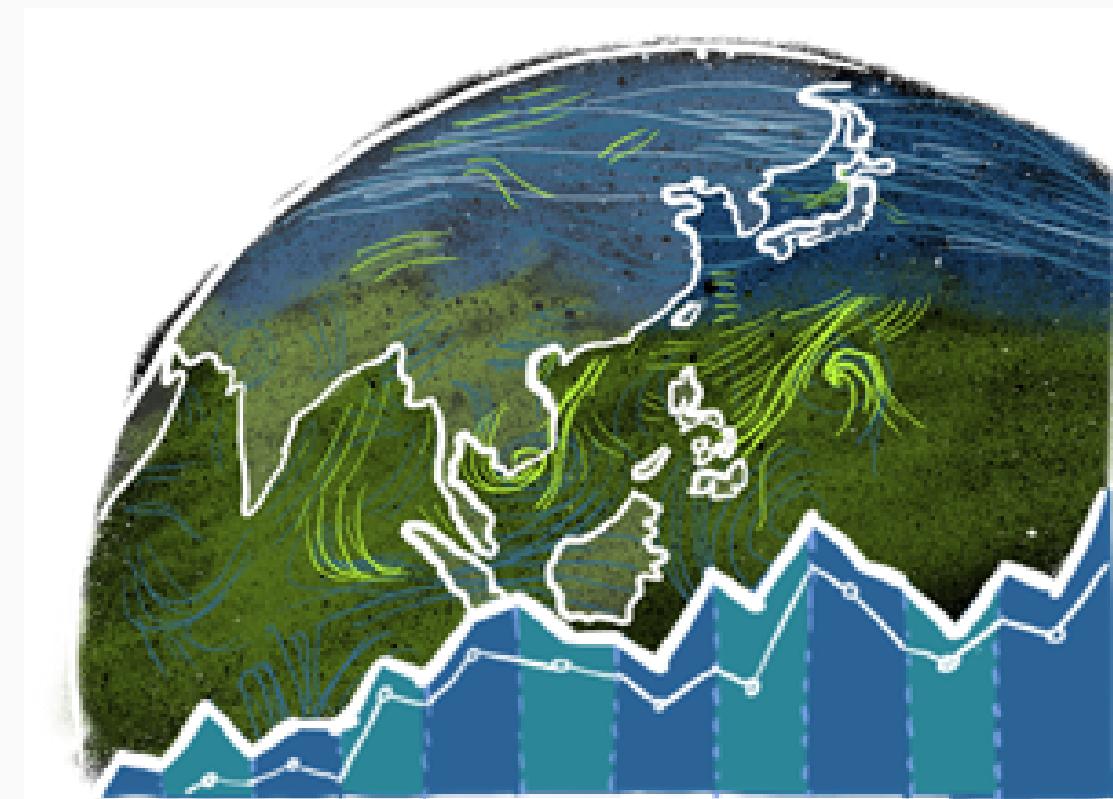


Daniel Vaulot

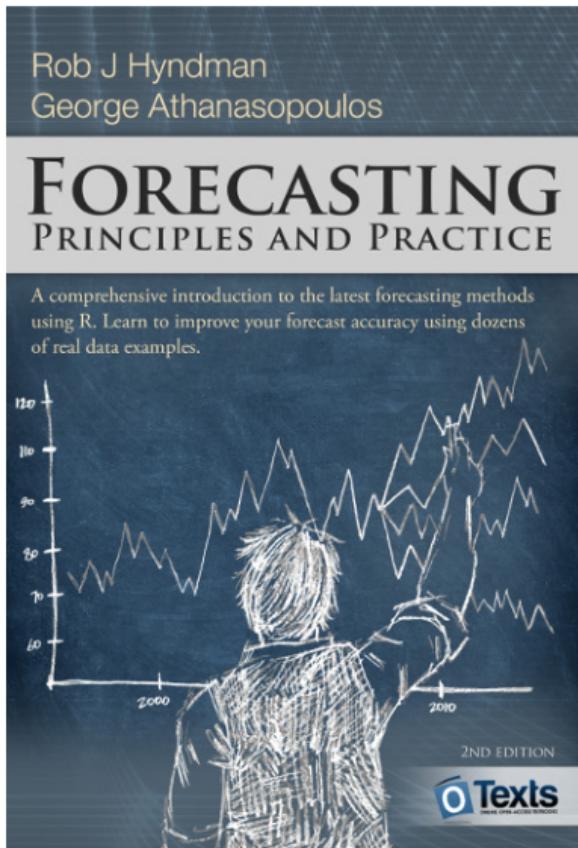
2020-04-10



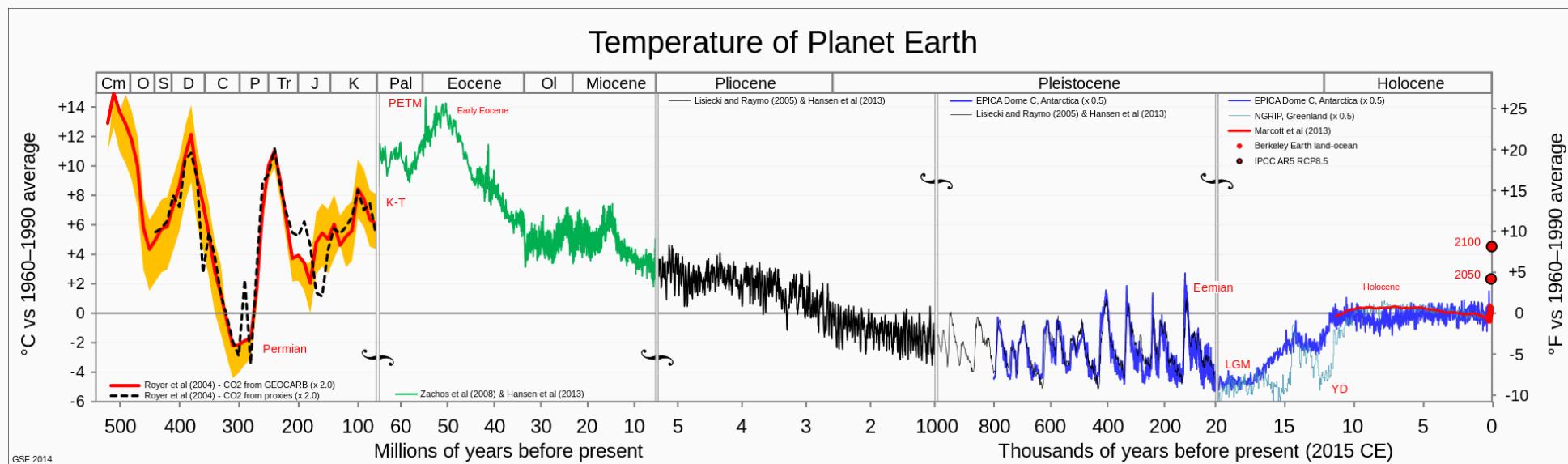
CNRS • SORBONNE UNIVERSITÉ
Station Biologique
de Roscoff

- Time series in earth sciences
- Phytoplankton time series
- Manipulating dates: lubridate package
- Time series objects: tsibble and feasts packages
- Other time series analysis strategies

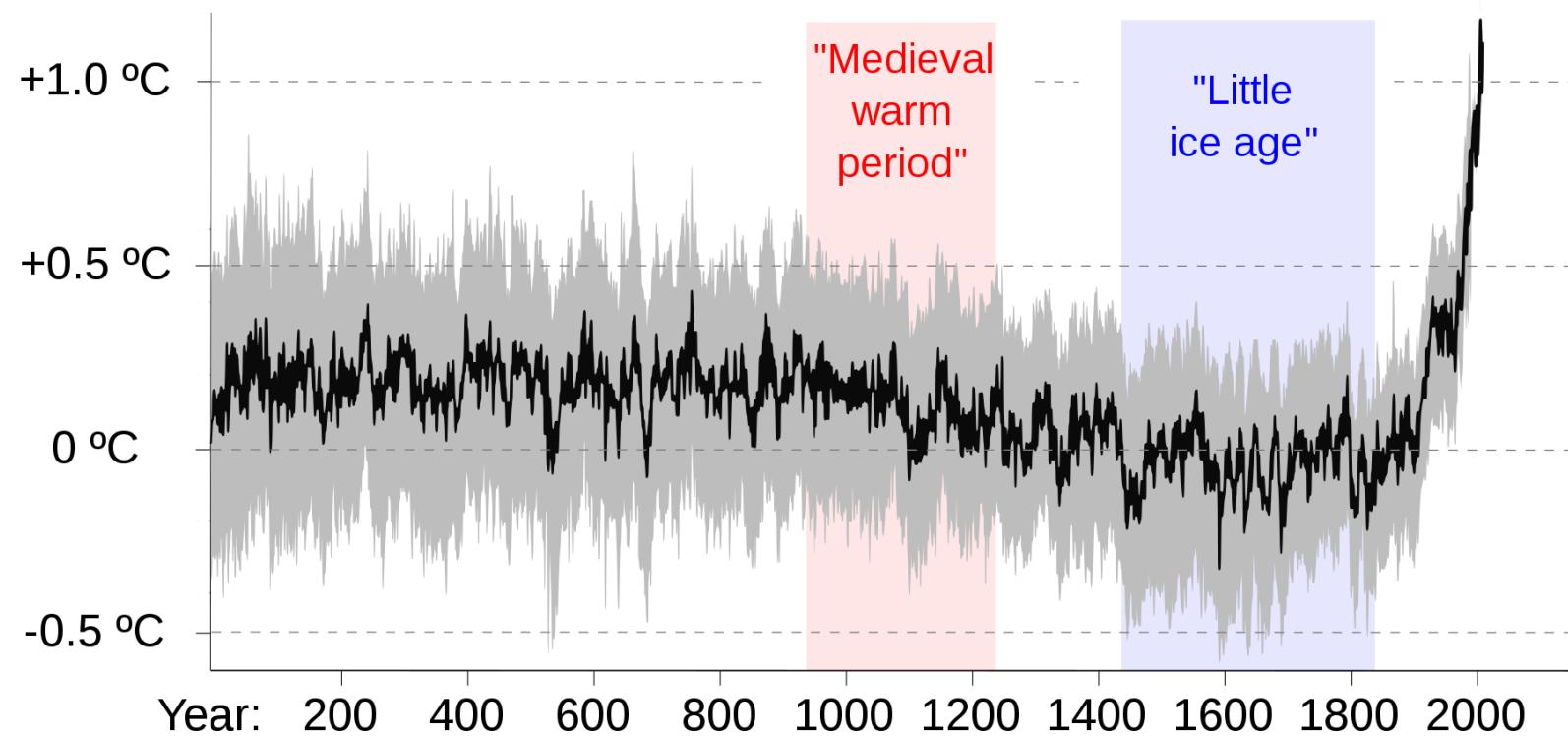
- lubridate
- tsibble
- feasts
- tsibbledata
- slider
- fable
- zoo
- WaveletComp
- ggplot2
- stringr
- dplyr



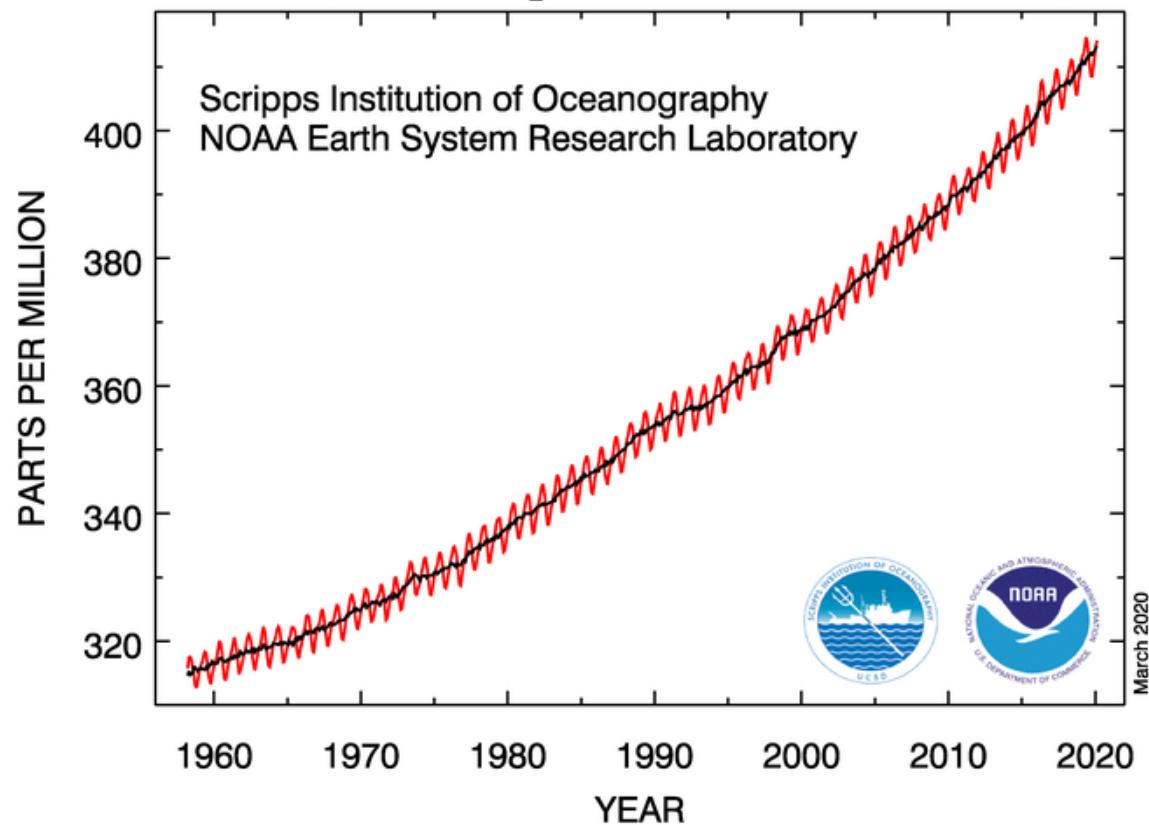
- Forecasting in practice Chapters 1-4:
<https://otexts.com/fpp3/>
- Hunter-Cevera et al. 2016. Physiological and ecological drivers of early spring blooms of a coastal phytoplankton. *Science* 354:326–329.

