

Time series attributes

MANIPULATING TIME SERIES DATA IN R



Harrison Brown

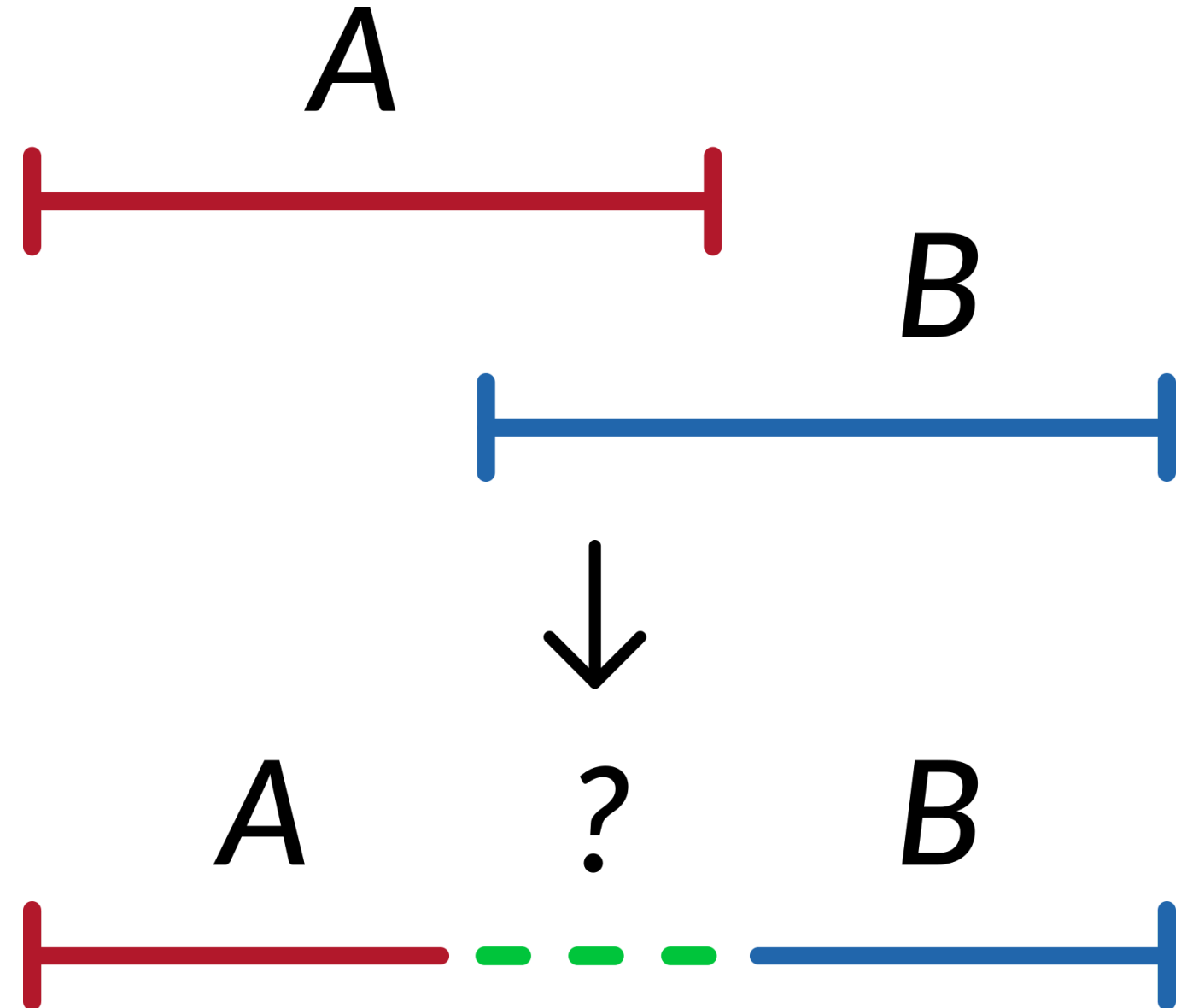
Graduate Researcher in Geography

Temporal attributes

- Start point
- End point
- Frequency

Temporal attributes can be used to:

