Log-Returns in Excel 365

From Price Arrays to Return Arrays with Dynamic Arrays and LAMBDA

Gianluca Fusai, Luca Luigi Alberici, Marco Migliardi

Dipartimento SEI
Università del Piemonte Orientale
gianluca.fusai@uniupo.it
Website in U.P.O.

Faculty of Finance
Bayes Business School, London, UK
gianluca.fusai.1@citystgeorges.ac.uk
Website in Bayes

These notes can be freely distributed under the solely requirement that the author's name is explicitly cited.

Main References

Excel File and Slides

- `03_LogReturns.xlsx`
- Lesson page, with the slides and the workbook download:
  `https://gianlucafusai.com/excelling-in-finance/computing-returns`

Lecture Notes

- Fusai, G., Alberici, L. L., Migliardi, M. "Log-Returns in Excel 365: From Price Arrays to Return Arrays with Dynamic Arrays and `LAMBDA`." *Excelling in Finance* tutorial series.

Related Material

- Microsoft documentation: `DROP`, `TAKE`, `INDEX`, `SEQUENCE`, `TRIMRANGE`, `LAMBDA`, `LET`, `ISOMITTED`.
- Campbell, Lo, MacKinlay, *The Econometrics of Financial Markets*, Chap. 1 (returns, aggregation).

Agenda

- The input: a named price column (trim operator `:.`)
- Log-returns vs. simple returns (one-page recap)
- The core idea: **shifted-array alignment**
- Main path: one-period returns via `DROP`
- One rule, seven formulas
- k-period returns: overlapping vs. non-overlapping
- Where the remainder goes: end vs. start indices
- Reusable `LAMBDA` functions: `fx_LogReturn`, `fx_Return`
- Workflow summary, exercises, bridge to the next tutorial

Prices In, Returns Out

© Gianluca Fusai

The Input: a Named Price Column

- Type or paste the closing prices in **one column, oldest on top**. Any source works: a download, a data provider.
- Name the column in Formulas > Name Manager with the trim-range operator `:.`, and count the prices:

```
d_HistPrices    ='LogReturns (1 period)'!$C$9:.$C$1000
n_NObs          =ROWS(d_HistPrices)
```

- `C9:.C1000` is `C9:C1000` without its trailing empty cells: the same as `TRIMRANGE(C9:C1000,2)`.
- The name always ends at the last price, and **grows** when prices are added below.
- Prefixes: `d_` dynamic range (changes size), `n_` any other name (inputs, constants), `fx_` LAMBDA function.
- Requires Excel 365 or Excel for the web.

Fact

Pitfall

Keep the column **empty below the prices**: anything typed there joins the range, and every return computed from it.

From here on every formula uses only `d_HistPrices` and `n_NObs`.

© Gianluca Fusai

Prices In, Returns Out

Old world: `=LN(B2/B1)` in a cell, drag a thousand rows.

Excel 365: one formula, the entire return vector; its length follows the price column automatically.

Every formula in this deck is one of:

Pattern	Excel operator	Role here
Named input	trim operator `:.`, `ROWS`	Input price array and its length
Dynamic reference	`DROP`, `TAKE`, `INDEX`	Aligned shifted views
Pointwise transform	`LN`, `/`, -1	Ratio + logarithm, elementwise

Fact

No recursion here

Unlike the `SCAN` tutorial, nothing in this deck is recursive. A return depends on *two prices at the same time index*, not on the previous return. Pointwise array formulas, not `SCAN`, are the right tool.

Log and Simple Returns in One Slide

With $P(t)$ the price at time t and Δ the sampling step:

$$R_\Delta(t) = \frac{P(t+\Delta)}{P(t)} - 1, \qquad R_\Delta(t) \in (-1, \infty)$$

$$r_\Delta(t) = \ln\left(\tfrac{P(t+\Delta)}{P(t)}\right) = \ln(1 + R_\Delta(t)), \quad r_\Delta(t) \in (-\infty, \infty).$$

Why finance prefers log-returns:

- **Additivity.** $r(t, t+k\Delta) = \sum_{j=0}^{k-1} r_\Delta(t+j\Delta)$ by telescoping. Simple returns compound multiplicatively.
- **Support.** Unbounded below; simple returns are capped at -100%.
- **Symmetry.** r and $-r$ cancel exactly; $-10\%/+10\%$ simple does *not* restore the investment.

Fact

One-line takeaway

Sums are easier to model than products.

Small-move approximation: $r \approx R - R^2/2$; the two measures coincide for small moves and diverge for large ones.