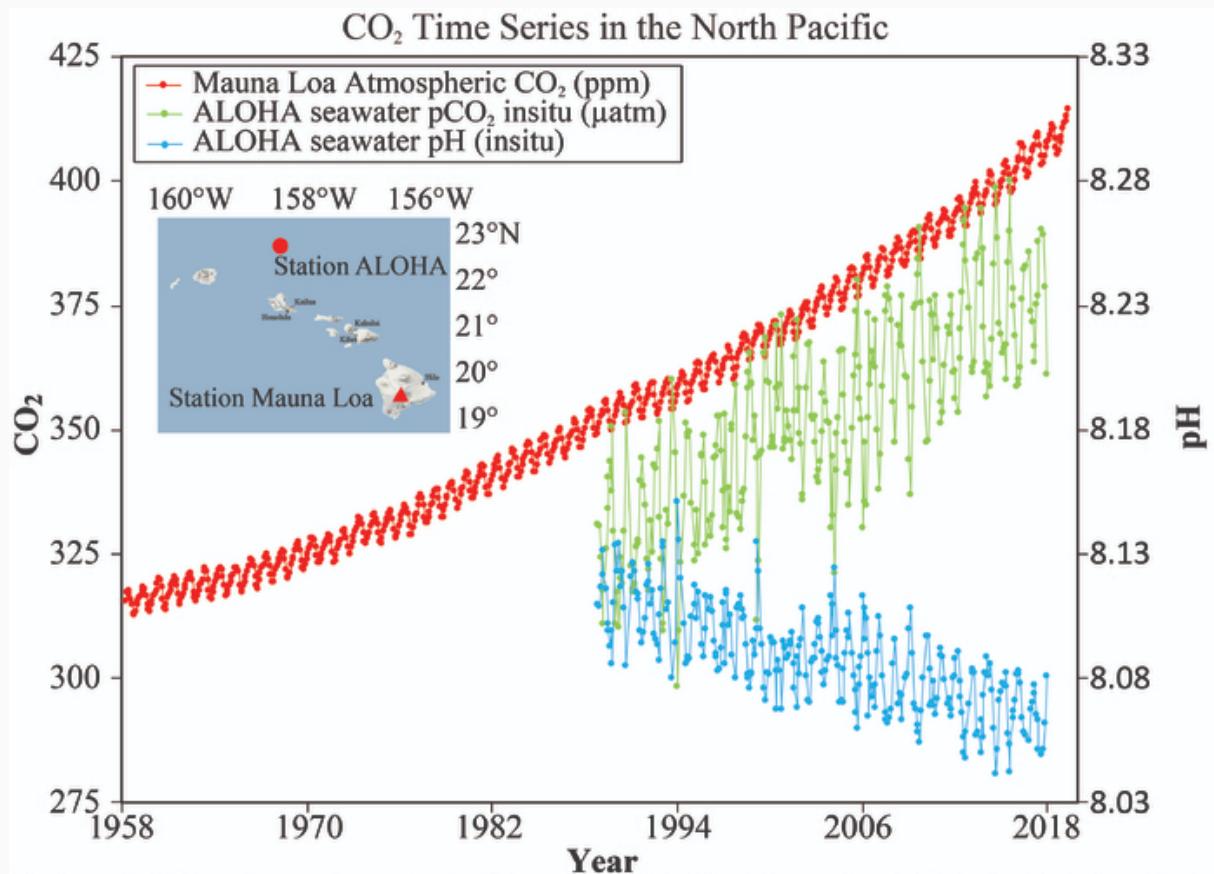


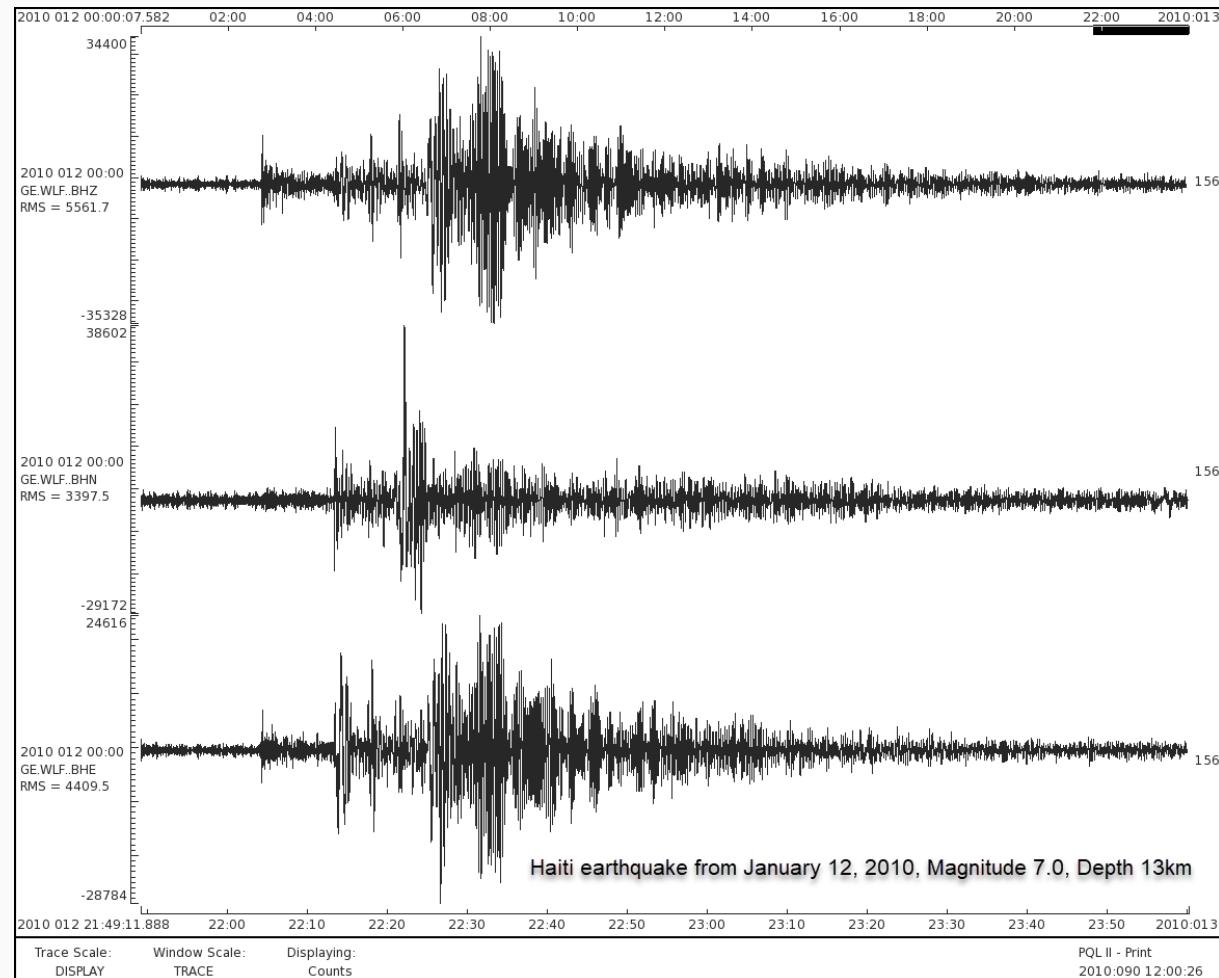
Global Average Temperature Change

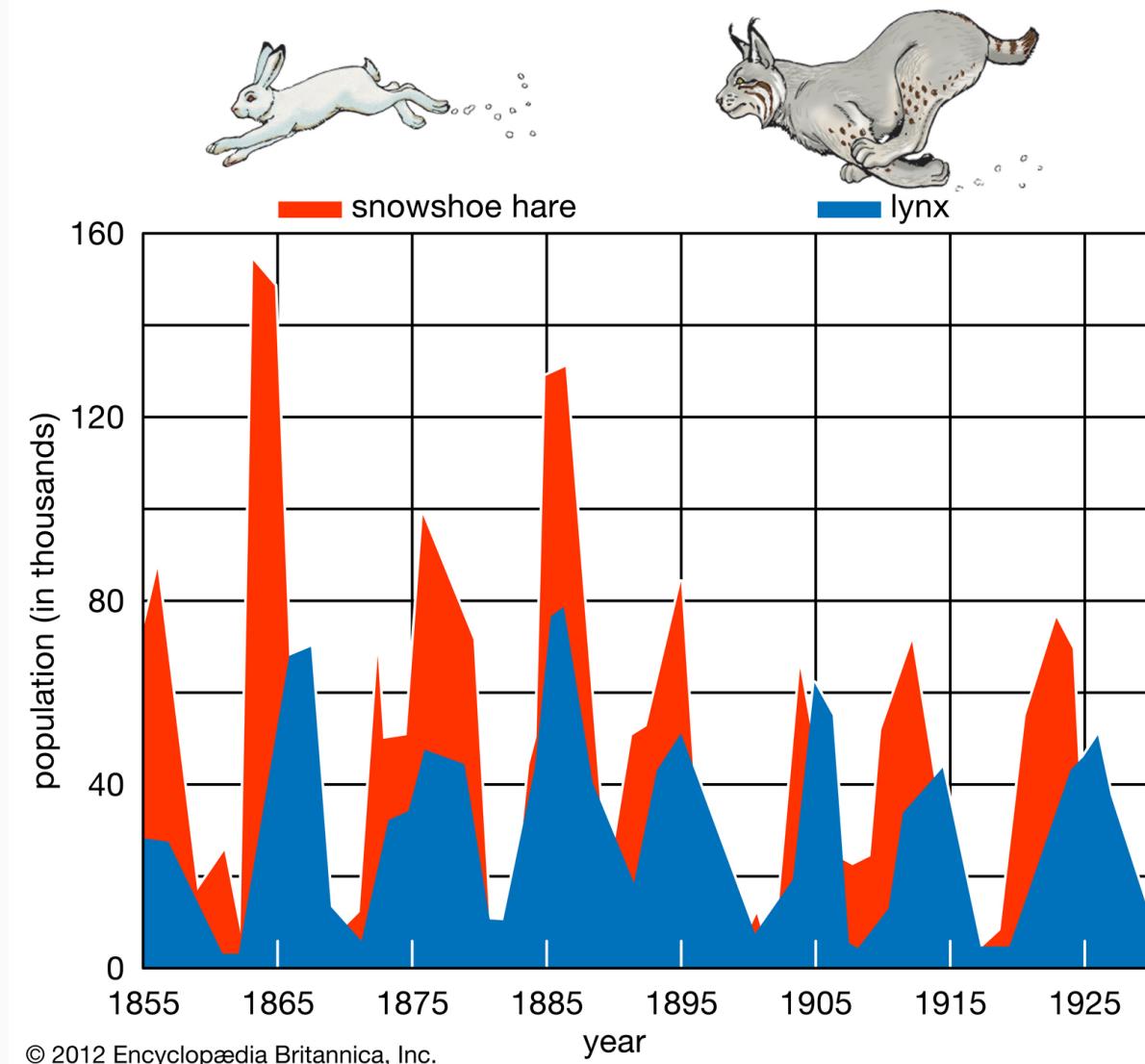


Atmospheric CO₂ at Mauna Loa Observatory









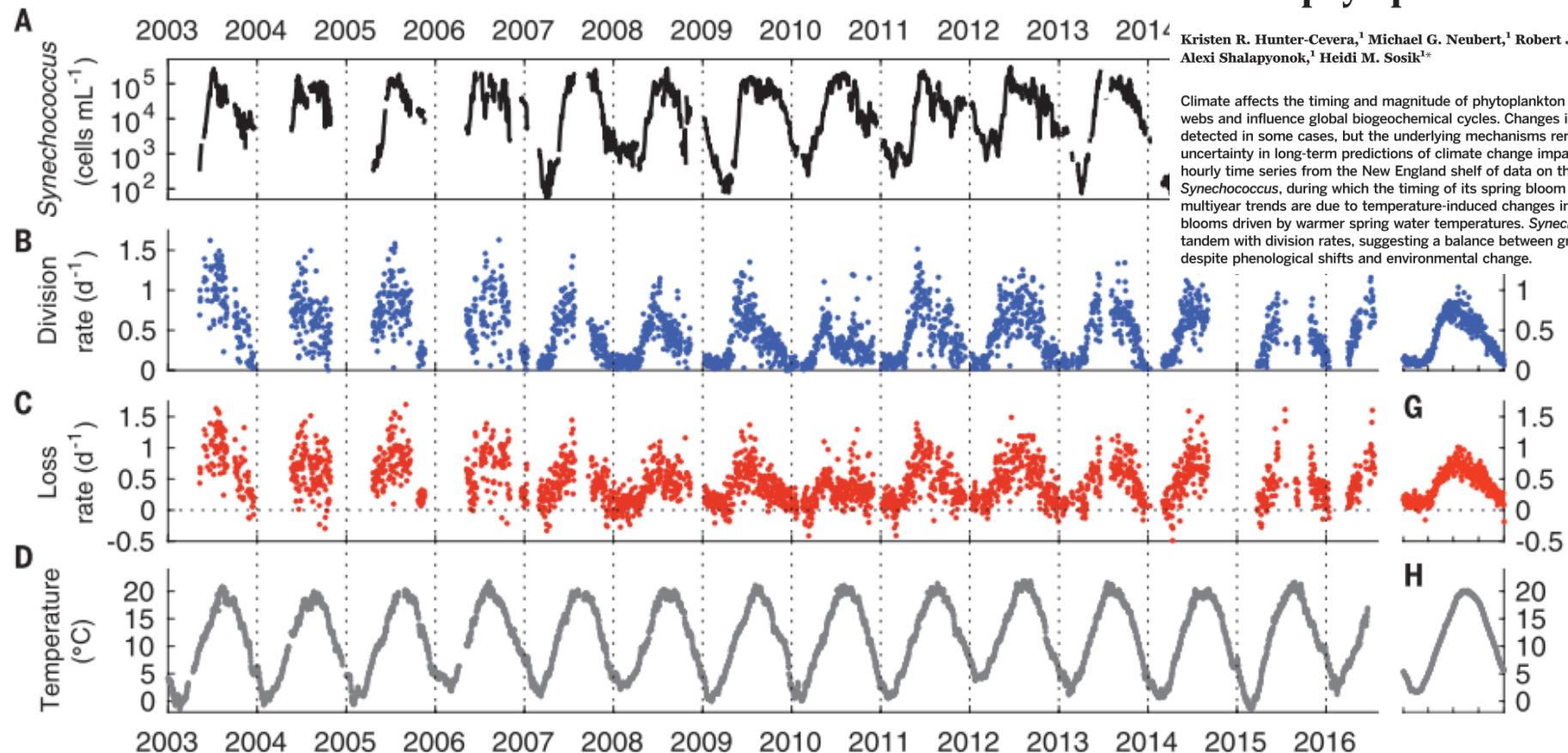
Synechococcus

PLANKTON DYNAMICS

Physiological and ecological drivers of early spring blooms of a coastal phytoplankton

Kristen R. Hunter-Cevera,¹ Michael G. Neubert,¹ Robert J. Olson,¹ Andrew R. Sollow,² Alexi Shalapyonok,¹ Heidi M. Sosik^{1*}

Climate affects the timing and magnitude of phytoplankton blooms that fuel marine food webs and influence global biogeochemical cycles. Changes in bloom timing have been detected in some cases, but the underlying mechanisms remain elusive, contributing to uncertainty in long-term predictions of climate change impacts. Here we describe a 13-year hourly time series from the New England shelf of data on the coastal phytoplankton *Synechococcus*, during which the timing of its spring bloom varied by 4 weeks. We show that multiyear trends are due to temperature-induced changes in cell division rate, with earlier blooms driven by warmer spring water temperatures. *Synechococcus* loss rates shift in tandem with division rates, suggesting a balance between growth and loss that has persisted despite phenological shifts and environmental change.



