align: : CHEAT SHEET

Basics

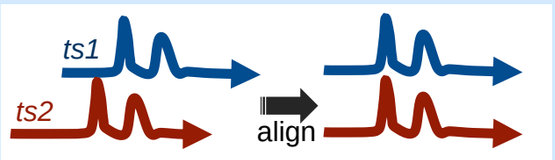
align is an R package for time-series data alignment.

There are multiple reasons why data may not be optimally aligned, e.g.:

- Data is from sensors at different points in a process-line.
- Data is from sensors with different response characteristics.
- Data logged with incorrect time stamp or insufficient buffering.

Sometimes smaller alignment issues can be worked around by reducing the time-series resolution, but often extra insights can be gained if you can work at the highest resolution available...

For two unaligned time-series, ***ts1*** and ***ts2***:

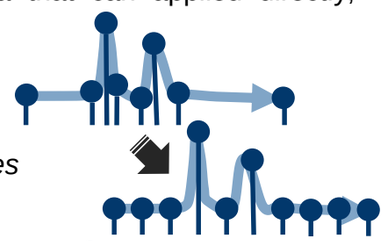


*There are multiple alignment methods, some better suited to particular applications, and **align** (hopefully) provides a simple coding structure for implementing, studying and comparing different alignment strategies.*

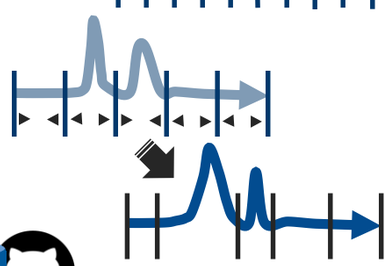
Miscellanea

Although the main **align** functions expect at least two time-series, some use sub-routines to reshape data that can applied directly, e.g.:

regularize()
to standardise irregular time-series



warp()
to redistribute time-series



Built in the open
and acknowledged gratefully

Coding Structure

The main **align** functions are named ***[type]_align()*** and intended for use with ***vectors*** and elements/columns in ***data.frames***, or object classes converted to either.

General alignment calls: **vectors** called directly

_align(ts1, ts2, …) with **data.frame**s (e.g. **df1** and **df2**) …

_align(df1, by=c("ts1", "ts2"), …)
_align(df1, df2, by=c("ts1"), …)
_align(df1, df2, by=c("ts1", "ts2"), …)

… **by** argument identifies columns to be aligned

Catching outputs: optional argument, **output**

return <- _align(…, output)

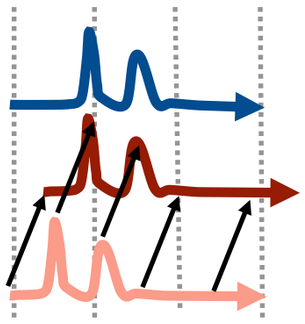
_align()s typically plot fit analysis, report summary to console and return aligned data as a **data.frame**

NOTE: ***_align()****s are applied to both **ts2** and its source **data.frame** if not also the **ts1** source, so they can align **data.frame**s with common (or similar) time-series…*

Output term	returns
"plot"	plot(alignment)
"summary"	summary(alignment)
"ans"	alignment product, data.frame
"alignment"	Generic _align() function output; alignment class object
c(…, …)	Multiple outputs; any of above; all run but ONLY last returned to catch

Linear

Linear alignment - *applying a fixed offset.*



*The simplest and least aggressive of the alignments, moves **ts2** (and **df2**) relative to **ts1** (and **df1**), without warping, and pads with NAs.*

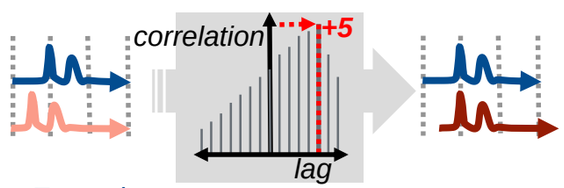
n_align()
Optional argument n (default n=0)