- Understand when and for how long the data was sampled
- Find the times where multiple time series overlap



Temporal attributes

Key temporal attributes:

- Start point

```
start(AirPassengers)
```

```
[1] 1949 1
```

Temporal attributes

Key temporal attributes:

- Start point
- End point

```
end(ftse)
```

```
[1] 1998.646
```

- Decimal date (`numeric`)

```
end(ftse) %>%  
  lubridate::date_decimal()
```

```
[1] "1998-08-24 20:18:27 UTC"
```

Temporal attributes

Key temporal attributes:

- Starting point
- Endpoint
- **Frequency**

```
frequency(monthly_data)
```

```
[1] 12
```

```
frequency(weekly_data)
```

```
[1] 52.14286
```

```
frequency(ftse)
```

```
[1] 260
```

```
ftse
```

```
1991.4962 2443.6  
1991.5    2460.2  
1991.5038 2448.2  
1991.5077 2470.4  
1991.5115 2484.7  
1991.5154 2466.8  
...
```

Regular vs. irregular time series

Regular



- Evenly-spaced intervals
- No missing values
- Uses decimal date for 'irregular' intervals

Irregular



- Spacing can be irregular
 - Weekdays, random days
- Missing observations
- Decimal date or `Date` / `POSIXct` data

Let's practice!

MANIPULATING TIME SERIES DATA IN R

Time series objects with the zoo package

MANIPULATING TIME SERIES DATA IN R



Harrison Brown

Graduate Researcher in Geography

The zoo package

`zoo`

- Provides `zoo` class of objects
- Functions for manipulating and visualizing time series data

zoo versus base R

base R:

- `stats::ts` class
- **Only** regularly-spaced intervals
- Can be tedious to create data frames

`zoo` :

- `zoo` class
- Regular or irregular intervals
- Methods for converting to data frame

Creating a zoo object

```
sample_values
```

```
[1] 4.224 5.495 6.156  
[4] 6.397 6.291 6.262  
...
```

```
sample_dates
```

```
[1] "2022-01-20" "2022-01-21"  
[3] "2022-01-22" "2022-01-23"  
[5] "2022-01-24" "2022-01-25"  
...
```

Creating a zoo object

```
library(zoo)
my_zoo <- zoo(x = sample_values,
              order.by = sample_dates)

my_zoo
```

```
2022-01-20 4.224
2022-01-21 5.495
2022-01-22 6.156
2022-01-23 6.397
2022-01-24 6.291
2022-01-25 6.262
...
```

Converting to zoo from ts

```
class(AirPassengers)
```

```
[1] "ts"
```

```
AP_zoo <- zoo::as.zoo(AirPassengers)  
class(AP_zoo)
```

```
[1] "zoo"
```

zoo versus ts

```
print(AirPassengers)
```

```
      Jan Feb Mar Apr May Jun Jul ...  
1949 112 118 132 129 121 135 148 ...  
1950 115 126 141 135 125 149 170 ...  
1951 145 150 178 163 172 178 199 ...  
1952 171 180 193 181 183 218 230 ...  
1953 196 196 236 235 229 243 264 ...  
...
```

```
print(AP_zoo)
```