Synechoccus

Synechococcus

- Discovered in 1979 - Epifluorescence microscopy

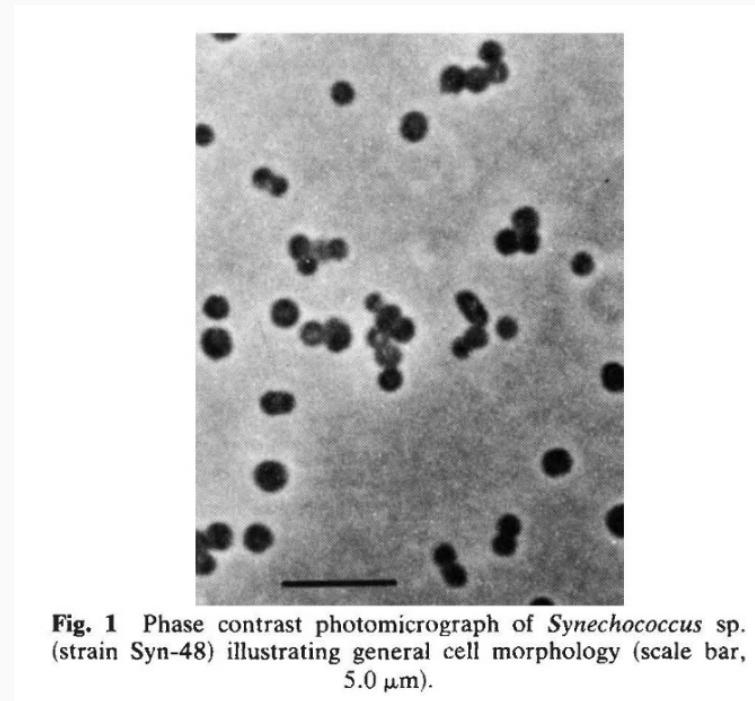
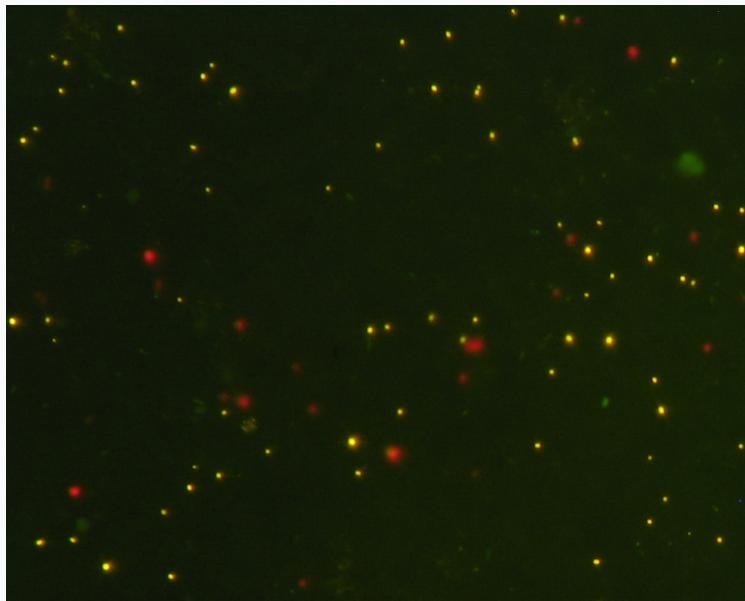
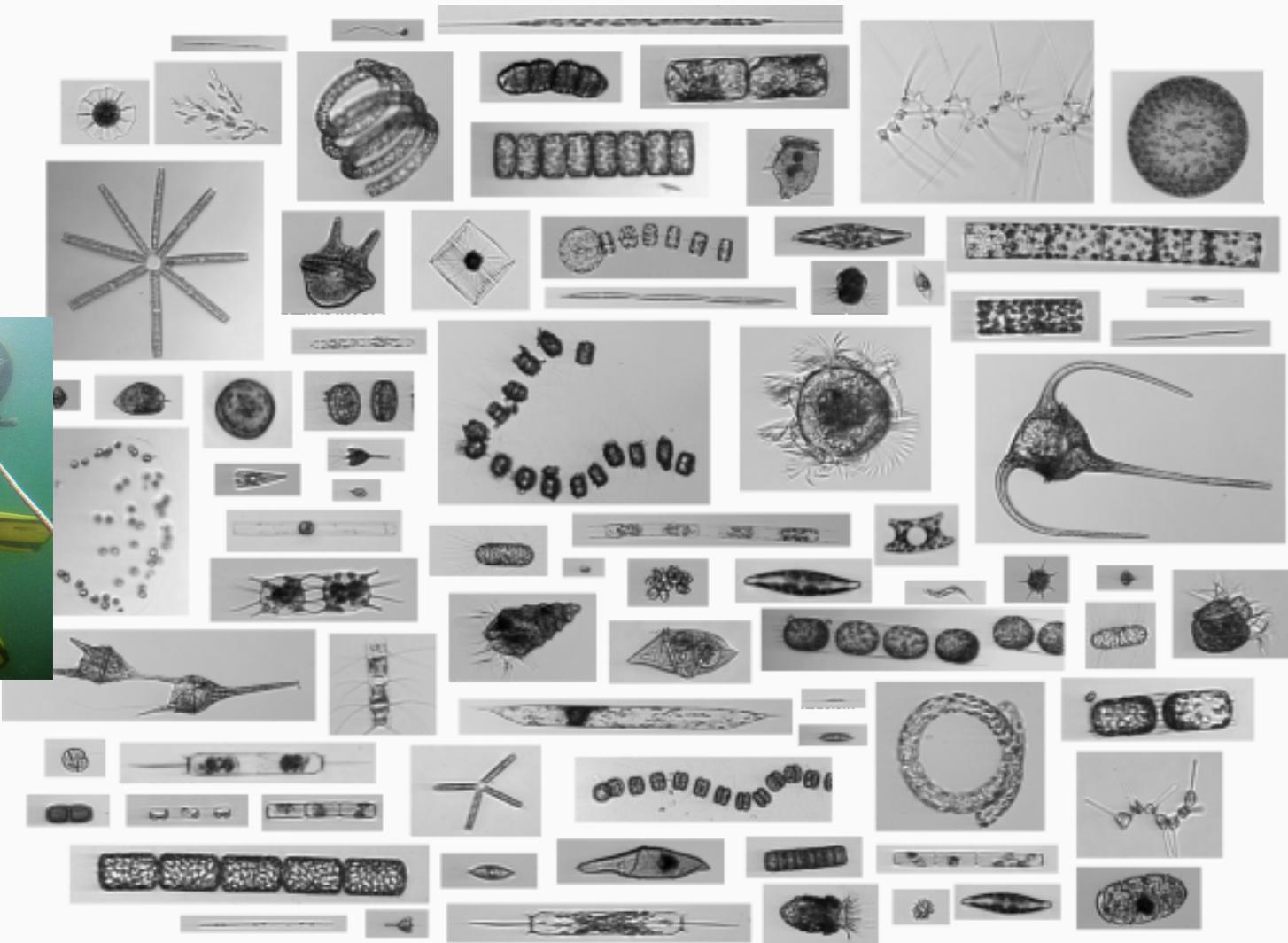


Fig. 1 Phase contrast photomicrograph of *Synechococcus* sp. (strain Syn-48) illustrating general cell morphology (scale bar, 5.0 μm).

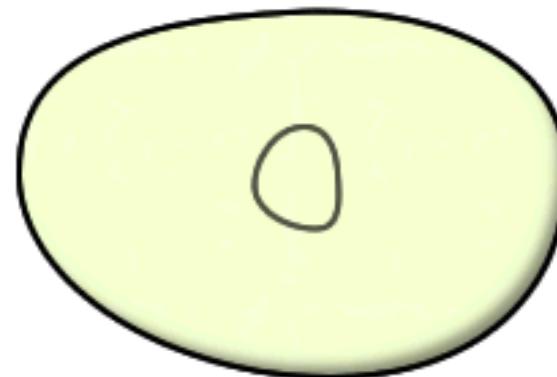
Synechoccus

- Imaging and flow cytometry



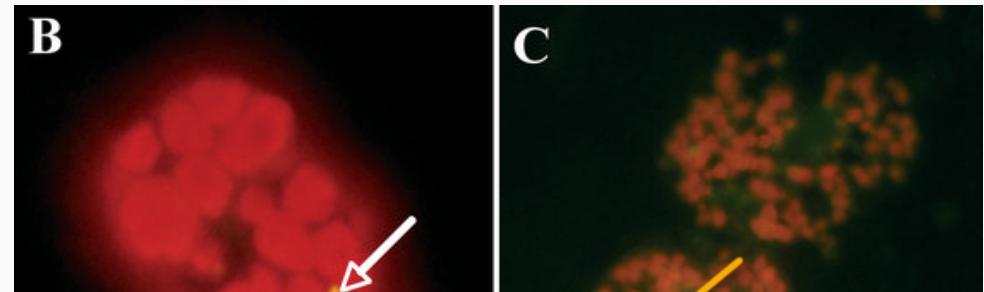
Synechoccus

- Binary fission
- Typically once every day



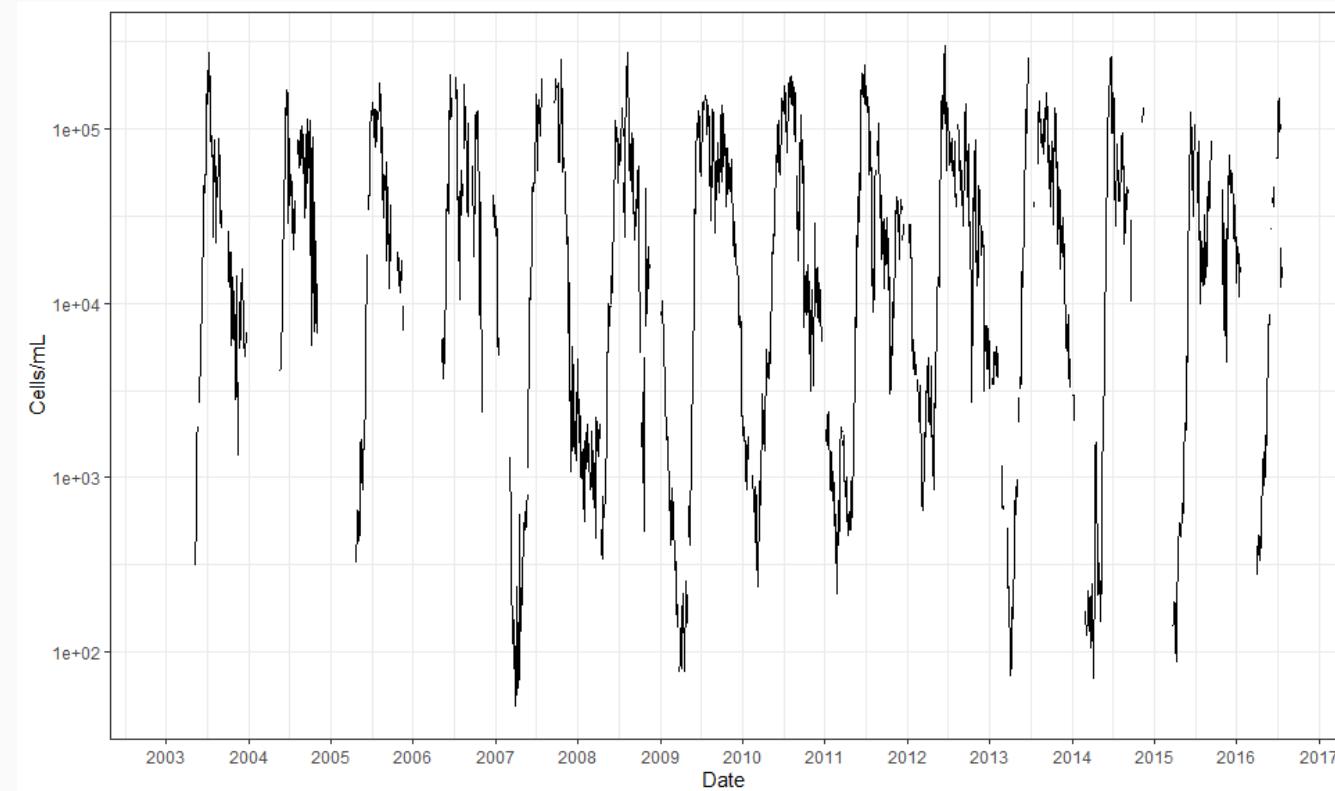
Synechoccus

- Virus
- Predation
- Cell death (UV, nutrient deprivation)