Example
n_align(…, n=5)

… like **cbind()** BUT does not need same or divisible number of columns

… moves ts2/df2 forward 5 rows relative to ts1/df1

cor_align()
*Like n_align() but it uses correlation between **ts1** and **ts2** to automatically assign **n***



Example
cor_align(…, min.overlap=20)

…optional argument, **min.overlap** sets the smallest **ts1**/**ts2** overlap, here 20 elements of vector or rows of data.frame

Non-linear

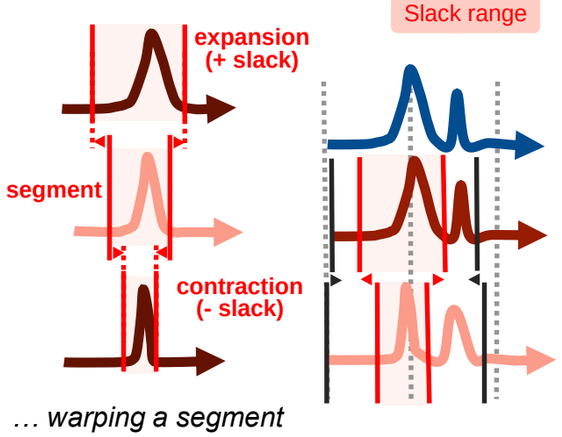
Non-linear alignment - *applying a variable offset based on optimising the **ts1** and **ts2** agreement.*

Arguably the most aggressive and least conservative of the alignment strategies

Example:
Correlation optimized warping (COW)

*… divides **ts2** into a series of subsets (called segments) and concertinas them using an expansion/contraction range (slack) to maximise correlation with **ts1***

segment width
cow_align(…, seg, slack)
Slack range



… warping a segment

*… and as part of segment series to align **ts2** with **ts1***

NOTE: *COW is not the only warping option, other examples include dynamic time warping*

Constrained

Constrained alignment - *applying an offset, either fixed or variable, based on an assumed relationship between **ts1** and **ts2** timings.*

*Often the preferred option if the nature of offsets are well understood and effects can be mapped from **ts1** onto **ts2***

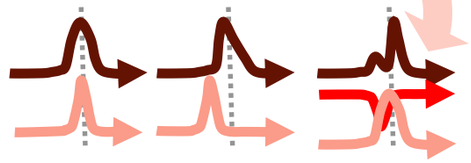
Example:
Fitted alignment

The time offset function for **ts2**/**df2**

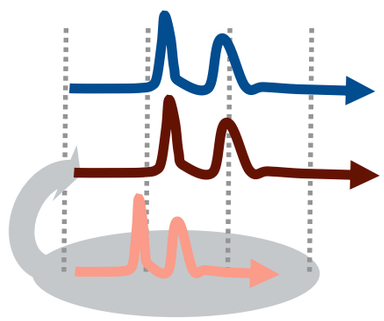
fit_align(…, fun, upper, lower)

the upper and lower limits for any constants in **fun** that require fitting

transform **ts2** using function based on **ts2** (or other time-series) and/or **diff(ts2)**, etc



*… function applied consistently across **ts2** to improve agreement with **ts1***



align: : CHEAT SHEET

Basics

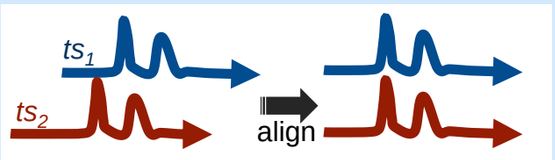
align is an R package for time-series data alignment.

There are multiple reasons why data may not be optimally aligned, e.g.:

- Data is from sensors at different points in a process-line.
- Data is from sensors (or of properties) with different response characteristics.
- Logging issues, incorrect time stamp or insufficient buffering.

Sometimes smaller alignment issues can be worked around by reducing the time-series resolution, but often extra insights can be gained if you can work at the highest resolution you have...