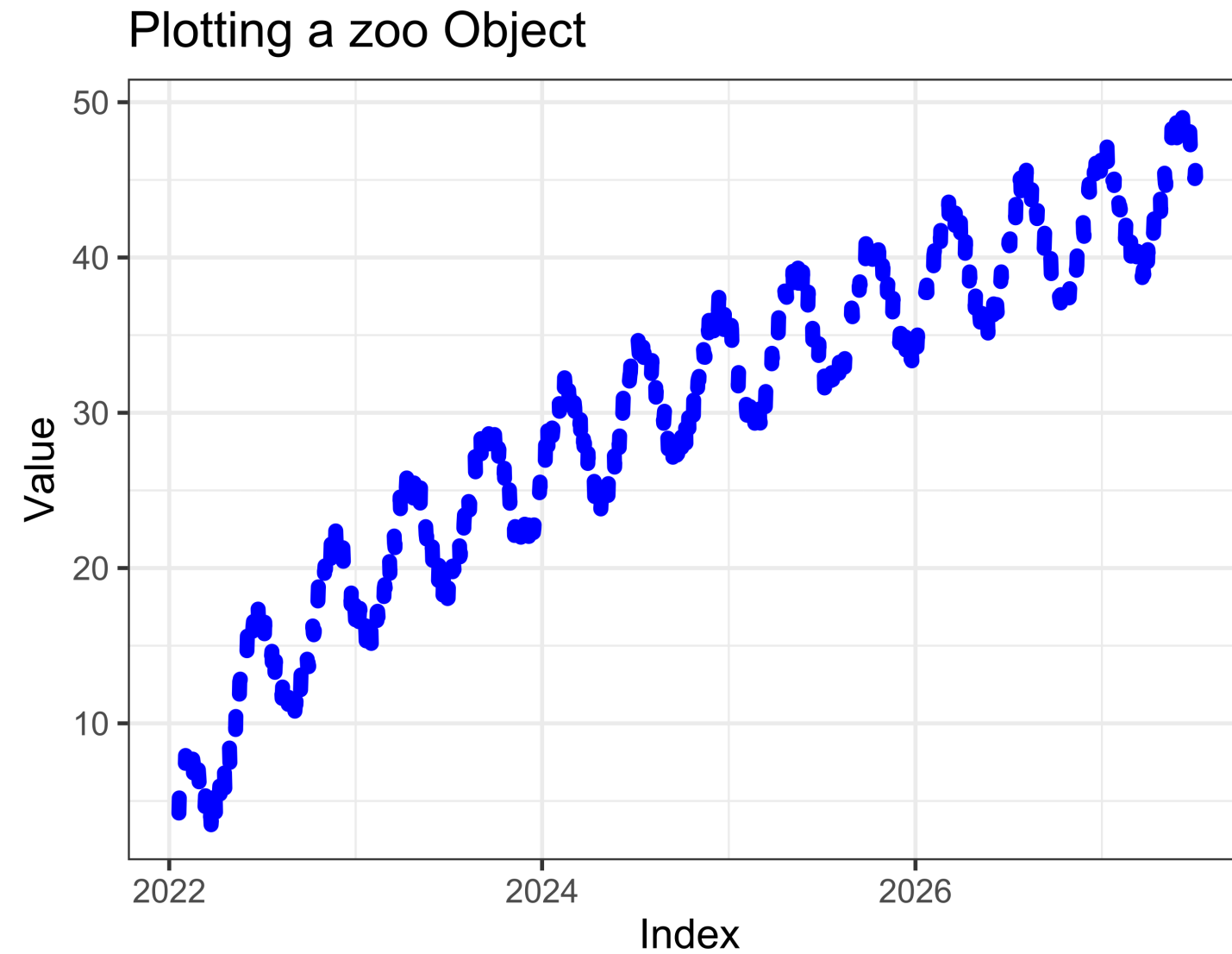
```
Jan 1949 112  
Feb 1949 118  
Mar 1949 132  
Apr 1949 129  
May 1949 121  
Jun 1949 135  
Jul 1949 148  
Aug 1949 148  
Sep 1949 136  
...
```

zoo and ggplot2

```
library(ggplot2)

ggplot(data = my_zoo,
       aes(x = Index, y = my_zoo)) +
  geom_line(color = "blue",
           size = 2,
           linetype = "dotted") +
  scale_y_continuous() +
  theme_bw() +
  labs(x = "Index",
       y = "Value",
       title = "Plotting a zoo Object")
```

zoo and ggplot2



Let's practice!