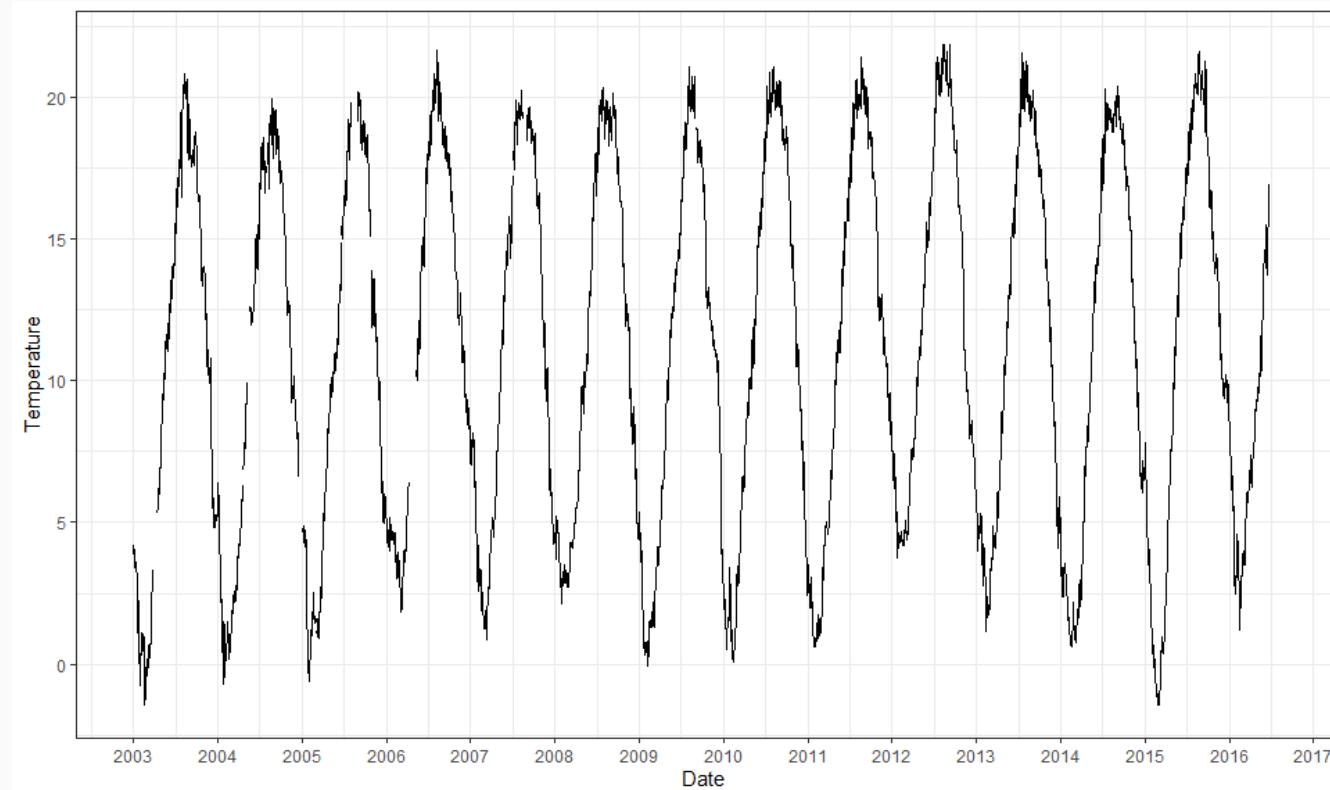


Synechoccus

Synechococcus



Synechoccus



Dates and times with lubridate :: CHEAT SHEET



Date-times



2017-11-28 12:00:00
A date-time is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC
`d <- as_datetime(1511870400)`
`## "2017-11-28 12:00:00 UTC"`

2017-11-28
A date is a day stored as the number of days since 1970-01-01
`d <- as_date(17498)`
`## "2017-11-28"`

12:00:00
An hms is a time stored as the number of seconds since 00:00:00
`t <- hms(as.hms(85))`
`## "00:01:25"`

PARSE DATE-TIMES (Convert strings or numbers to date-times)

- Identify the order of the year (y), month (m), day (d), hour (h), minute (m) and second (s) elements in your data.
- Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00 `ymd_hms()`, `ymd_hm()`, `ymd_h()`.
`ymd_hms("2017-11-28T14:02:00")`

2017-22-12 10:00:00 `ymd_hms()`, `ymd_hm()`, `ymd_h()`.
`ymd_hms("2017-22-12 10:00:00")`

11/28/2017 1:02:03 `mdy_hms()`, `mdy_hm()`, `mdy_h()`.
`mdy_hms("11/28/2017 1:02:03")`

1 Jan 2017 23:59:59 `dmy_hms()`, `dmy_hm()`, `dmy_h()`.
`dmy_hms("1 Jan 2017 23:59:59")`

20170131 `ymd()`, `ymd_h()`, `ymd_hm()`.
`ymd("20170131")`

July 4th, 2000 `dmy()`, `dmy_h()`, `dmy_hm()`.
`dmy("July 4th, 2000")`

4th of July '99 `dmy()`, `dmy_h()`, `dmy_hm()`.
`dmy("4th of July '99")`

2001: Q3 `yq()` Q for quarter. `yq("2001: Q3")`

2:01 `hms:hms()` Also `lubridate::hms()`, `hm()` and `ms()`, which return periods.* `hms:hms(sec = 0, min = 1, hours = 2)`

2017.5 `date_decimal(decimal, tz = "UTC")`
`date_decimal(2017.5)`

now(tzone = "") Current time in tz (defaults to system tz). `now()`

today(tzone = "") Current date in a tz (defaults to system tz). `today()`

fast_strptime() Faster strptime.
`fast_strptime("9/1/01", "%y/%m/%d")`

parse_date_time() Easier strptime.
`parse_date_time("9/1/01", "ymd")`

GET AND SET COMPONENTS

Use an accessor function to get a component. Assign into an accessor function to change a component in place.

`d ## "2017-11-28"`
`day(d) ## 28`
`day(d) <- 1`
`d ## "2017-11-01"`

2018-01-31 11:59:59 `date(x)` Date component. `date(dt)`

2018-01-31 11:59:59 `year(x)` Year. `year(dt)`
`isoyear(x)` The ISO 8601 year.
`epiyear(x)` Epidemiological year.

2018-01-31 11:59:59 `month(x, label, abbr)` Month. `month(dt)`

2018-01-31 11:59:59 `day(x)` Day of month. `day(dt)`
`wday(x, label, abbr)` Day of week. `wday(dt)`
`qday(x)` Day of quarter.

2018-01-31 11:59:59 `hour(x)` Hour. `hour(dt)`
`minute(x)` Minutes. `minute(dt)`

2018-01-31 11:59:59 `second(x)` Seconds. `second(dt)`

2018-01-31 11:59:59 `week(x)` Week of the year. `week(dt)`
`iso week` ISO 8601 week.
`epi week` Epidemiological week.

2018-01-31 11:59:59 `quarter(x, with_year = FALSE)` Quarter. `quarter(dt)`

2018-01-31 11:59:59 `semester(x, with_year = FALSE)` Semester. `semester(dt)`

2018-01-31 11:59:59 `am(x)` Is it in the am? `am(dt)`
`pm(x)` Is it in the pm? `pm(dt)`

2018-01-31 11:59:59 `dst(x)` Is it daylight savings? `dst(dt)`

2018-01-31 11:59:59 `leap_year(x)` Is it a leap year?
`leap_year(dt)`

2018-01-31 11:59:59 `update(object, ..., simple = FALSE)`
`update(dt, mday = 2, hour = 1)`

Round Date-times

floor_date(x, unit = "second")
Round down to nearest unit.
`floor_date(dt, unit = "month")`

round_date(x, unit = "second")
Round to nearest unit.
`round_date(dt, unit = "month")`

ceiling_date(x, unit = "second")
`change_on_boundary = NULL`
Round up to nearest unit.
`ceiling_date(dt, unit = "month")`

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE)
Roll back to last day of previous month. `rollback(dt)`

Stamp Date-times

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also `stamp_date()` and `stamp_time()`.

- Derive a template, create a function
`sf <- stamp("Created Sunday, Jan 17, 1999 3:34")`
- Apply the template to dates
`sf(ymd("2010-04-05"))`
`## [1] "Created Monday, Apr 05, 2010 00:00"`

Tip: use a date with day > 12

Time Zones

R recognizes ~600 time zones. Each encodes the time zone, Daylight Savings Time, and historical calendar variations for an area. R assigns one time zone per vector.

Use the UTC time zone to avoid Daylight Savings.

OlsonNames() Returns a list of valid time zone names. `OlsonNames()`

5:00 Pacific
4:00 Mountain
6:00 Central
7:00 Eastern
7:00 with_tz(time, tzzone = "") Get the same date-time in a new time zone (a new clock time).
`with_tz(dt, "US/Pacific")`

7:00 Pacific
PT Mountain
7:00 Central
7:00 Eastern
7:00 with_tz(time, tzzone = "") Get the same clock time in a new time zone (a new date-time).
`force_tz(dt, "US/Pacific")`



Dates are often read as character from files. In order to be recognized in R as dates, they must be parsed

```
library(lubridate)  
(bdy <- ymd(19530822))
```

```
[1] "1953-08-22"
```

```
(bdy <- dmy("22-08-1953"))
```

```
[1] "1953-08-22"
```

```
(bdy <- dmy("22 aug 1953"))
```

```
[1] "1953-08-22"
```

```
bdy <- dmy(22-08-1953)
```

Warning: All formats failed to parse. No formats found.

| do not forget the quotes

```
(bdy_time <- dmy_hm("22-08-1953 16:00"))
```

```
[1] "1953-08-22 16:00:00 UTC"
```

```
bdy <- dmy("22-08-1953")
year(bdy)
```

```
[1] 1953
```

```
month(bdy)
```

```
[1] 8
```

```
day(bdy)
```

```
[1] 22
```

```
wday(bdy, label=TRUE, abbr = FALSE, week_start = 1)
```

```
[1] samedi
```

```
7 Levels: lundi < mardi < mercredi < jeudi < vendredi < ... < dimanche
```

```
# Rounding to start of the month  
floor_date(bday, unit="month")
```

```
[1] "1953-08-01"
```

```
# Adding one month  
bday + months(1)
```

```
[1] "1953-09-22"
```

```
# Adding years  
bday + years(66)
```

```
[1] "2019-08-22"
```

```
# Creating intervals  
(binterval ← lubridate::interval(bday, Sys.Date()))
```

```
[1] 1953-08-22 UTC--2020-04-09 UTC
```

```
# Number of seconds in interval  
int_length(binterval)
```

```
[1] 2102716800
```

```
# Number of days  
seconds_to_period(int_length(binterval))
```

```
[1] "24337d 0H 0M 0S"
```

Tidy tools for time series.

PACKAGES



tsibble
Temporal data frames and tools



fable
Tidy forecasting



feasts
Feature extraction and statistics



```
library(tidyverse)
library(lubridate)

# Time series manipulation
library(tsibble)

# Time series graphics and statistics
library(feasts)

# Moving computations
library(slidr)

# Forecasting functions
library(fable)

# Interpolation
library(zoo)

theme_set(theme_bw())
```

- Like a dataframe or a tibble but geared for time series
- index : the time variable - here `year` - but can be day, minute etc...
- key : one or several column that corresponding to different time series - here `country`

```
library(tsibbledata)
global_economy
```

```
# A tsibble: 15,150 x 9 [1Y]
# Key:   Country [263]
```

	Country	Code	Year	GDP	Growth	CPI	Imports	Exports	Population
	<fct>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Afghanistan	AFG	1960	537777811.	NA	NA	7.02	4.13	8996351
2	Afghanistan	AFG	1961	548888896.	NA	NA	8.10	4.45	9166764
3	Afghanistan	AFG	1962	546666678.	NA	NA	9.35	4.88	9345868
4	Afghanistan	AFG	1963	751111191.	NA	NA	16.9	9.17	9533954
5	Afghanistan	AFG	1964	800000044.	NA	NA	18.1	8.89	9731361
6	Afghanistan	AFG	1965	1006666638.	NA	NA	21.4	11.3	9938414
7	Afghanistan	AFG	1966	1399999967.	NA	NA	18.6	8.57	10152331
8	Afghanistan	AFG	1967	1673333418.	NA	NA	14.2	6.77	10372630

```
key(global_economy)
```

```
[[1]]
```

```
Country
```

- A series with 2 keys

```
olympic_running
```

```
# A tsibble: 312 x 4 [4Y]
# Key:      Length, Sex [14]
#       Year Length Sex     Time
#       <dbl> <fct>  <chr> <dbl>
1   1896 100m    men     12
2   1900 100m    men     11
3   1904 100m    men     11
4   1908 100m    men    10.8
5   1912 100m    men    10.8
6   1916 100m    men     NA
7   1920 100m    men    10.8
8   1924 100m    men    10.6
9   1928 100m    men    10.8
10  1932 100m   men    10.3
# ... with 302 more rows
```

```
key(olympic_running)
```

```
[[1]]
Length
[[2]]
Sex
```