For two unaligned time-series, ***ts*$_1$** and ***ts*$_2$**:

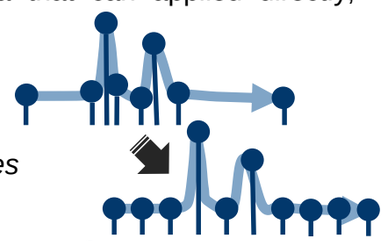


*There are multiple alignment methods, some better suited to particular applications, and **align** (hopefully) provides a simple coding structure for implementing, studying and comparing different alignment strategies.*

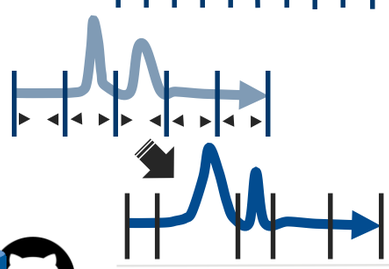
Miscellanea

Although the main **align** functions expect at least two time-series, some use sub-routines to reshape data that can applied directly, e.g.:

regularize()
to standardise irregular time-series



warp()
redistribute time-series



Built in the open
and acknowledged gratefully

Coding Structure

The main **align** functions are form ***[type]_align()*** and intended for use with ***vectors*** and elements/columns in ***data.frames***, or object classes converted to either.

General alignment calls: **vectors** called directly

_align(ts1, ts2, …)

with **data.frame**s (e.g. **df1** and **df2**) …

_align(df1, by=c("ts1", "ts2"), …)
_align(df1, df2, by=c("ts1"), …)
_align(df1, df2, by=c("ts1", "ts2"), …)

… **by** argument identifies columns to be aligned

Catching outputs: optional argument, **output**

return <- _align(…, output)

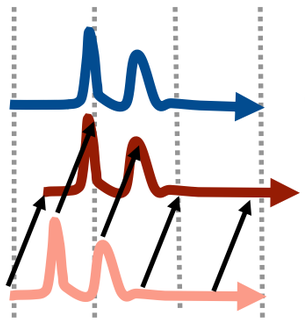
_align()s typically plot fit analysis, report summary to console and return aligned data as **data.frame**

NOTE: ***_align()****s are applied to both **ts2** and its source **data.frame** if not also the **ts1** source, so they can align **data.frame**s with common (or similar) time-series…*

Output term	returns
"plot"	plot(alignment)
"summary"	summary(alignment)
"ans"	alignment product, data.frame
"alignment"	Generic _align() function output; alignment class object
c(…, …)	Multiple outputs; any of above; all run but ONLY last returned to catch

Linear

Linear alignment - *applying a fixed offset.*



The simplest and least aggressive of the alignments, moves ts2 (and df2) relative to ts1 (and df1), without warping, and pads with NAs

n_align()

Optional argument n (default n=0)

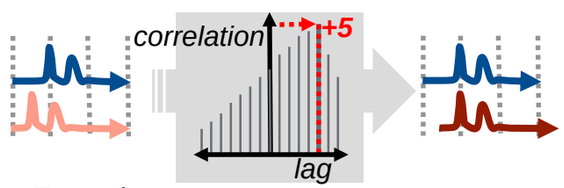
Example
n_align(…, n=5)

… like **cbind()** BUT does not need same or divisible number of columns

… moves ts2/df2 forward 5 rows relative to ts1/df1

cor_align()

*Like n_align() but it uses correlation between **ts1** and **ts2** to automatically assign **n***



Example
cor_align(…, min.overlap=20)

…optional argument, **min.overlap** sets the smallest ts1/ts2 overlap, here to 20 elements of vector or rows of data.frame

Non-linear

Non-linear alignment - *applying a variable offset based on optimising the ts1 and ts2 agreement.*