© Gianluca Fusai

The Core Idea

© Gianluca Fusai

This is the only idea of the whole deck.

A return is a *pointwise* transformation of the element-by-element *ratio* of two *aligned* price arrays:

- **Numerator:** later prices $(P_2, P_3, \ldots, P_n)^\top$
- **Denominator:** earlier prices $(P_1, P_2, \ldots, P_{n-1})^\top$
- Both have length $n - 1$: the original array, each missing one endpoint.

Fact

The one rule of return computation

numerator	=	drop the *first* price
denominator	=	drop the *last* price
return	=	`LN(numerator / denominator)`

Every implementation below realises the two drops with a different Excel operator.

The five prices of the sheet `LogReturns (1 period)`: (97.4, 99.2, 94.2, 98.5, 101.8) in `C9:C13`.

i	1	2	3	4
numerator `DROP(d_HistPrices, 1)`	99.2	94.2	98.5	101.8
denominator `DROP(d_HistPrices,-1)`	97.4	99.2	94.2	98.5
$R_i = P_{i+1}/P_i - 1$	+1.85%	−5.04%	+4.56%	+3.35%
$r_i = \ln(P_{i+1}/P_i)$	+1.83%	−5.17%	+4.46%	+3.30%

Fact

What to notice

Two `DROP`s line the array up against its own shifted copy. Pointwise ratio → gross returns. Apply `LN` (or -1) elementwise. No copy-down, no recursion, no size known in advance.

Main Path: `DROP`

One-Period Returns with DROP

Log-returns: one formula (cell E10).

```
=LET(Prices_, d_HistPrices,
    LN(DROP(Prices_, 1) / DROP(Prices_, -1)))
```

Simple returns: replace LN with a trailing -1 (cell F10).

```
=LET(Prices_, d_HistPrices,
    DROP(Prices_, 1) / DROP(Prices_, -1) - 1)
```

Spills into a column of length $n - 1$: four returns from five prices, the first log-return being 1.83%.

Fact

Three reasons to prefer DROP

- **No length counter.** Unlike TAKE, INDEX, OFFSET, DROP does not need to know n: add a price below the column and the returns follow.
- **Non-volatile.** OFFSET recalculates on every workbook change; DROP does not.
- **Readable.** DROP(d_HistPrices, 1) / DROP(d_HistPrices, -1) reads like the shift rule itself.

One Rule, Seven Formulas

© Gianluca Fusai

Seven Formulas, Same Result

Row 10 of `LogReturns (1 period)` writes the rule with seven formulas (`E`–`K`) plus `fx_LogReturn` (`L`), each starting with `=LET(Prices_, d_HistPrices, ...)`. Numerator on top, denominator below; `LN(numerator / denominator)` gives the log-returns.

Col.	Operator	Numerator / Denominator
`E`	`DROP` (main path)	`DROP(Prices_, 1)` `DROP(Prices_, -1)`
`G`	`INDEX` (range form)	`INDEX(Prices_,2,1):INDEX(Prices_,n_NObs,1)` `INDEX(Prices_,1,1):INDEX(Prices_,n_NObs-1,1)`
`H`	`INDEX+SEQUENCE`	`INDEX(Prices_, SEQUENCE(n_NObs-1, 1, 2))` `INDEX(Prices_, SEQUENCE(n_NObs-1))`
`I`	`OFFSET` (volatile)	`OFFSET(Prices_, 1, 0, n_NObs-1, 1)` `OFFSET(Prices_, 0, 0, n_NObs-1, 1)`
`J`	`TAKE`	`TAKE(Prices_, 1-n_NObs)` `TAKE(Prices_, n_NObs-1)`

Column `F` is the simple return. **Row 16:** largest difference of each log-return column from column `E` — all **0**.

A LAMBDA in a Cell, then a Named Function

Column K: an anonymous LAMBDA, defined and called in the same cell.

```
=LAMBDA(Prices_,
  LET(num_, DROP(Prices_, 1), den_, DROP(Prices_, -1),
      LN(num_ / den_)))(d_HistPrices)
```

- Prices_ is the parameter, as in fx_LogReturn; the trailing (d_HistPrices) calls the function.
- Same values as column E: the point is syntactic.

Column L: the same idea, saved in the Name Manager as fx_LogReturn.

```
=fx_LogReturn(d_HistPrices)
```

Fact

From formula to function

Write the LAMBDA in a cell, test it on real data, then save it under a name. The sheet prints each formula with FORMULATEXT just above its column, in row 8.