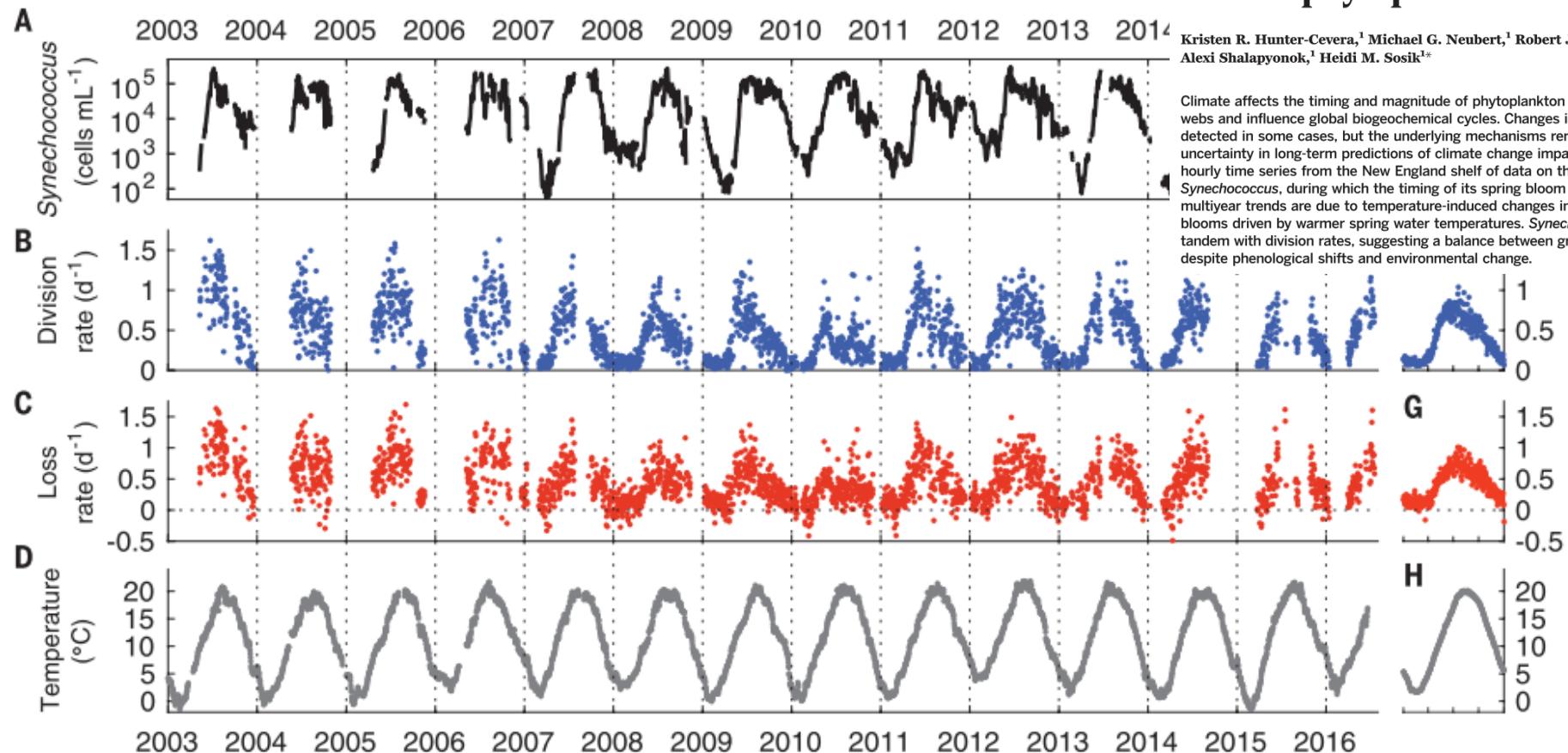
Synechococcus

PLANKTON DYNAMICS

Physiological and ecological drivers of early spring blooms of a coastal phytoplankton

Kristen R. Hunter-Cevera,¹ Michael G. Neubert,¹ Robert J. Olson,¹ Andrew R. Sollow,² Alexi Shalapyonok,¹ Heidi M. Sosik^{1*}

Climate affects the timing and magnitude of phytoplankton blooms that fuel marine food webs and influence global biogeochemical cycles. Changes in bloom timing have been detected in some cases, but the underlying mechanisms remain elusive, contributing to uncertainty in long-term predictions of climate change impacts. Here we describe a 13-year hourly time series from the New England shelf of data on the coastal phytoplankton *Synechococcus*, during which the timing of its spring bloom varied by 4 weeks. We show that multiyear trends are due to temperature-induced changes in cell division rate, with earlier blooms driven by warmer spring water temperatures. *Synechococcus* loss rates shift in tandem with division rates, suggesting a balance between growth and loss that has persisted despite phenological shifts and environmental change.



```
syn ← read_tsv("data/Syn_temp_light_data.txt", na = c("", "NaN"))
```

Parsed with column specification:

```
cols(  
  Date = col_character(),  
  `Daily Avg. Synechococcus (cells/mL)` = col_double(),  
  `Division rate (1/d)` = col_double(),  
  `Loss rate (1/d)` = col_double(),  
  `Net growth rate (1/d)` = col_double(),  
  `Daily minimum mode of cell volume (um^3)` = col_double(),  
  `PE fluorescence at dawn` = col_double(),  
  `PE fluorescence at dawn normalized by cell volume at dawn (1/um^3)` = col_double(),  
  `Temperature (degree Celsius)` = col_double(),  
  `Daily incident radiation (MJ/um^2)` = col_double()  
)
```

- Many columns have names not compatible with R
- No data of interest in first rows and last rows
- We are going to concentrate on only 4 columns:
 - temperature
 - light
 - *Synechococcus* concentration

Date	Daily Avg. <i>Synechococcus</i> (cells/mL)	Division rate (1/d)	Loss rate (1/d)	Net growth rate (1/d)	Daily minimum mode of cell volume (μm^3)	PE fluorescence at dawn	PE fluorescence at dawn normalized by cell volume at dawn (1/ μm^3)	Temperature (degree Celsius)	Daily incident radiation (MJ/ μm^2)
01-Jan-2003								4.21	2.11
02-Jan-2003								4.06	1.84
03-Jan-2003								3.68	1.53
04-Jan-2003								3.74	3.38
05-Jan-2003								3.82	8.14

```

syn ← syn %>%
  mutate(Date = lubridate::dmy(Date)) %>%
  rename(date = Date,
         temperature= `Temperature (degree Celsius)` ,
         syn_ml = `Daily Avg. Synechococcus (cells/mL)` ,
         radiation= `Daily incident radiation (MJ/um^2)` ) %>%
  select(date, syn_ml, temperature, radiation) %>%
  filter(date > lubridate::ymd("2003-05-28") & date < lubridate::ymd("2018-09-04"))

```

date	syn_ml	temperature	radiation
2003-05-29	3344.51	11.95	22.83
2003-05-30	3521.90	12.00	28.28
2003-05-31	3939.43	11.92	19.40
2003-06-01	4796.46	11.70	
2003-06-02	4348.95	11.55	

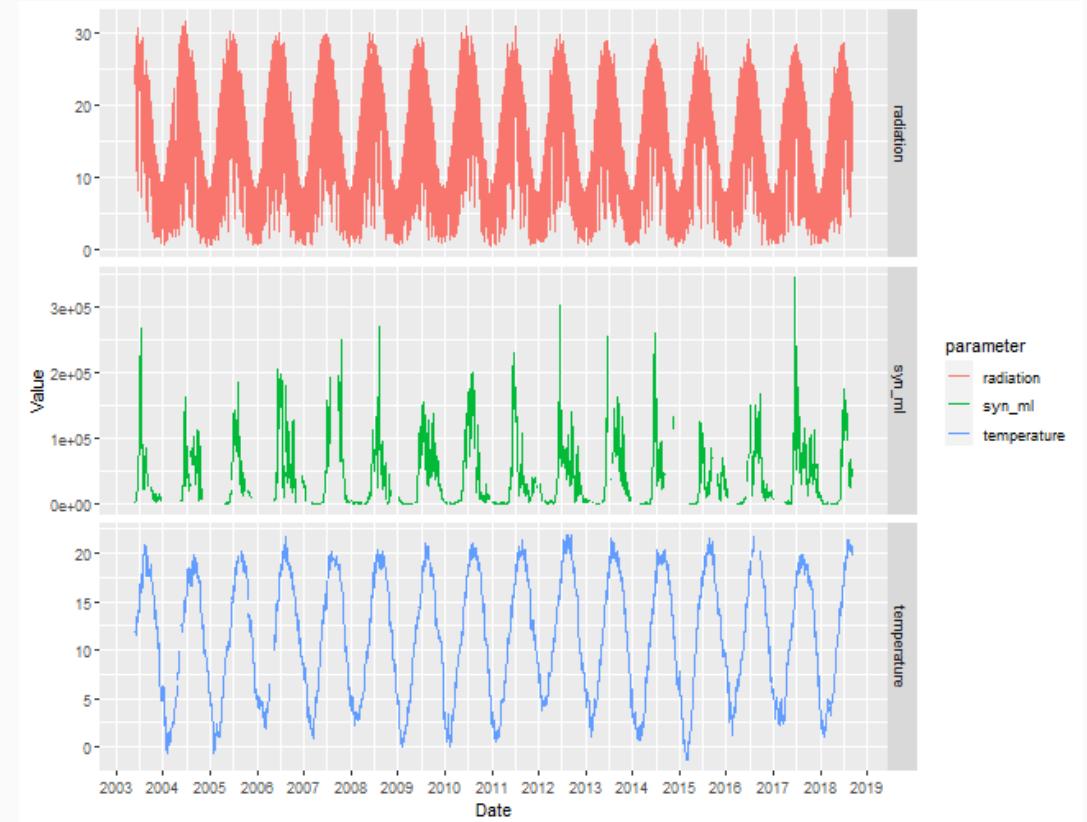
```
syn_long ← syn %>%
  pivot_longer(-date,
    names_to = "parameter",
    values_to = "value" )

tsyn ← tsibble::as_tsibble(syn_long,
                           index = date,
                           key = parameter,
                           regular = TRUE,
                           validate = TRUE)
```

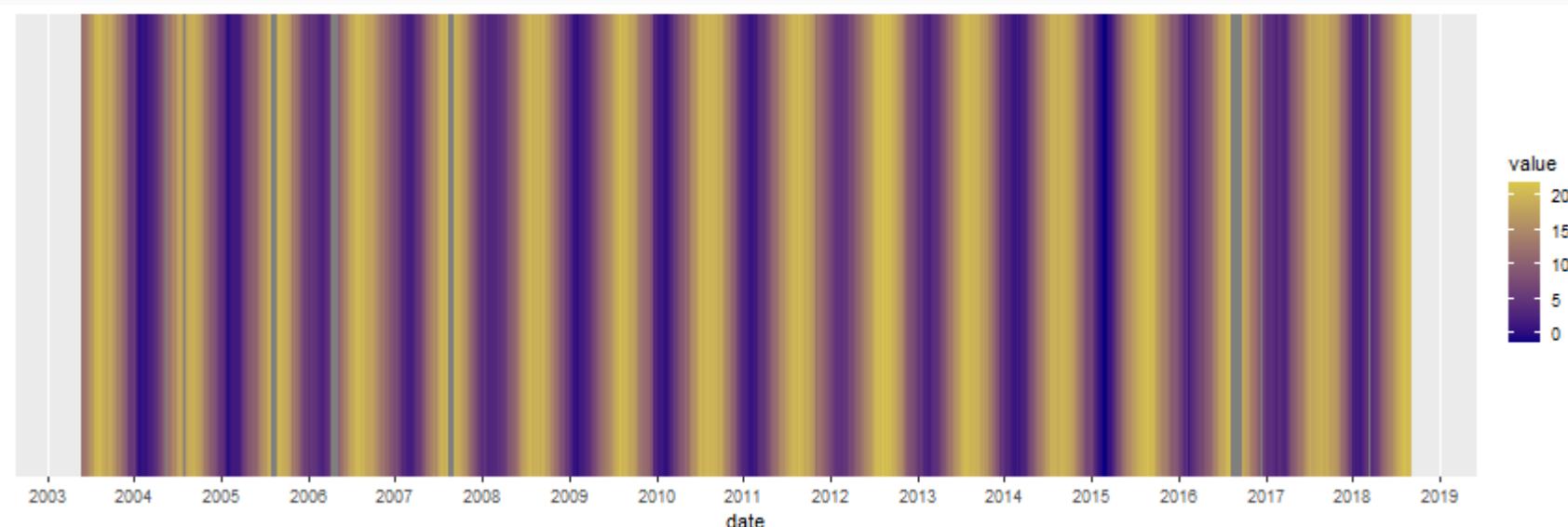
```
# A tsibble: 16,731 x 3 [1D]
# Key:      parameter [3]
#           date      parameter value
#           <date>    <chr>     <dbl>
1 2003-05-29 radiation  22.8
2 2003-05-30 radiation  28.3
3 2003-05-31 radiation  19.4
4 2003-06-01 radiation  NA
5 2003-06-02 radiation  NA
6 2003-06-03 radiation  NA
7 2003-06-04 radiation  NA
8 2003-06-05 radiation  10.6
9 2003-06-06 radiation  29.6
10 2003-06-07 radiation NA
# ... with 16,721 more rows
```

```
autoplot(tsyn) +  
  xlab("Date") + ylab("Value") +  
  scale_x_date(date_breaks="1 year",  
               date_labels = "%Y") +  
  facet_grid(rows = vars(parameter),  
             scales= "free_y")
```