Arguably the most aggressive and least conservative of the alignment strategies

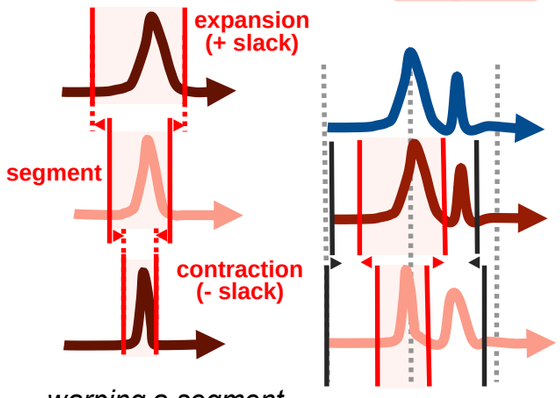
Example:
Correlation optimized warping (COW)

… divides ts2 into a series of subsets (called segments) and concertinas them using an expansion/contraction range (slack) to maximise correlate with ts1

segment width

cow_align(…, seg, slack)

Slack range



… warping a segment

*… and as part of segment series to align **ts2** with **ts1***

NOTE: *COW is not the only warping option, other examples include dynamic time warping*

Constrained

Constrained alignment - *applying an offset, either fixed or variable, based on an assumed relationship between ts1 and t2 timings.*

Often the preferred option if the nature of offsets are well understood and effects can be mapped from ts1 onto ts2

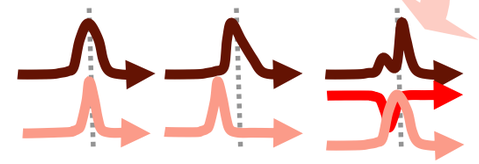
Example:
Fitted alignment

The time offset function for ts2/df2

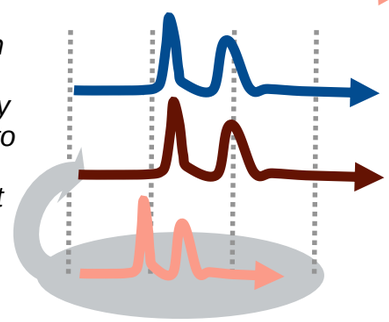
fit_align(…, fun, upper, lower)

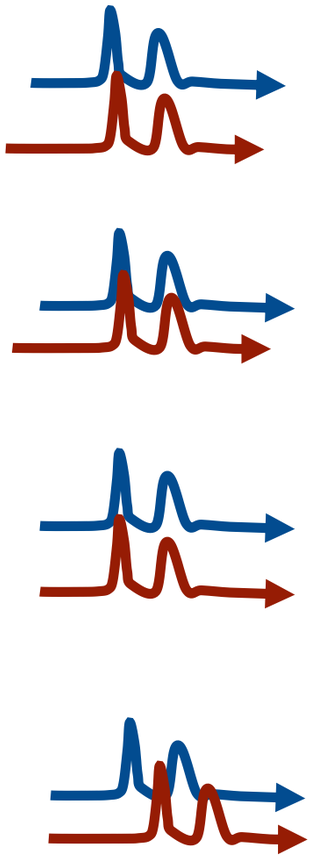
the upper and lower limits for any constants in **fun** that require fitting

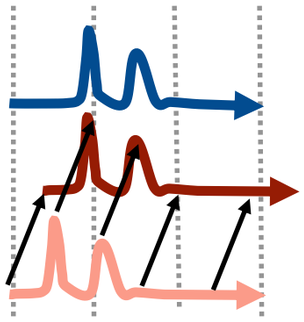
transform **ts2** using function based on **ts2** (or other non-aligned time-series) and/or **diff(ts2)**, etc