Pros and Cons

Operator	Readable	Needs *n*?	Volatile?
`DROP`	**yes**	**no**	**no**
`TAKE`	**yes**	yes	no
`INDEX+SEQUENCE`	no	yes	no
`INDEX` (range form)	no	yes	no
`OFFSET`	yes	yes	**yes (legacy)**

Fact

The default choice

`DROP` wins on *all three* criteria simultaneously. Use `DROP` unless a constraint forces otherwise; avoid `OFFSET` in analytics because of volatility.

k-Period Returns

k-period log-return:

$$r^{(k)}(t) \;=\; \ln\left(\frac{P(t+k\Delta)}{P(t)}\right) \;=\; \sum_{j=0}^{k-1} r_\Delta(t+j\Delta).$$

Given n prices, two sampling conventions:

Convention	Count	Windows
Overlapping	$n-k$	Mechanical overlap induces serial dependence
Non-overlapping	$m=\lfloor (n-1)/k \rfloor$	No mechanical overlap between windows

Fact

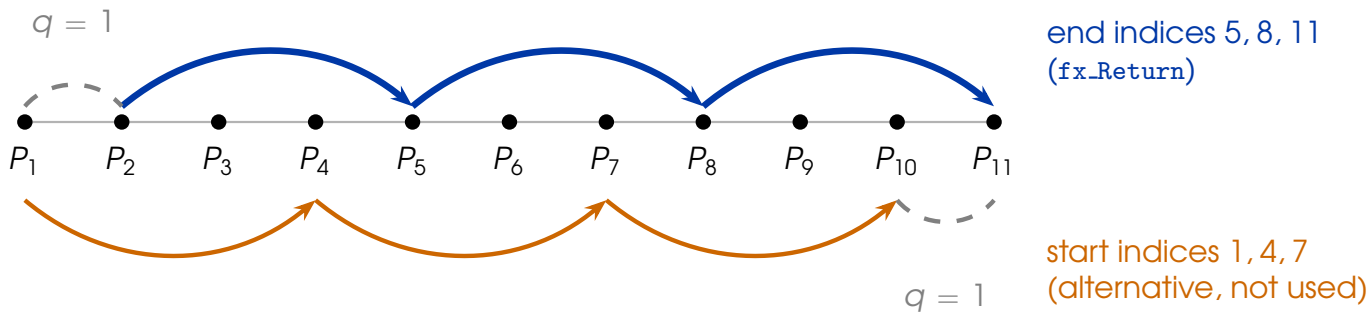
When to use which

Overlapping: descriptive statistics, volatility scaling, large-sample estimators.

Non-overlapping: hypothesis tests, i.i.d.-based models, clean backtests.

Toy Example: $n = 11,\ k = 3$

Scheme	Start (denom.)	End (num.)	Count
Overlapping	$1, 2, \ldots, 8$	$4, 5, \ldots, 11$	$n - k = 8$
Non-overlapping (anchored at the latest price)	$2, 5, 8$	$5, 8, 11$	$\lfloor 10/3 \rfloor = 3$



- End indices: `SORT(SEQUENCE(3, 1, 11, -3))` $= 5, 8, 11$; start $=$ end $- k = 2, 5, 8$.
- The last return ends on P_{11}; the remainder $q =$ `MOD`$(10, 3) = 1$ period ($P_1 \to P_2$) is unused, at the *start*.

Overlapping Windows in the Workbook

Sheet `LogReturns (k period Overlap)`: 22 prices in `G9:G30`, $k = 6$ in `D7` (name `n_k`). One formula, generalising the one-period case:

```
=LET(Prices_, d_HistPrices,
    LN(DROP(Prices_, n_k) / DROP(Prices_, -n_k)))
```

The sheet splits it into columns, to inspect each step:

Cell	Formula	Content
H9	`=SEQUENCE(n_NObs-n_k)`	Start: 1, ... , 16
I9	`=H9#+n_k`	End = Start + k: 7, ... , 22
J9	`=DROP(d_HistPrices,n_k)`	Numerator
K9	`=DROP(d_HistPrices,-n_k)`	Denominator
L9	`=LN(J9#/K9#)`	Log-return
M9	`=fx_Return(d_HistPrices,n_k)`	Same, with the function

Fact

Result

$n - k = 16$ returns; the first is $\ln(49.00/46.99) = 4.19\%$. `D11` = largest difference between `L` and `M` = 0.

Sheet `LogReturns (k period not Overl)`: the same 22 prices, $k = 6$.