- Can also use ggplot2 of course

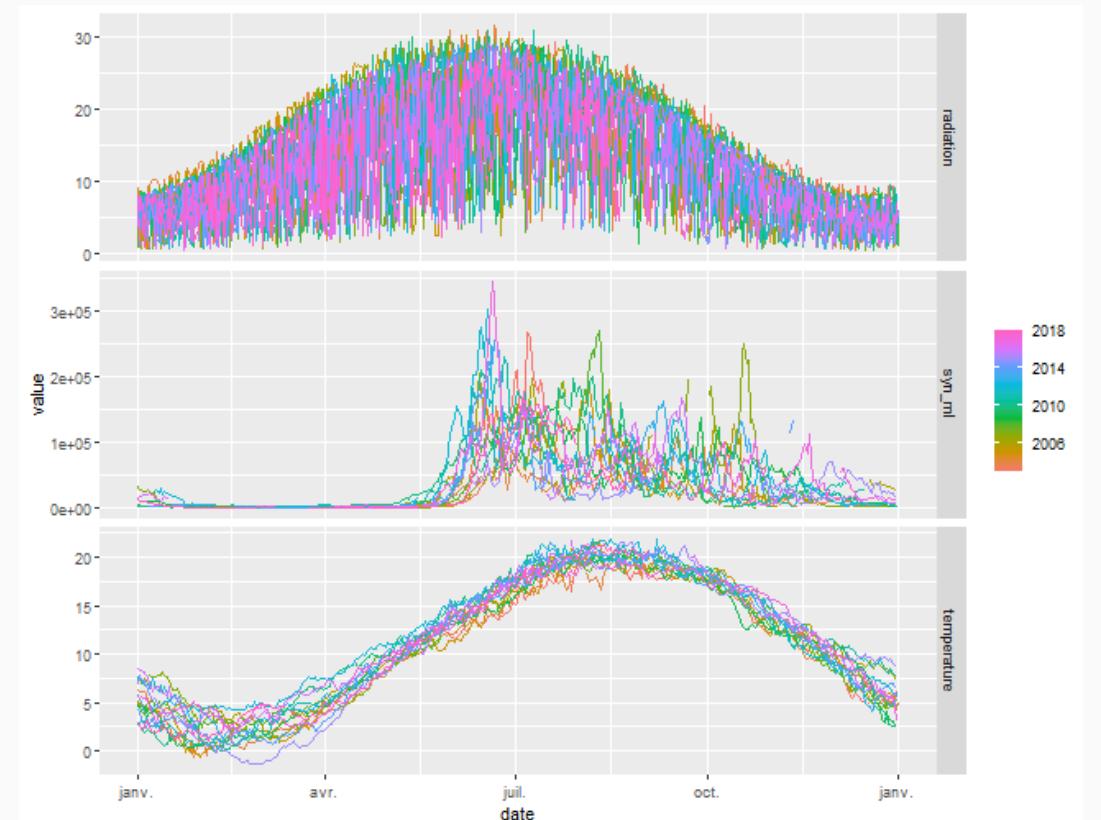


```
tsyn %>%
  filter( parameter = "temperature") %>%
  ggplot(aes(x = date, y = 1)) +
  geom_tile(aes(fill = value)) +
  scale_fill_gradient2(low = "navy", mid = "yellow", high = "red", midpoint = 28) +
  ylab("") + scale_y_discrete(expand = c(0, 0)) +
  scale_x_date(date_breaks="1 year", date_labels = "%Y")
```



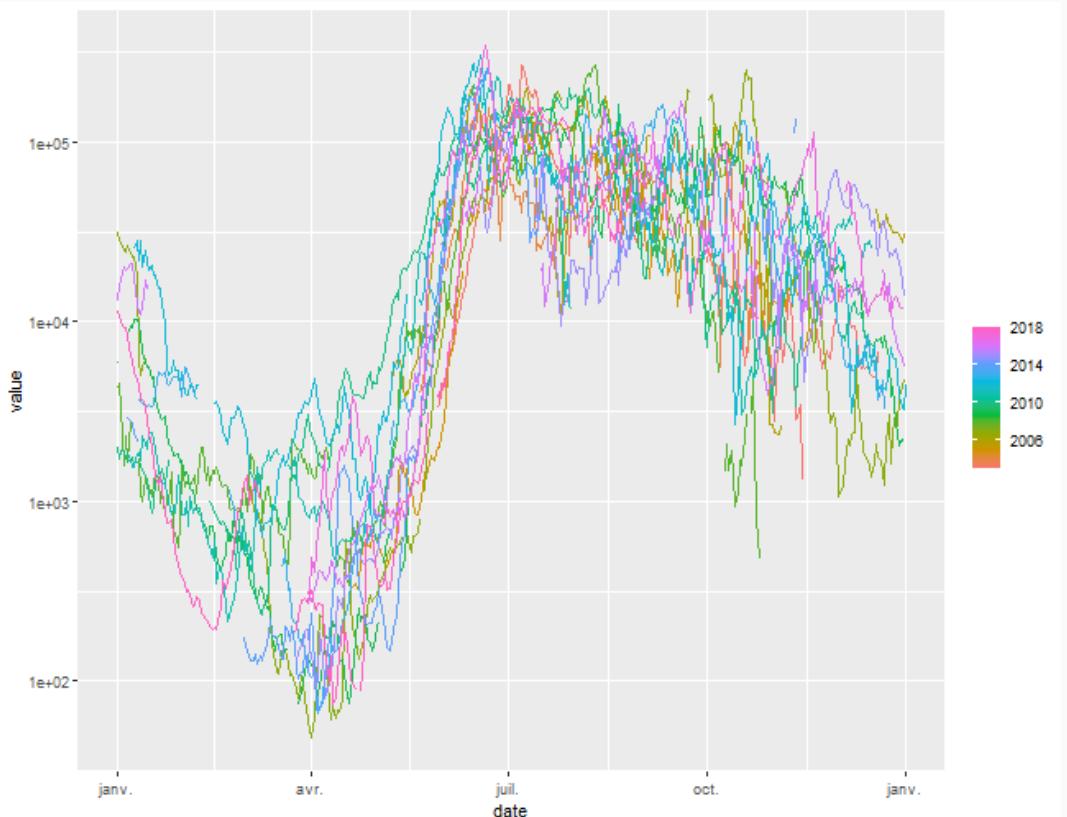
For all variables

```
tsyn %>%  
  feasts::gg_season(y=value, period="year")
```



For cell concentration better to use log scale

```
tsyn %>% filter(parameter = "syn_ml") %>%
  feasts::gg_season(y=value, period="year") +
  scale_y_log10()
```



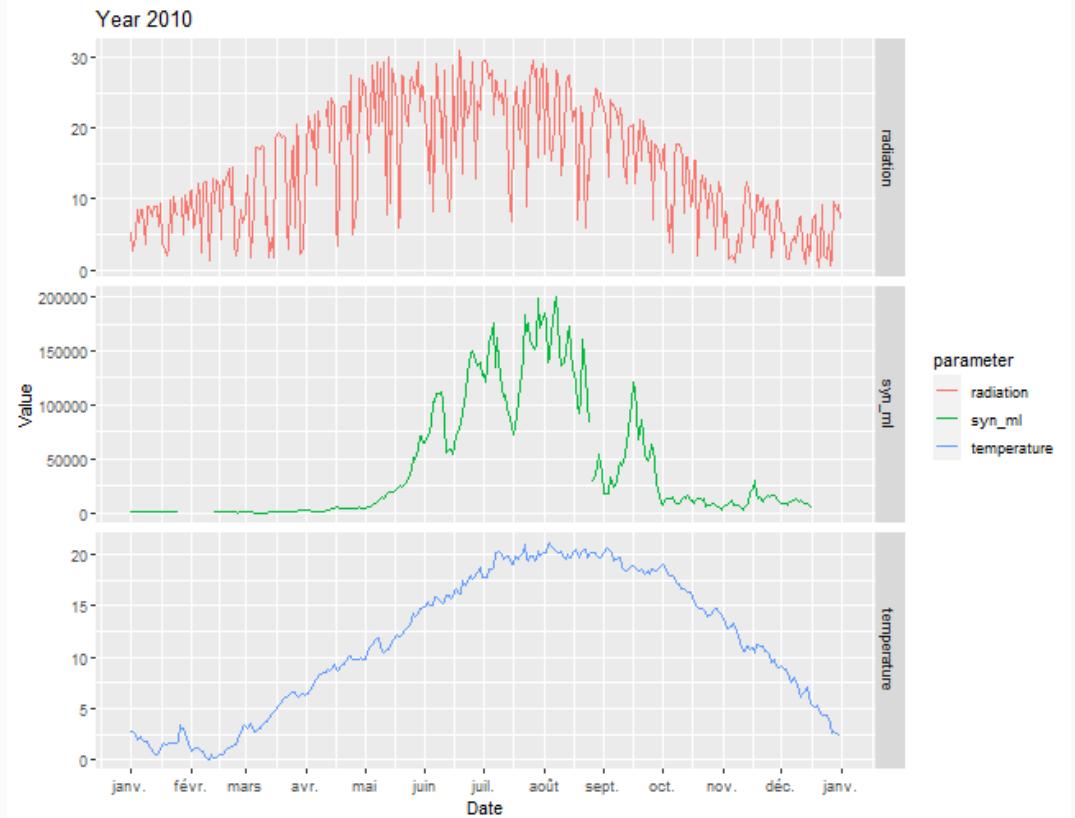
```

tsyn %>%
  filter(year(date) == 2010) %>%
  autoplot() +
  xlab("Date") + ylab("Value") +
  scale_x_date(date_breaks = "1 month",
               date_labels = "%b") +
  facet_grid(rows = vars(parameter),
             scales = "free_y") +
  ggtitle("Year 2010")

```

Note holes in data.

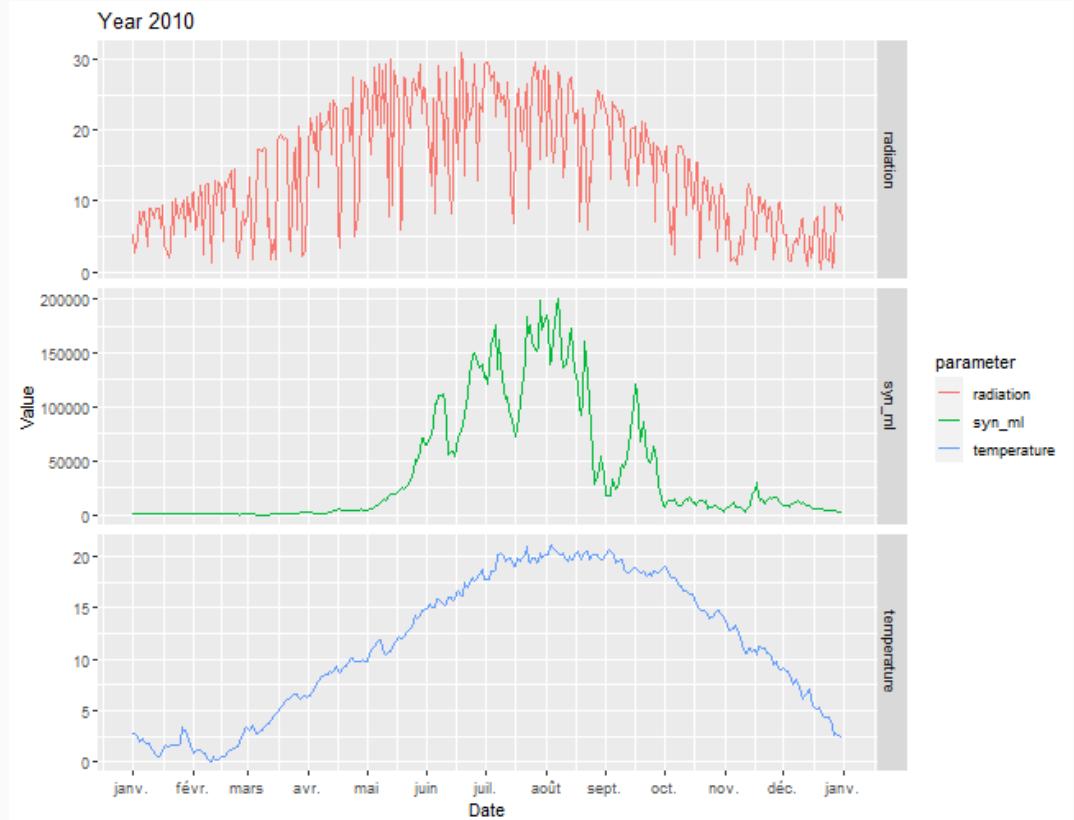
These holes must be filled before time series analysis



Package zoo

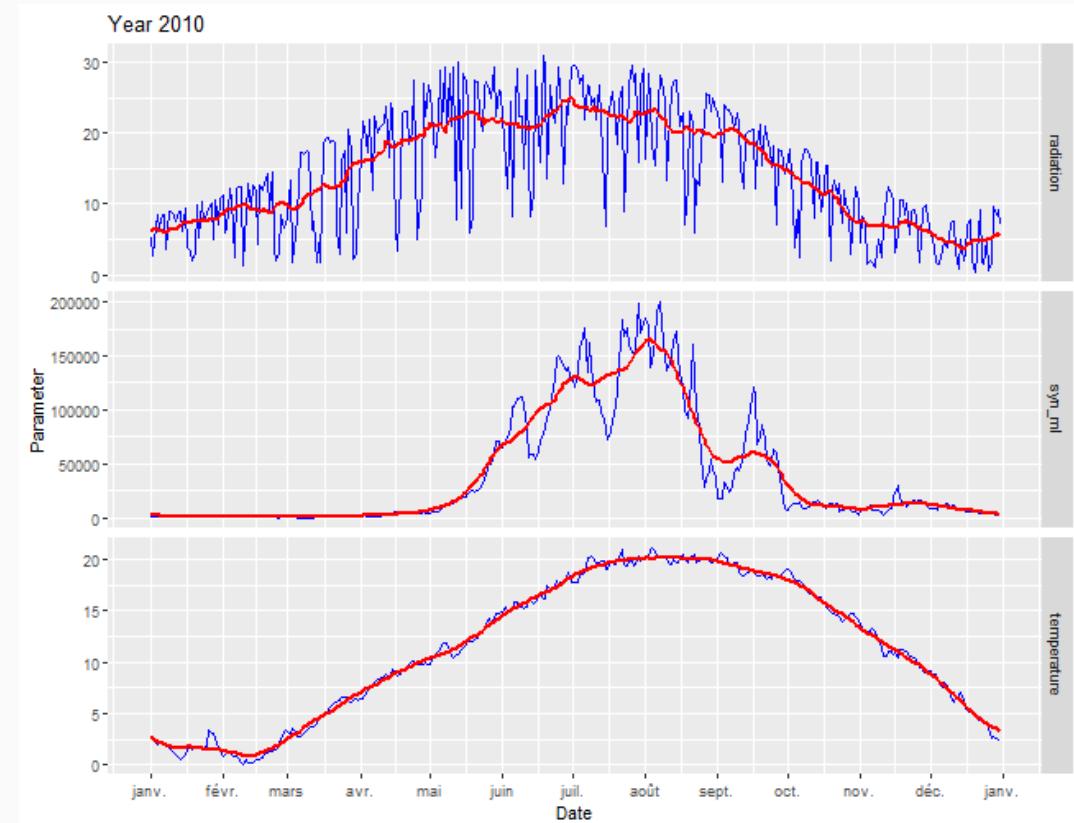
- `zoo::na.approx()` - Linear interpolation
- `zoo::na.spline()` - Spline interpolation

```
tsyn_filled <- tsyn %>%
#  filter(parameter = "temperature") %>%
  tsibble::group_by_key()%>%
  mutate(value = zoo::na.approx(value, na.rm = FALSE))
```