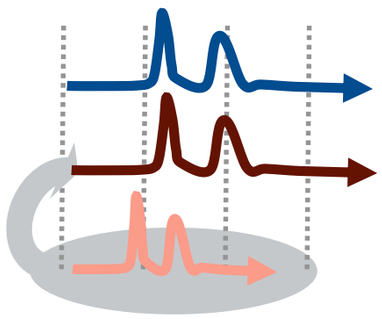


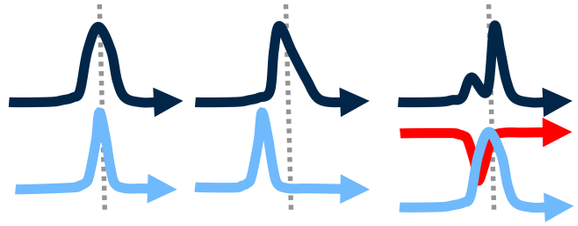
… function applied consistently across ts2 to improve agreement with ts1

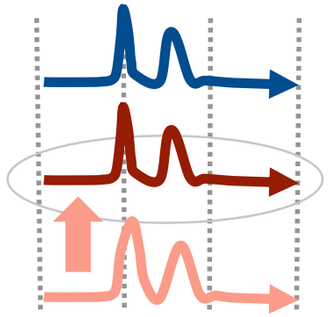


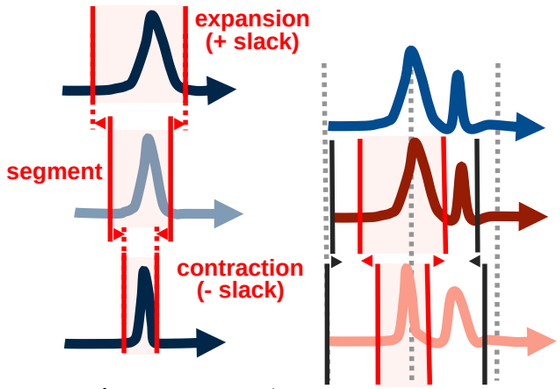






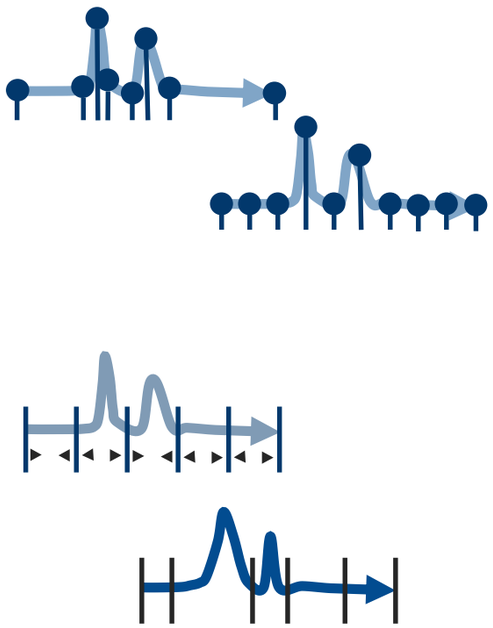


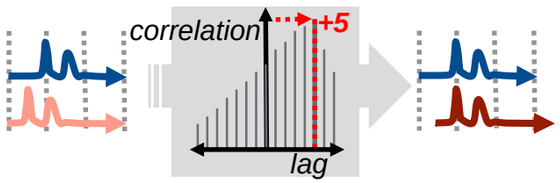


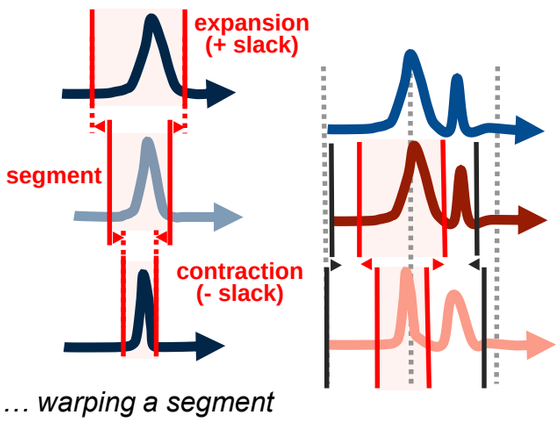


… warping a segment

*… and as part of segment series to align **ts2** with **ts1***