Cell	Formula	Content
D9	`=ROUNDDOWN((n_NObs-1)/n_k,0)`	$m = 3$ returns (name `n_m`)
D10	`=MOD(n_NObs-1,n_k)`	$q = 3$ leftover periods
I9	`=SORT(SEQUENCE(n_m,1,n_NObs,-n_k))`	End index: 10, 16, 22
J9	`=I9#-n_k`	Start index: 4, 10, 16
K9	`=INDEX(d_HistPrices,I9#)`	Numerator
L9	`=INDEX(d_HistPrices,J9#)`	Denominator
M9	`=LN(K9#/L9#)`	First: $\ln(45.60/47.92) = -4.96\%$
N9	`=fx_Return(d_HistPrices,n_k,FALSE)`	Same, with the function

Fact

Anchored at the latest price

`SEQUENCE` starts at $n = 22$ and steps back by k; `SORT` puts the end indices in date order. Blocks $P_4 \to P_{10}$, $P_{10} \to P_{16}$, $P_{16} \to P_{22}$: the last return ends on the last price, and the $q = 3$ periods from P_1 to P_4 (grey in the sheet) are unused.

© Gianluca Fusai

Where Does the Remainder Go? End vs. Start Indices

n prices give $n-1$ one-period intervals:

$$n-1 = mk + q, \qquad m = \left\lfloor \tfrac{n-1}{k} \right\rfloor, \qquad q = (n-1) \bmod k, \quad 0 \le q < k.$$

	End indices (`fx_Return`)	**Start indices (alternative)**
Anchor	Last price, steps backward	First price, steps forward
Indices	`SORT(SEQUENCE(n_m,1,n_NObs,-n_k))`	`SEQUENCE(n_m,1,1,n_k)`
Remainder	At the start	At the end
P_n used	Always	Only if $q = 0$
P_1 used	Only if $q = 0$	Always
$n = 22, k = 6$	$P_4 \to P_{10} \to P_{16} \to P_{22}$	$P_1 \to P_7 \to P_{13} \to P_{19}$

The two coincide when $\texttt{MOD}(n-1, k) = 0$, and always for overlapping windows.

Fact

Neither is more correct

End-anchored when the series must end with the latest observation; start-anchored when a fixed start date matters. Choose consciously, and name the function so the choice stays visible.

Reusable LAMBDA Functions

Defined once in the Name Manager; called anywhere in the workbook as `=fx_LogReturn(d_HistPrices)`.

```
                most recent on the bottom, normally the named
                range d_HistPrices (see "The price series").

=LAMBDA(Prices_,
  LET(
    Text1_, "Prices_: one column, oldest price on top",
    Text2_, "Shift rule: drop the first price (numerator), drop the last price (denominator)",
    num_,   DROP(Prices_, 1),
    den_,   DROP(Prices_, -1),
    Text3_, "One-period log-returns: n - 1 values, in date order",
```

- `num_ = DROP(Prices_, 1)` is the numerator, `den_ = DROP(Prices_, -1)` the denominator; `LN(num_ / den_)` applies pointwise. It *is* the shift rule.

Fact

Naming convention

Parameters and `LET` variables end with an underscore (`Prices_`, `num_`, `den_`), so they never clash with cell addresses or workbook names (`d_HistPrices`, `n_k`). `Text1_`, `Text2_`, ... are comments: `LET` accepts names that are never used.

One function for every case: k periods, overlapping or not, log or simple. Defaults: $k = 1$, overlapping, log.

```
 1/0 (in Excel TRUE = 1 is FALSE, so IF(overlap = 1, ...) ignored
 TRUE).

=LAMBDA(Prices_, [k_], [overlap_], [log_],
  LET(
    Text1_,  "Defaults: one period, overlapping windows, log-returns (flags accept 1/0 or TRUE/FALSE)",
    h_,      IF(ISOMITTED(k_), 1, k_),
    ov_,     IF(ISOMITTED(overlap_), TRUE, overlap_),
    lg_,     IF(ISOMITTED(log_), TRUE, log_),
    Text2_,  "Non-overlapping blocks end at n, n-k, n-2k, ...: anchored at the latest price, sorted into date order",
    n_,      ROWS(Prices_),
    endIx_,  SORT(SEQUENCE(ROUNDDOWN((n_ - 1) / h_, 0), 1, n_, -h_)),
    Text3_,  "Shift rule: numerator = later prices, denominator = prices k periods earlier",
    num_,    IF(ov_, DROP(Prices_, h_), INDEX(Prices_, endIx_, 1)),
    den_,    IF(ov_, DROP(Prices_, -h_), INDEX(Prices_, endIx_ - h_, 1)),
    Text4_,  "Gross return, then log-return or simple return",
    gross_,  num_ / den_,
```

overlap_	log_	**Output (in date order)**
1 / TRUE	1 / TRUE	Overlapping k-period log-returns (default)
1 / TRUE	0 / FALSE	Overlapping k-period simple returns
0 / FALSE	1 / TRUE	Non-overlapping k-period log-returns
0 / FALSE	0 / FALSE	Non-overlapping k-period simple returns

