



IMPORTING AND MANAGING FINANCIAL DATA IN R

Making irregular data regular

Regular date-time sequences

- Time observations are same distance apart
- Create regular date-time sequences using `seq()` methods:
 - `seq.Date()`
 - `seq.POSIXt()` (`POSIXct` and `POSIXlt`)

```
> from_date <- as.Date("2017-01-01")
> to_date <- as.Date("2017-01-03")
> date_seq <- seq(from = from_date,
                  to = to_date,
                  by = "day")
```

start() and end() functions

- start() first index value
- end() last index value

```
> regular_xts <- xts(seq_along(date_seq), order.by = date_seq)
> start(regular_xts)
[1] "2017-01-01"

> end(regular_xts)
[1] "2017-01-03"

> seq(from = start(regular_xts),
      to = end(regular_xts),
      by = "day")
[1] "2017-01-01" "2017-01-02" "2017-01-03"
```

Zero-width xts objects

- xts object with an index, no data

```
> zero_width_xts <- xts(, order.by = date_seq)
> zero_width_xts
Data:
numeric(0)
```

Index:

```
  Date[1:3], format: "2017-01-01" "2017-01-02" "2017-01-03"

> str(zero_width_xts)
An 'xts' object of zero-width
```

Creating regular from irregular data

- Add observation at each date-time in regular sequence
- NA in the result

Merge irregular xts with regular zero-width xts (1)

```
> irregular
      Price
2017-01-02 20.01
2017-01-04 20.02
2017-01-10 20.05
```

```
> date_seq <- seq(from = start(irregular),
                  to = end(irregular),
                  by = "day")

> regular_xts <- xts(, date_seq)
```

Merge irregular xts with regular zero-width xts (2)

```
> merge(irregular, regular_xts)
      Price
2017-01-02 20.01
2017-01-03    NA
2017-01-04 20.02
2017-01-05    NA
2017-01-06    NA
2017-01-07    NA
2017-01-08    NA
2017-01-09    NA
2017-01-10 20.05
```

Filling missing values

```
> merged_xts <- merge(irregular, regular_xts)
> na.locf(merged_xts)
```

	Price
2017-01-02	20.01
2017-01-03	20.01
2017-01-04	20.02
2017-01-05	20.02
2017-01-06	20.02
2017-01-07	20.02
2017-01-08	20.02
2017-01-09	20.02
2017-01-10	20.05

Filling missing values

```
> merge(irregular, regular_xts, fill = na.locf)
      Price
2017-01-02 20.01
2017-01-03 20.01
2017-01-04 20.02
2017-01-05 20.02
2017-01-06 20.02
2017-01-07 20.02
2017-01-08 20.02
2017-01-09 20.02
2017-01-10 20.05
```



IMPORTING AND MANAGING FINANCIAL DATA IN R

Let's practice!



IMPORTING AND MANAGING FINANCIAL DATA IN R

Aggregating to lower frequency

Low frequency data

- Timestamps have too much resolution
- Represent the first quarter of 2017
 - "2017-01-01" (first)
 - "2017-03-31" (last)
 - "2017-02-01" (middle)

Example

- Compare the *daily* 10-year Treasury constant maturity rate with USA Gross Domestic Product (*quarterly*)
- FRED symbols:
 - DGS10
 - GDP

Merge aggregated data with low-frequency data

```
> # Aggregate to quarterly  
> QGS10 <- apply.quarterly(DGS10, median, na.rm = TRUE)  
  
> # Merge quarterly aggregate with quarterly GDP  
> QGS10_GDP <- merge(QGS10, GDP)  
> QGS10_GDP
```

	DGS10	GDP
2015-01-01	NA	17783.6
2015-03-31	1.97	NA
2015-04-01	NA	17998.3
2015-06-30	2.19	NA
2015-07-01	NA	18141.9
2015-09-30	2.20	NA
2015-10-01	NA	18222.8
2015-12-31	2.23	NA

Low frequency date-time classes

- `yearmon()` for monthly data
- `yearqtr()` for quarterly data

```
> as.Date("2017-01-01")  
[1] "2017-01-01"  
>  
> as.yearmon("2017-01-01")  
[1] "Jan 2017"  
>  
> as.yearqtr("2017-01-01")  
[1] "2017 Q1"
```

Convert index to lowest frequency

```
> # Convert both indexes to yearqtr  
> index(QGS10) <- as.yearqtr(index(QGS10))  
> index(GDP) <- as.yearqtr(index(GDP))
```

```
> # Merging 'just works'  
> merge(QGS10, GDP)  
      DGS10      GDP  
2015 Q1  1.97 17783.6  
2015 Q2  2.19 17998.3  
2015 Q3  2.20 18141.9  
2015 Q4  2.23 18222.8
```


Align with beginning-of-period timestamp

```
> # Last observation carried backward  
> QGS10_GDP_locb <- na.locf(QGS10_GDP, fromLast = TRUE)
```

```
> # Subset by beginning-of-period index  
> QGS10_GDP_first_period <- QGS10_GDP_locb[index(GDP)]  
> QGS10_GDP_first_period
```

	DGS10	GDP
2015-01-01	1.97	17783.6
2015-04-01	2.19	17998.3
2015-07-01	2.20	18141.9
2015-10-01	2.23	18222.8



IMPORTING AND MANAGING FINANCIAL DATA IN R

Let's practice!



