasets DAX and REXP, which are provided by PMWR. Both are data frames of one column; the rownames are the dates.

```
> library("zoo")  
> DAX <- zoo(DAX[[1]], as.Date(row.names(DAX)))  
> REXP <- zoo(REXP[[1]], as.Date(row.names(REXP)))
```

Calling returns on a zoo series will result in a zoo series.

```
> str(DAX)
```

```
'zoo' series from 2014-01-02 to 2015-12-30  
Data: num [1:505] 9400 9435 9428 9506 9498 ...  
Index: Date[1:505], format: "2014-01-02" "2014-01-03" "2014-01-06" "2014-01-07" ...
```

```
> head(returns(DAX))
```

```
2014-01-03 2014-01-06 2014-01-07 2014-01-08 2014-01-09 2014-01-10  
0.003735 -0.000758 0.008294 -0.000879 -0.008026 0.005480
```

Matrices work as well: As an example, we combine both zoo series into a two-column matrix.

```
> returns(head(cbind(DAX, REXP), 5))
```

```
          DAX      REXP  
2014-01-03 0.003735 0.000611  
2014-01-06 -0.000758 0.001704  
2014-01-07 0.008294 0.000621  
2014-01-08 -0.000879 -0.000131
```

When a calendar timestamp is available, returns may also aggregate prices over specific holding periods.

```
> returns(DAX, period = "year")
```

```
2014 2015  
4.3  9.6
```

```
> returns(DAX, period = "month")
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	YTD
2014	-1.0	4.1	-1.4	0.5	3.5	-1.1	-4.3	0.7	0.0	-1.6	7.0	-1.8	4.3
2015	9.1	6.6	5.0	-4.3	-0.4	-4.1	3.3	-9.3	-5.8	12.3	4.9	-5.6	9.6

```
> returns(DAX, period = "2015")
```

```
9.6% [30 Dec 2014 -- 30 Dec 2015]
```

```
> returns(DAX, period = "annualised")
```

```
6.9% [02 Jan 2014 -- 30 Dec 2015]
```

Again, this also works for matrices.

```
> returns(cbind(DAX, REXP), period = "year")
```

```

DAX REXP
2014 4.3  7.1
2015 9.6  0.5
```

```
> returns(cbind(DAX, REXP), period = "month")
```

```

      DAX  REXP
2014-01-31 -1.0  1.8
2014-02-28  4.1  0.4
2014-03-31 -1.4  0.1
2014-04-30  0.5  0.3
2014-05-30  3.5  0.9
2014-06-30 -1.1  0.4
2014-07-31 -4.3  0.4
2014-08-29  0.7  1.0
2014-09-30  0.0 -0.1
2014-10-31 -1.6  0.1
2014-11-28  7.0  0.4
2014-12-30 -1.8  1.0
2015-01-30  9.1  0.3
2015-02-27  6.6  0.1
2015-03-31  5.0  0.3
2015-04-30 -4.3 -0.5
2015-05-29 -0.4 -0.2
2015-06-30 -4.1 -0.8
2015-07-31  3.3  0.7
2015-08-31 -9.3  0.0
2015-09-30 -5.8  0.4
2015-10-30 12.3  0.4
2015-11-30  4.9  0.3
2015-12-30 -5.6 -0.6
```

Despite the way these holding-period returns are printed: the result of the function call is a numeric vector (the return numbers), with additional information added through attributes. It is thus natural to compute with the returns, e.g. to calculate means, extremes or similar quantities.

```
> range(returns(DAX, period = "month"))
```

```
[1] -0.0928  0.1232
```

There are methods for toLatex and toHTML for monthly returns. For instance, the table

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	YTD
2014	-1.0	4.1	-1.4	0.5	3.5	-1.1	-4.3	0.7	0.0	-1.6	7.0	-1.8	4.3
2015	9.1	6.6	5.0	-4.3	-0.4	-4.1	3.3	-9.3	-5.8	12.3	4.9	-5.6	9.6

is essentially prepared by the call

```
> toLatex(returns(DAX, period = "month"))
```

See the vignette source for the tabular header. More examples are in the FinT<sub>E</sub>X vignette; say

```
> vignette("FinTeX", package = "PMwR" )
```

to open it.