- **Optional arguments.** `[k_]`, `[overlap_]`, `[log_]` may be omitted; `ISOMITTED` supplies the defaults `h_` $= 1$, `ov_` $=$ `TRUE`, `lg_` $=$ `TRUE`.
- **Overlapping.** `DROP(Prices_, h_)` / `DROP(Prices_, -h_)`: $n - k$ returns.
- **Non-overlapping.** `endIx_` $=$ end indices $n, n-k, n-2k, \ldots$, sorted into date order; `INDEX` reads the prices at `endIx_` and `endIx_ - h_`: `ROUNDDOWN((n_ - 1) / h_, 0)` returns, the last one ending on the last price.
- **Flags.** `IF(ov_, ...)`, `IF(lg_, ...)`: `TRUE` or any non-zero number takes the first branch, so 1/0 and `TRUE`/`FALSE` both work.
- **Return type.** `gross_` $=$ `num_ / den_`, then `LN(gross_)` or `gross_ - 1`.

```
= fx_Return(d_HistPrices)              -- one period, log (= fx_LogReturn)
= fx_Return(d_HistPrices, 5)           -- 5 periods, overlapping, log
= fx_Return(d_HistPrices, 5, FALSE)    -- 5 periods, non-overlapping, log
= fx_Return(d_HistPrices, 21, 0, 0)    -- 21 periods, non-overlapping, simple
```

Sheet `fx_Return` ($k = 5$): the four cases in columns `I`–`L`, your choice in `H` (`n_k`, `n_overlap`, `n_log`), and four checks, all **0**:

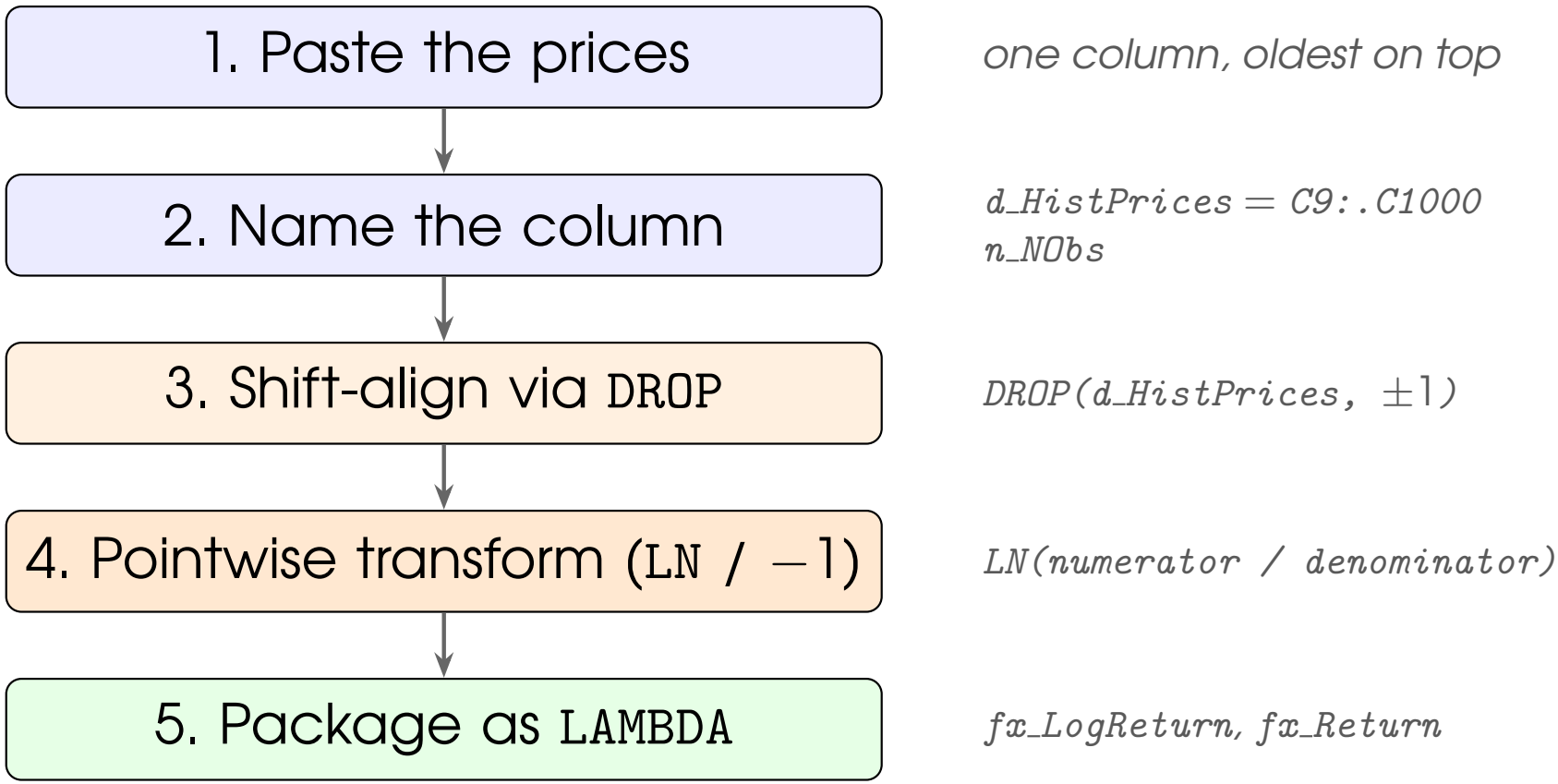
- one period: `fx_Return` $=$ `fx_LogReturn`;
- telescoping: sum of the log-returns $= \ln(P_n/P_1)$;
- $\ln(1+R) = r$;
- `TRUE/FALSE` $= 1/0$.