MANIPULATING TIME SERIES DATA IN R

Manipulating zoo objects

MANIPULATING TIME SERIES DATA IN R



Harrison Brown

Graduate Researcher in Geography

Unordered observations

Key time series attributes:

- Index
 - Order of observations
 - Allows time series to be sorted and compared
 - Most often defined by date or time

unordered_dates

```
[1] "1985-02-15" "1977-03-01"  
[3] "1970-06-10" "1971-01-16"  
[5] "1973-07-28" "1976-11-24"  
...
```

unordered_values

```
[1] 10.57  9.64  8.47  
[4] 9.07   9.16  9.39
```

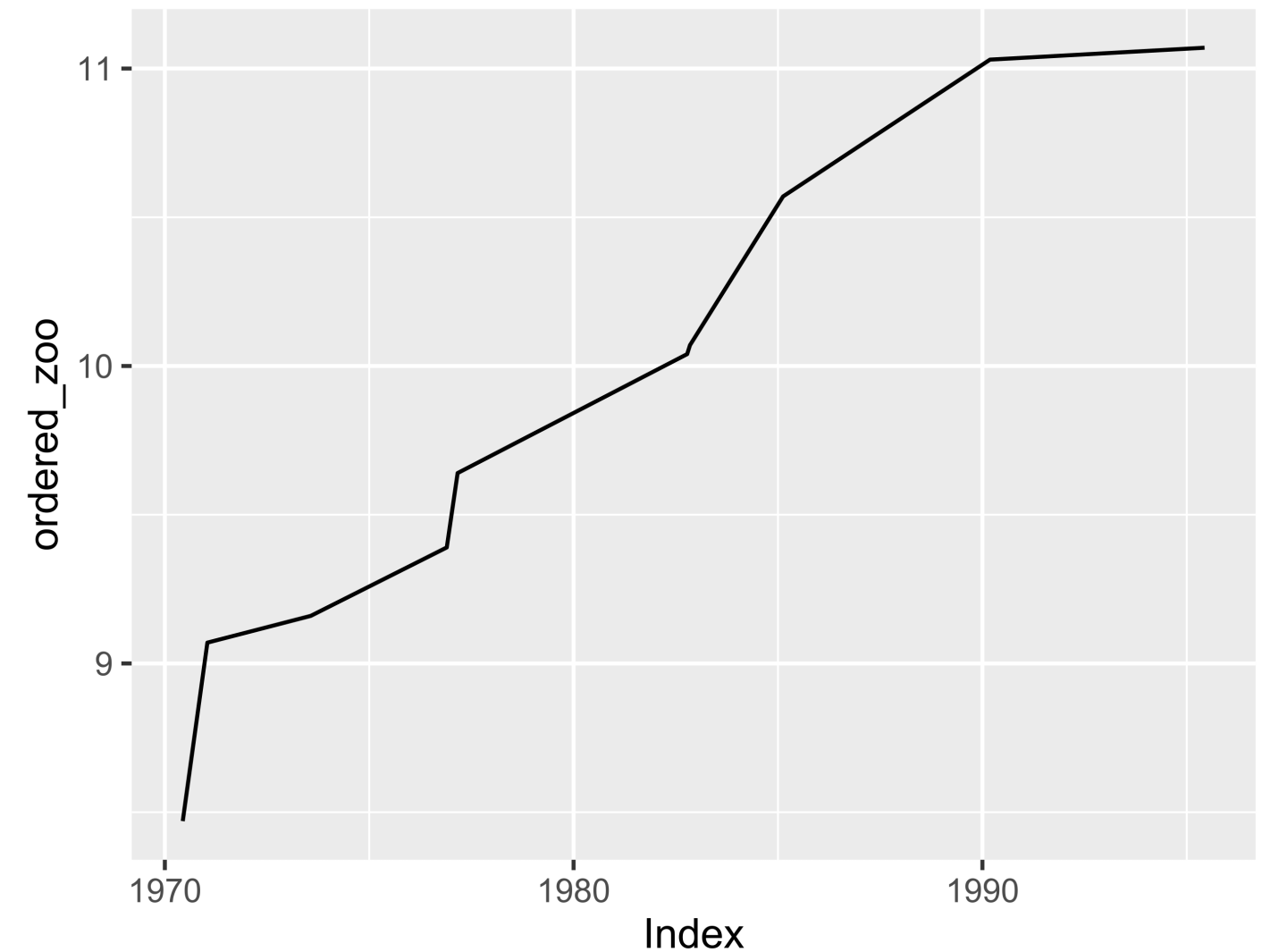
Unordered observations

```
ordered_zoo <-  
  zoo(unordered_values,  
      order.by = unordered_dates)
```

```
head(ordered_zoo)
```

```
1970-06-10  8.47  
1971-01-16  9.07  
1973-07-28  9.16  
1976-11-24  9.39  
1977-03-01  9.64  
1982-10-12 10.04
```

```
autoplot(ordered_zoo)
```