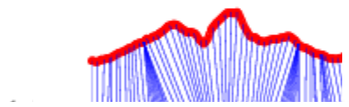


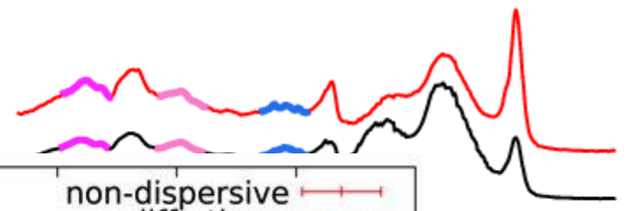
… warping a segment

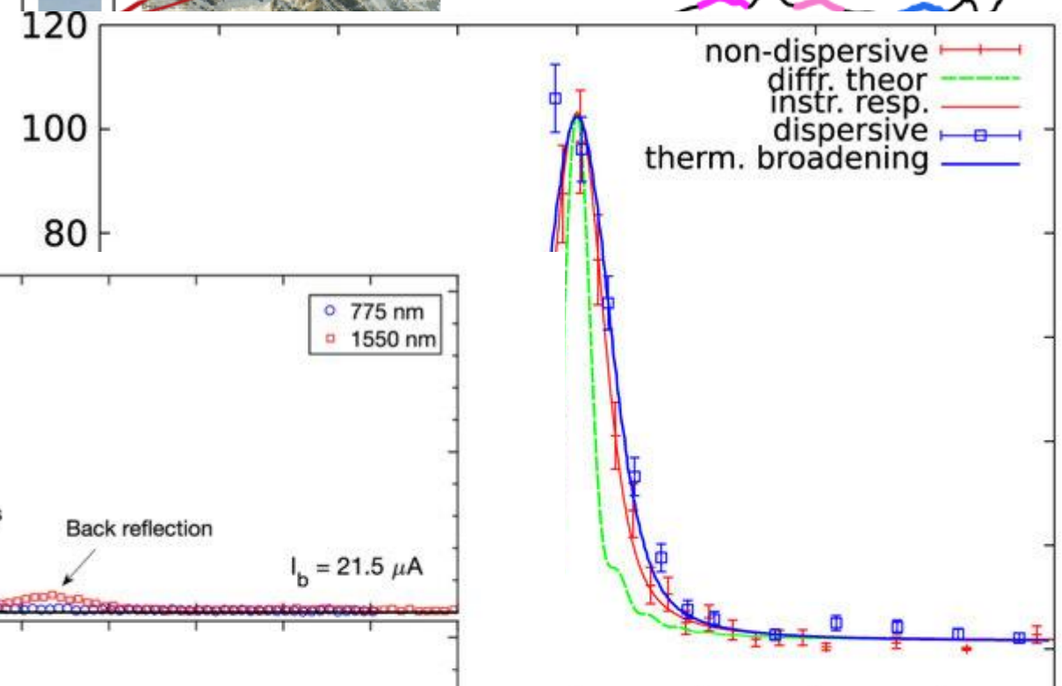
*… and as part of segment series to align **ts2** with **ts1***

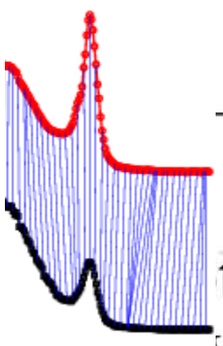

align
https://karlropkins.github.io/align/

Raw Data



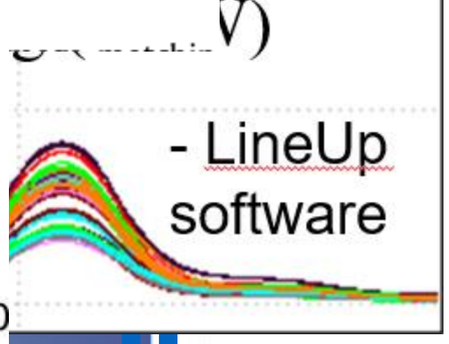






timized

V)



- LineUp software

44181 (eV)