Fact

One formula, the entire pipeline

Every downstream formula in the workbook becomes a single `fx_Return(...)` call. The sampling logic is centralised in one place; any change propagates automatically.

Workflow and Bridge



Every formula in this tutorial is a step in this single pipeline.

© Gianluca Fusai

The Central Message

Fact

A return is an aligned ratio of two shifted price arrays. Dynamic arrays — and `DROP` *in particular — let Excel carry out that alignment in one formula.*

Two takeaways:

- Classical spreadsheets ask the analyst to *enumerate* the rows. Dynamic arrays ask the analyst to *describe* the array. The descriptive perspective is shorter and more robust.
- Nothing in this deck is recursive. Compare with the `SCAN` tutorial, where the running statistics M_t, MDD_t, etc. *are* recursive. Knowing which side of the divide a new formula lives on is the single most transferable mental reflex.

Exercises

All on `03_LogReturns.xlsx`.

1. **Telescoping.** Check that `=SUM(E10#)` equals $\ln(101.8/97.4) = 4.42\%$; repeat with the simple returns in column `F` and explain the gap.
2. **The trimmed range.** Type a sixth price in `C14`: does every column of row 10 gain a return? Then type a note in `C20`: explain the errors.
3. **Changing k.** On the non-overlapping sheet set `n_k` to 7, then to 5. How many returns, how many grey periods? Check against m and q.
4. **End vs. start indices.** Build the start-anchored version with `SEQUENCE(n_m,1,1,n_k)` and compare it with `fx_Return(d_HistPrices,n_k,FALSE)`. When do they agree?
5. **Volatility scaling.** Annualise the standard deviation of one-period log-returns with $\sqrt{252}$, $\sqrt{52}$ or $\sqrt{12}$. Why is the $\sqrt{k}$ rule exact only for uncorrelated returns?

Next tutorial: Descriptive Statistics of Financial Returns.

The return array produced here is the direct input to:

- sample mean, variance, skewness, kurtosis;
- autocorrelation and Ljung–Box test;
- histogram, QQ-plot, kernel density;
- stationarity diagnostics.

Further down the series:

- Covariance matrix estimation, GMV portfolio, efficient frontier.
- EWMA and GARCH volatility modelling (recursive again, via `SCAN`).
- Brownian-motion simulation and option pricing.

The `SCAN`-based running statistics of the `SCAN` tutorial and the pointwise return construction of this one are the two basic building blocks on which the rest of the series rests.

© Gianluca Fusai

Thank You

gianluca.fusai@uniupo.it
gianluca.fusai.1@citystgeorges.ac.uk