Index

```
zoo::index()
```

```
index(my_zoo)
```

```
[1] "2022-01-20" "2022-01-21"  
[3] "2022-01-22" "2022-01-23"  
[5] "2022-01-24" "2022-01-25"  
...
```

```
index(my_zoo) <- new_index
```

```
head(my_zoo)
```

```
2010-01-20 4.224  
2010-01-21 5.645  
2010-01-22 5.919  
2010-01-23 6.191  
2010-01-24 6.871  
2010-01-25 6.606
```

Core data

```
zoo::coredata()
```

```
coredata(my_zoo)
```

```
[1] 4.224 5.645 5.919  
[4] 6.191 6.871 6.606  
[7] 5.496 6.073 7.054  
[10] 6.849 7.291 7.064  
...
```

```
coredata(my_zoo)[1] <- 30
```

```
head(my_zoo)
```

```
2022-01-20 30.000  
2022-01-21 5.645  
2022-01-22 5.919  
2022-01-23 6.191  
2022-01-24 6.871  
2022-01-25 6.606
```

Overlapping time series

first_zoo

```
2022-08-01  4.06
2022-08-02 10.45
2022-08-03 15.43 *
2022-08-04 15.17 *
2022-08-05 19.87 *
2022-08-06 20.50 *
```

second_zoo

```
2022-08-03 15.43 *
2022-08-04 15.17 *
2022-08-05 19.87 *
2022-08-06 20.50 *
2022-08-07 18.56
2022-08-08 16.37
```

Overlapping time series

```
c(first_zoo, second_zoo)
```

```
Error in rbind.zoo(...) :  
  indexes overlap
```

```
index(first_zoo) %in% index(second_zoo)
```

```
[1] FALSE FALSE TRUE TRUE TRUE TRUE
```

```
!index(first_zoo) %in%  
  index(second_zoo)
```

```
[1] TRUE TRUE FALSE FALSE FALSE FALSE
```

Overlapping time series

```
subset <- !index(first_zoo) %in%  
          index(second_zoo)
```

```
c(first_zoo[subset], second_zoo)
```

```
2022-08-01  4.06  
2022-08-02 10.45  
2022-08-03 15.43  
2022-08-04 15.17  
2022-08-05 19.87  
2022-08-06 20.50  
2022-08-07 18.56  
2022-08-08 16.37
```

Let's practice!