- All points have same weight
- Smoothing increases with
- Package `slider` is currently developed

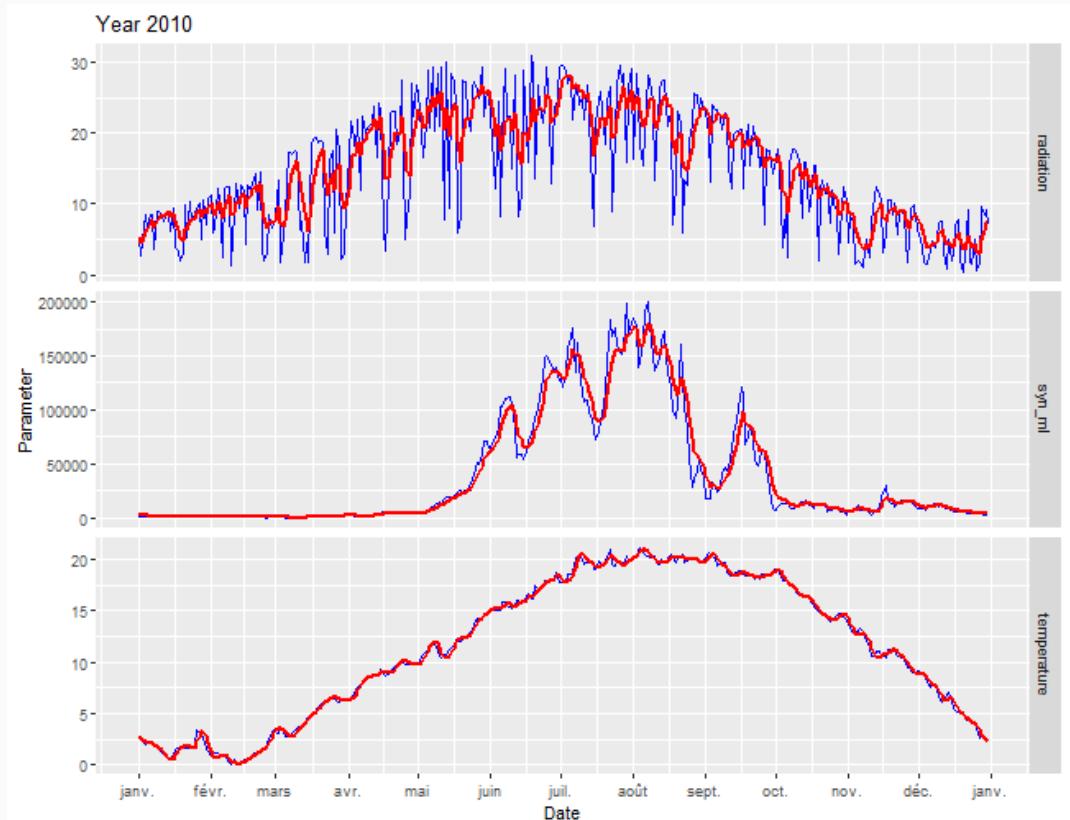
```
tsyn_MA ← tsyn_filled %>%  
  group_by_key() %>%  
  mutate(value_MA = tsibble::slide_dbl(value,  
                                         mean, na.rm = TRUE,  
                                         .size = 25,  
                                         .align = "center"))
```



- Forecasting method (use points from past only)
- Recent points have more weight
- Smoothing increases with α decreasing

```
tsyn_ETS ← tsyn_filled %>%
  model(fable::ETS(value ~
    season(period = "1 year") +
    trend(alpha = 0.3)))
```

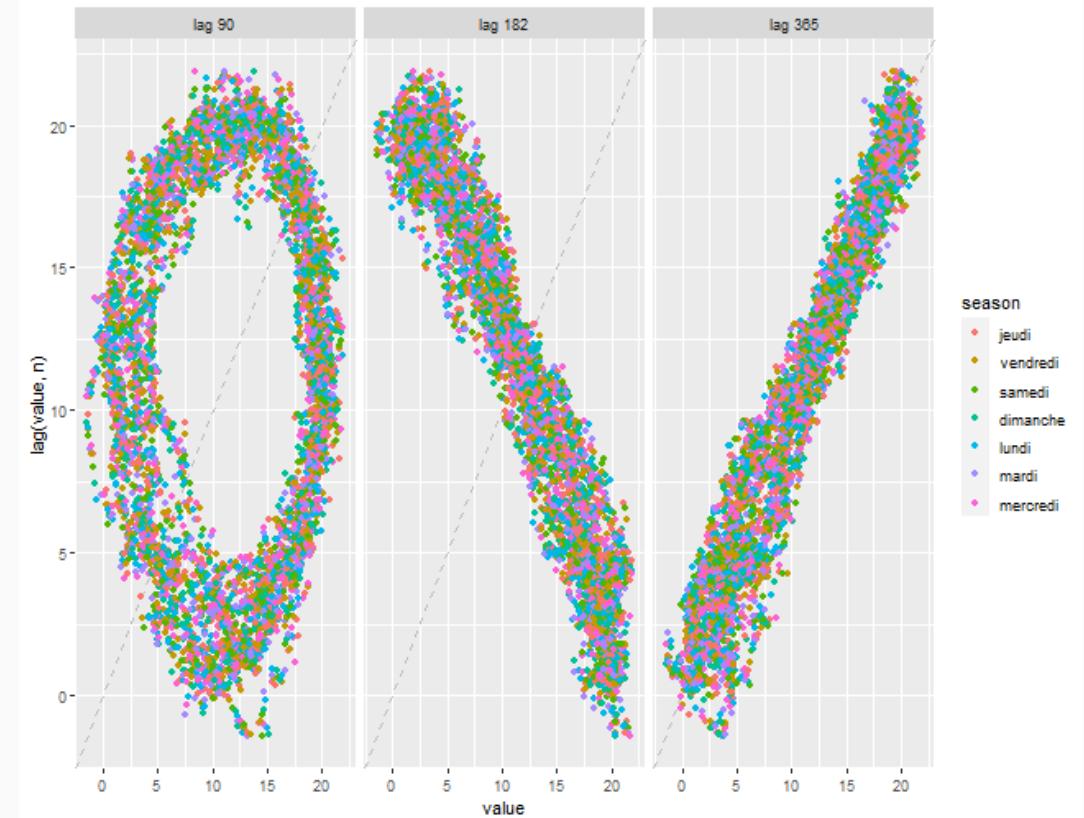
parameter	.model	date	value	level	slope	remainder
radiation	fable::ETS(value ~ season(period = "1 year") + trend(alpha = 0.3))	2003-05-28	19.89920	0.5388970		
radiation	fable::ETS(value ~ season(period = "1 year") + trend(alpha = 0.3))	2003-05-29	22.83	21.15567	0.5391361	0.1170316
radiation	fable::ETS(value ~ season(period = "1 year") + trend(alpha = 0.3))	2003-05-30	28.28	23.67036	0.5397947	0.3035379



Use function `feasts::gg_lag` to visualize

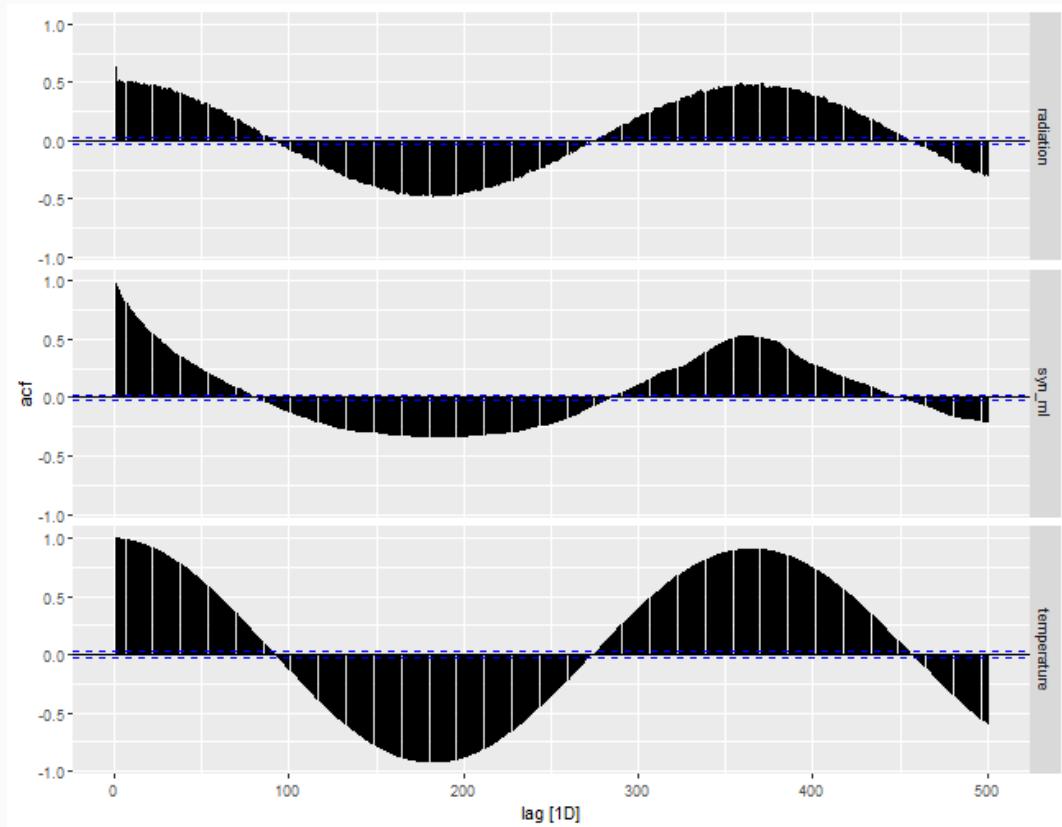
- For temperature

```
tsyn %>%
  filter(parameter = "temperature") %>%
  feasts::gg_lag(lags=c(90, 182, 365), geom = "point")
```



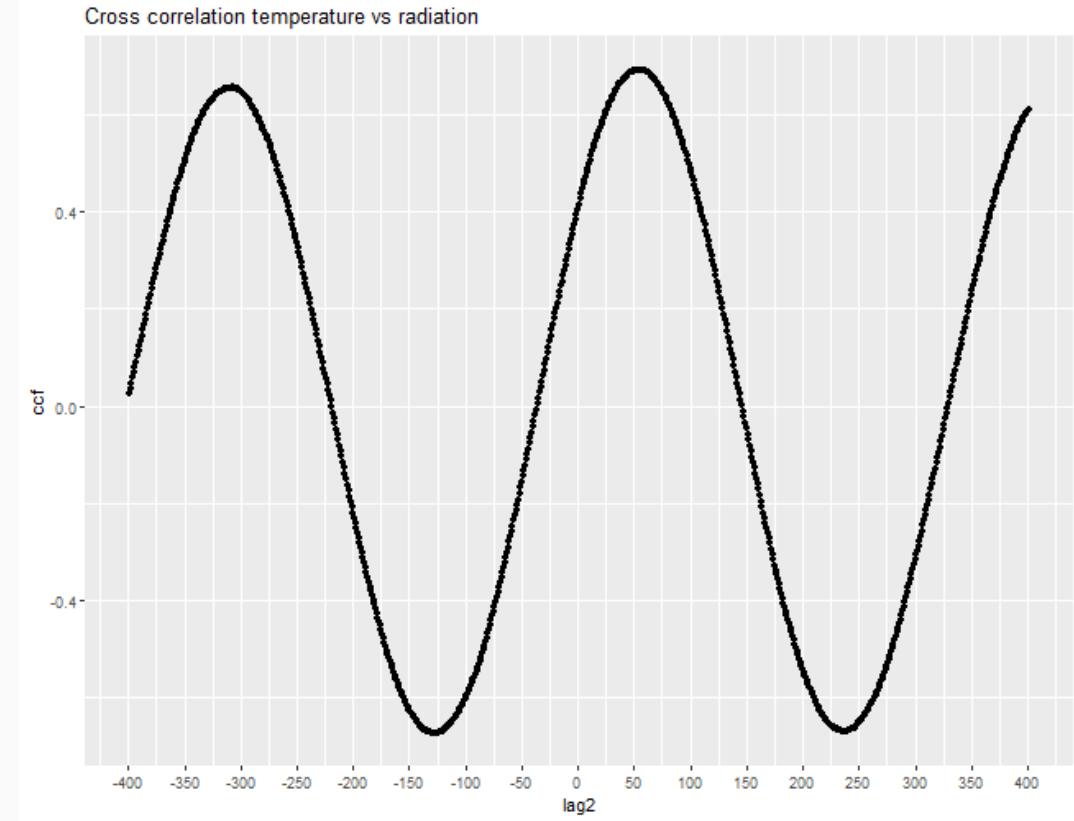
Use function `feasts::ACF` to compute

```
tsyn_filled %>%  
  feasts::ACF(value, lag_max = 500) %>%  
  autoplot() +  
  scale_x_continuous(breaks=seq(0,500, by=100))
```