IMPORTING AND MANAGING FINANCIAL DATA IN R

Aggregating and combining intraday data

Timezones!



Timezones!

- Internally, xts index is seconds since midnight 1970-01-01 in UTC
- `merge()` uses internal index
- `merge()` result will have timezone of the first object

Timezones!

```
> datetime <- as.POSIXct("2017-01-18 10:00:00", tz = "UTC")  
> london <- xts(1, datetime, tzone = "Europe/London")  
> tokyo <- xts(1, datetime, tzone = "Asia/Tokyo")
```

```
> merge(london, tokyo)  
              london tokyo  
2017-01-18 10:00:00      1      1  
  
> merge(tokyo, london)  
              tokyo london  
2017-01-18 19:00:00      1      1
```

Creating regular intraday data

```
> head(dc_trades)
                Price
2016-01-16 08:00:58 20.85
2016-01-16 08:01:56 20.85
2016-01-16 08:03:35 20.85
2016-01-16 08:07:44 20.84
2016-01-16 08:45:58 20.85
2016-01-16 08:46:49 20.85
```

Creating regular intraday data

```
> datetimes <- seq(from = as.POSIXct("2016-01-16 08:00"),  
  to = as.POSIXct("2016-01-17 18:00"),  
  by = "1 min")
```

```
> regular_xts <- xts(, order.by = datetimes)
```

```
> merged_xts <- merge(dc_trades, regular_xts)
```

```
> head(merged_xts)
```

	Price
2016-01-16 08:00:00	NA
2016-01-16 08:00:58	20.85
2016-01-16 08:01:00	NA
2016-01-16 08:01:56	20.85
2016-01-16 08:02:00	NA
2016-01-16 08:03:00	NA

Subset to trading hours

```
> # All observations should be NA
> all(is.na(merged_xts["2016-01-16 19:00/2016-01-17 07:00"]))
[1] TRUE

> # xts time-of-day subsetting
> merged_trade_day <- merged_xts["T08:00/T18:00"]

> # Now there are no observations
> nrow(merged_trade_day["2016-01-16 19:00/2016-01-17 07:00"])
[1] 0
```

Fill missing values by trading day

- split-lapply-rbind paradigm from Introduction to xts and zoo

```
> # split() data into list of non-overlapping chunks
> trade_day_list <- split(merged_trade_day, "days")
>
> # lapply() a function to each chunk (list element)
> filled_trade_day_list <- lapply(trade_day_list, na.locf)
>
> # Combine list of chunks using do.call() and rbind()
> filled_trade_day <- do.call(rbind, filled_trade_day_list)
```

Aggregate irregular intraday data

- Aggregate dense intraday data with `to.period()`
- `period`: new periodicity (e.g. seconds, hours, days, etc)
- `k`: number of periods per new observation

Aggregate irregular intraday data (1)

```
> head(dc_price)
              DC.Price
2016-01-16 00:00:07 20.84224
2016-01-16 00:00:08 20.84225
2016-01-16 00:00:08 20.84225
2016-01-16 00:00:11 20.84225
2016-01-16 00:00:25 20.84224
2016-01-16 00:00:44 20.84224
```

Aggregate irregular intraday data (2)

```
> xts_5min <- to.period(dc_price, period = "minutes", k = 5)
head(xts_5min, n = 4)
```

		dc_price.Open	dc_price.High	dc_price.Low	dc_price.Close
2016-01-16	00:03:49	20.84224	20.84227	20.84140	20.84160
2016-01-16	00:09:50	20.84160	20.84160	20.84156	20.84156
2016-01-16	00:14:57	20.84156	20.84156	20.84154	20.84154
2016-01-16	00:19:23	20.84154	20.84154	20.83206	20.83211

```
xts_aligned <- align.time(xts_5min, n = 60 * 5)
head(xts_aligned, n = 4)
```

		dc_price.Open	dc_price.High	dc_price.Low	dc_price.Close
2016-01-16	00:05:00	20.84224	20.84227	20.84140	20.84160
2016-01-16	00:05:00	20.84160	20.84160	20.84156	20.84156
2016-01-16	00:15:00	20.84156	20.84156	20.84154	20.84154
2016-01-16	00:20:00	20.84154	20.84154	20.83206	20.83211



IMPORTING AND MANAGING FINANCIAL DATA IN R

Let's practice!