MANIPULATING TIME SERIES DATA IN R

Converting between zoo and data frame

MANIPULATING TIME SERIES DATA IN R



Harrison Brown

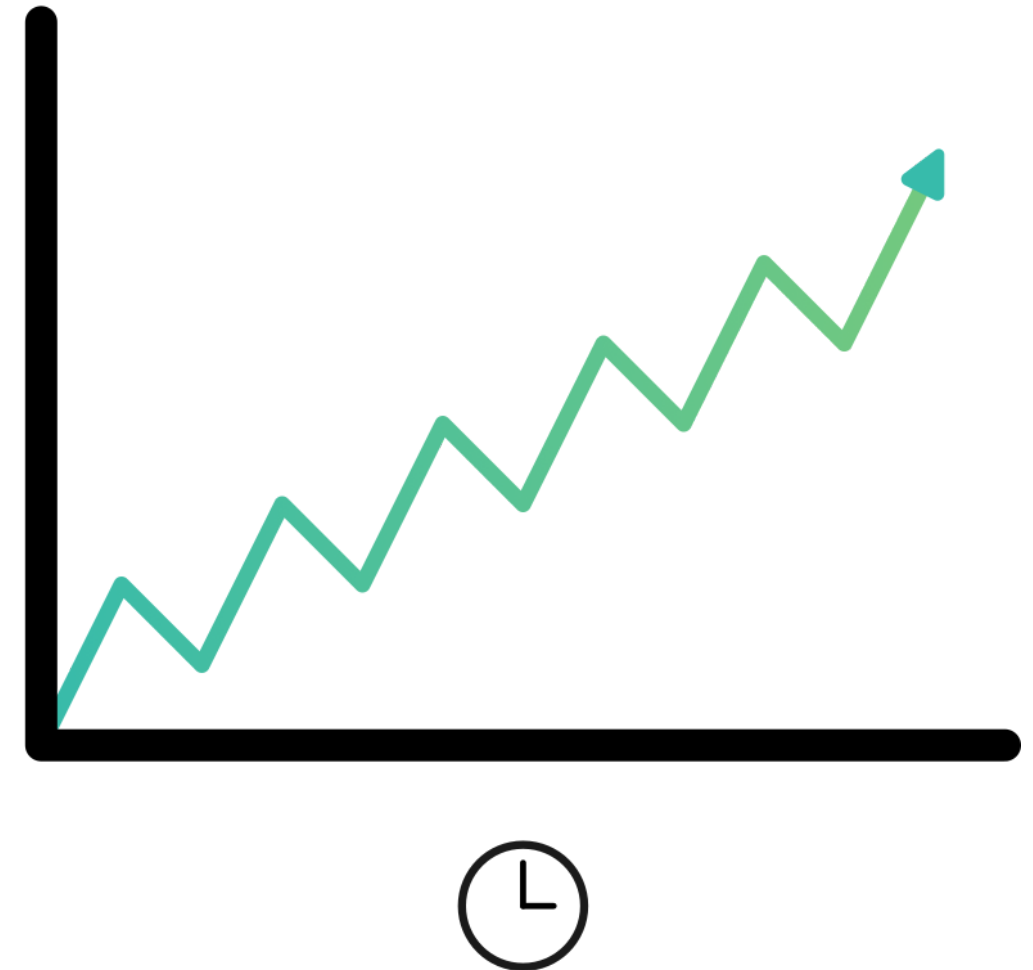
Graduate Researcher in Geography

Converting between data frame and zoo objects

	<i>X</i>
	
	
	
	
	



X



data.frame versus zoo

`data.frame` :

- Each variable stored as a column
- Index stored as column
 - e.g.: `data$dates`
- `geom_*()` only:

```
ggplot(example_dataframe,  
       aes(x = dates, y = values)) +  
  geom_line()
```

`zoo` :

- Each variable stored as matrix column
- Index stored as attribute
 - e.g.: `index(data)`
- `geom_*()` and `autoplot()` :

```
ggplot(maunaloa,  
       aes(x = Index, y = maunaloa)) +  
  geom_line()  
# Or...  
autoplot(maunaloa)
```

Data frame to zoo

```
head(prices)
```

	date	value
1	2013-01-01	4.88
2	2013-01-02	5.03
3	2013-01-03	5.11
4	2013-01-04	4.77
5	2013-01-05	5.04
6	2013-01-06	5.05

```
my_dataframe$my_column
```

```
prices_zoo <-  
  zoo(x = prices$value,  
      order.by = prices$date)  
prices_zoo
```

```
2013-01-01 4.88  
2013-01-02 5.03  
2013-01-03 5.11  
2013-01-04 4.77  
2013-01-05 5.04  
2013-01-06 5.05
```

Fortifying a zoo object

```
data_df <-  
  data.frame(index(my_zoo),  
             coredata(my_zoo))
```

```
zoo::fortify.zoo()
```

```
fortify.zoo(my_zoo)
```

	Index	var1	var2	var3	var4
1	1	1628.75	1678.1	1772.8	2443.6
2	2	1613.63	1688.5	1750.5	2460.2
3	3	1606.51	1678.6	1718.0	2448.2
4	4	1621.04	1684.1	1708.1	2470.4
5	5	1618.16	1686.6	1723.1	2484.7
6	6	1610.61	1671.6	1714.3	2466.8
...					

Let's practice!

MANIPULATING TIME SERIES DATA IN R