- Same as autocorrelation but now between 2 series (light and temperature)
- Use function `feasts::CCF`
- Go from long form to wide form because variables must be in 2 different columns

```
lag_max = 400
tsyn_filled %>%
  pivot_wider(names_from = "parameter",
              values_from = "value") %>%
  as_tsibble(index = date) %>%
  feasts::CCF(temperature, radiation,
              lag_max = lag_max) %>%
  tibble::rowid_to_column(var = "lag2") %>%
  mutate(lag2 = lag2 - lag_max) %>%
  ggplot() +
  geom_point(aes(x=lag2, y = ccf)) +
  scale_x_continuous(breaks=seq(-lag_max,lag_max, by=50))
  ggttitle("Temperature vs radiation")
```

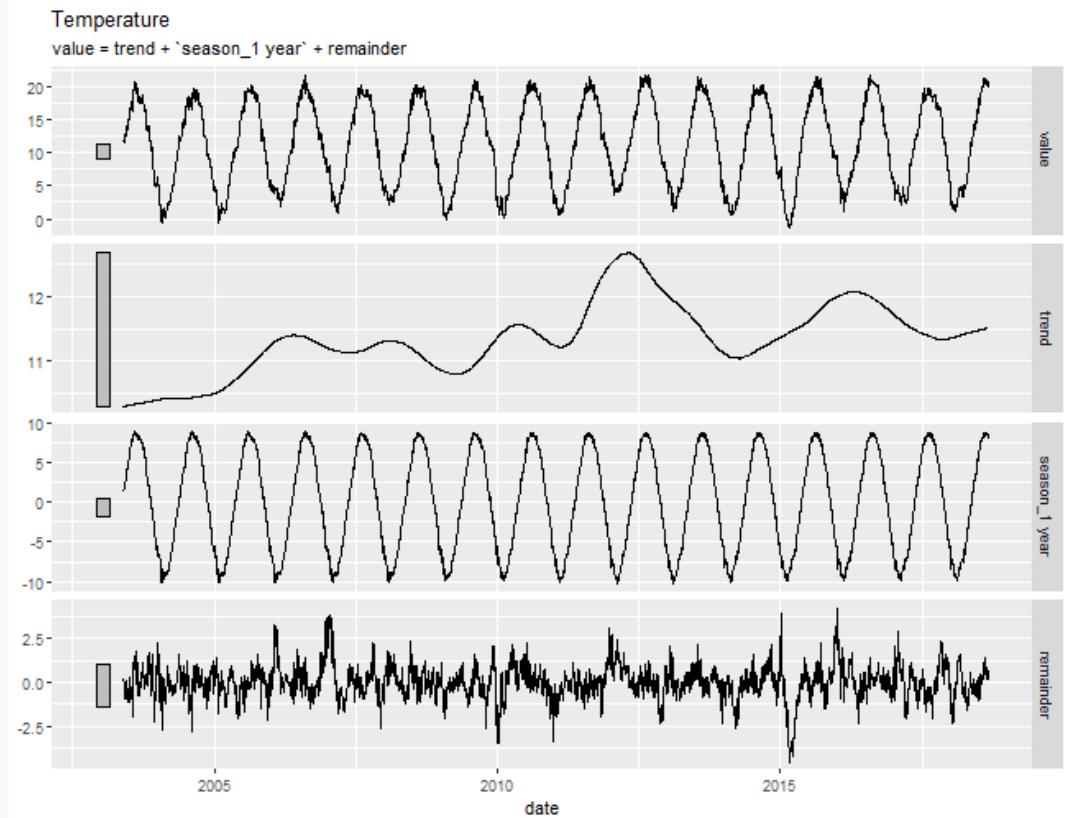


Three elements: season + trend + error

- Use function `feasts::STL`
- For temperature

```
dcmp_temp ← tsyn_filled %>%
  filter(parameter = "temperature") %>%
  model(stl = feasts::STL(value ~
    season(period= "1 year", wi
    robust = TRUE))
```

parameter	.model	date	value	trend	season_1year	remainder	season_adjust
temperature	stl	2003-05-29	11.95	10.30981	1.482655	0.1575318	10.46735
temperature	stl	2003-05-30	12.00	10.31027	1.726744	-0.0370158	10.27326
temperature	stl	2003-05-31	11.92	10.31073	1.812566	-0.2032953	10.10743

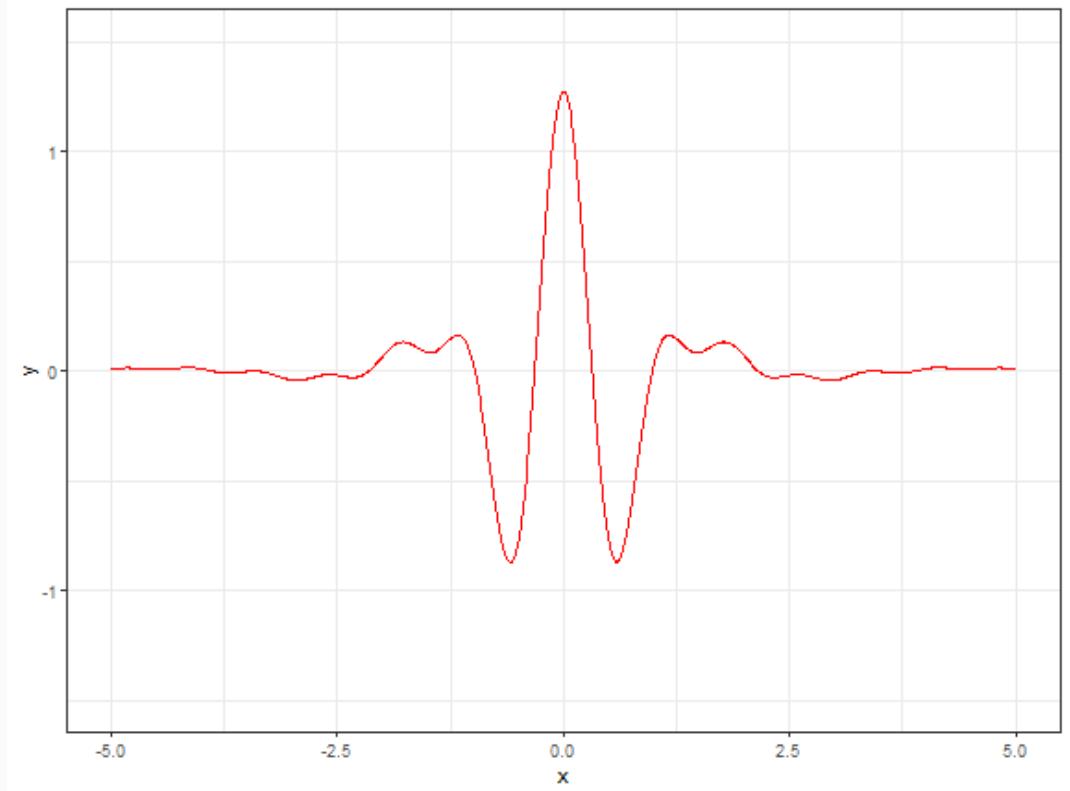


Extension of Fourier analysis

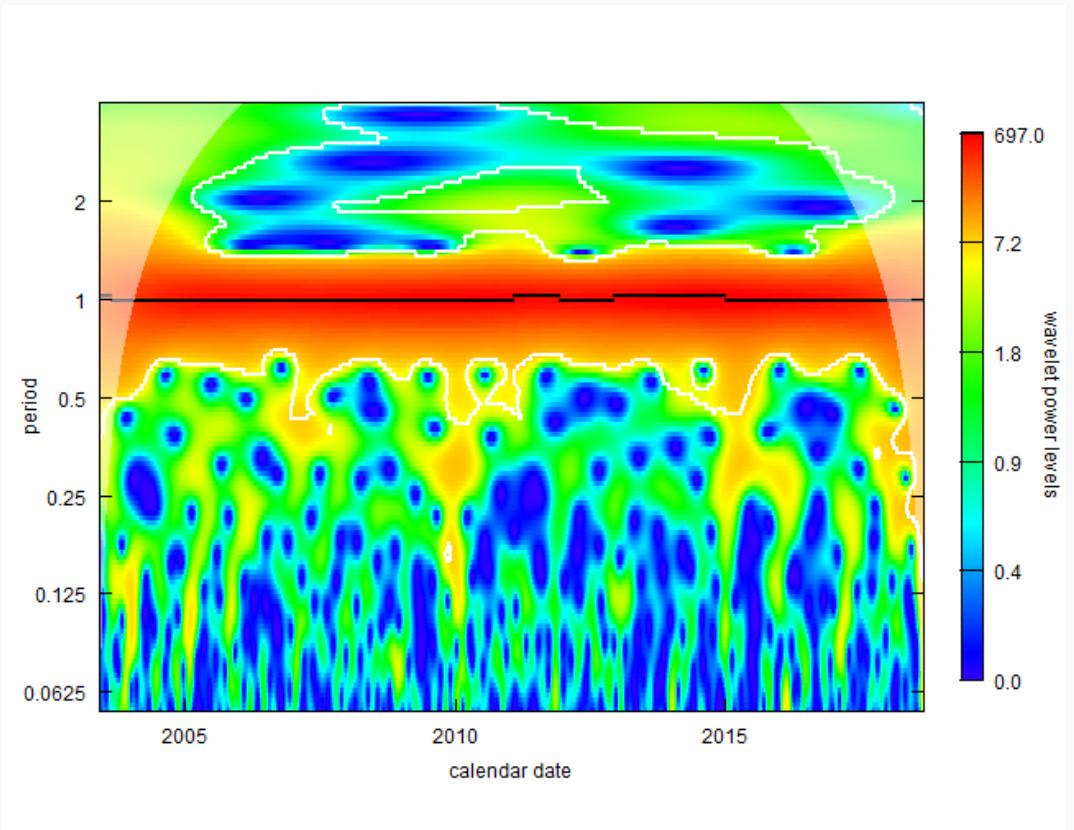
<https://en.wikipedia.org/wiki/Wavelet>

Package `WaveletComp`

[http://www.hs-
stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf](http://www.hs-stat.com/projects/WaveletComp/WaveletComp_guided_tour.pdf)

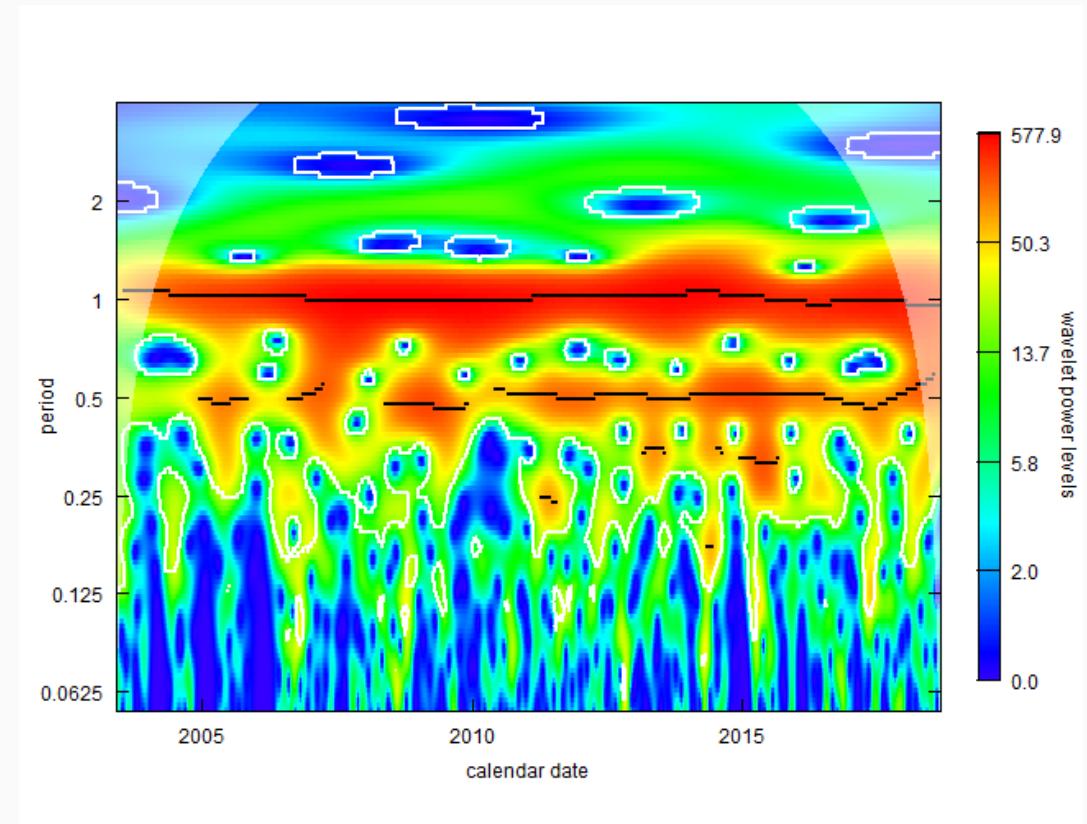


```
tsyn_wavelet ← tsyn_filled %>%
  filter(parameter = "temperature") %>%
  WaveletComp::analyze.wavelet( my.series = "value",
                                loess.span = 0,
                                dt = 1/365, dj = 1/20,
                                lowerPeriod = 1/18,
                                upperPeriod = 4,
                                make.pval = TRUE,
                                n.sim = 10)
```



Synechococcus

```
tsyn_wavelet ← tsyn_filled %>%
  filter(parameter = "syn_ml") %>%
  mutate(value = log10(value)) %>%
  tidyr::fill(value, .direction = "down") %>%
  WaveletComp::analyze.wavelet( my.series = "value",
                                loess.span = 0,
                                dt = 1/365, dj = 1/20,
                                lowerPeriod = 1/18,
                                upperPeriod = 4,
                                make.pval = TRUE,
                                n.sim = 10)
```



- Time series in environmental sciences
 - Manipulate dates
 - Create tsibble objects
 - Visualize time series
 - Smooth time series
 - Determine periodicity and trends
-
- Modelling
 - Forecasting