Boring but important disclaimers:

- If you are not getting this from the GitHub repository or the associated Canvas page (e.g. CourseHero, Chegg etc.), you are probably getting the substandard version of these slides Don't pay money for those, because you can get the most updated version for free at

  `https://github.com/julianmak/academic-notes`

  The repository principally contains the compiled products rather than the source for size reasons.
- Associated Python code (as Jupyter notebooks mostly) will be held on the same repository. The source data however might be big, so I am going to be naughty and possibly just refer you to where you might get the data if that is the case (e.g. JRA-55 data). I know I should make properly reproducible binders etc., but I didn't...
- I do not claim the compiled products and/or code are completely mistake free (e.g. I know I don't write Pythonic code). Use the material however you like, but use it at your own risk.
- As said on the repository, I have tried to honestly use content that is self made, open source or explicitly open for fair use, and citations should be there. If however you are the copyright holder and you want the material taken down, please flag up the issue accordingly and I will happily try and swap out the relevant material.

OCES 3301 :

basic Data Analysis in ocean sciences

Session 8: fairly basic time-series analysis

Outline

(Just overview here; for actual content see Jupyter notebooks)

- Interpolation and extrapolation

  $\rightarrow$ e.g. missing data, irregular data, un-collocated data

  $\rightarrow$ nearest neighbour, linear, cubic spline
- Application: tide gauge data

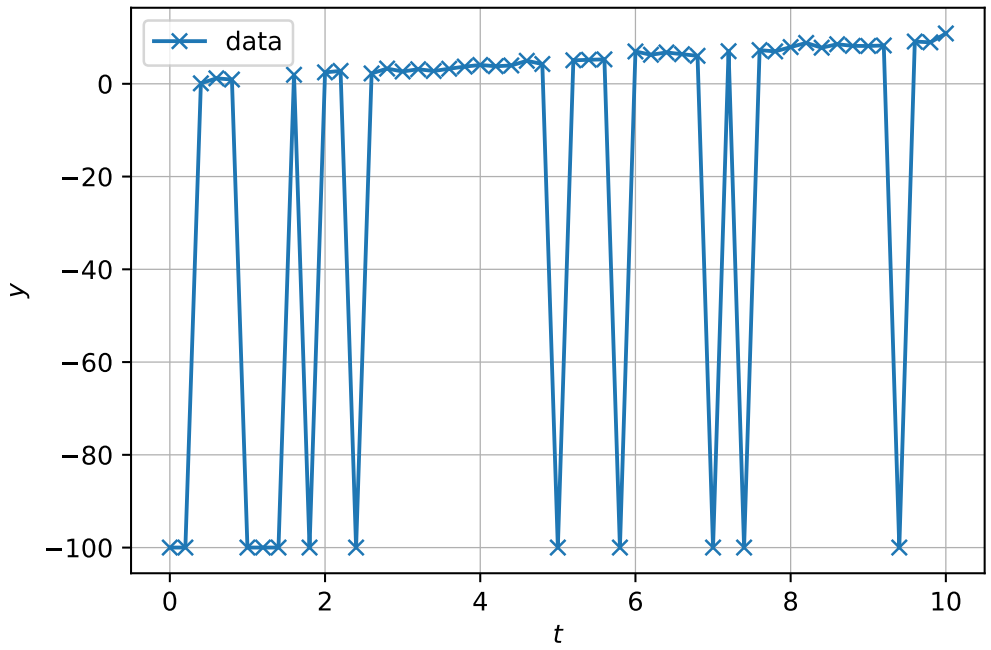
Recall: El-Niño 3.4 SST

1		1948		2019								
2	1948	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99
3	1949	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99	-99.99
4	1950	24.55	25.06	25.87	26.28	26.18	26.46	26.29	25.88	25.74	25.69	25.47	25.29
5	1951	25.24	25.71	26.90	27.58	27.92	27.73	27.60	27.02	27.23	27.20	27.25	26.91
6	1952	26.67	26.74	27.17	27.80	27.79	27.18	26.53	26.30	26.36	26.26	25.92	26.21
7	1953	26.74	27.00	27.57	28.04	28.28	28.12	27.43	26.94	27.01	26.87	26.88	27.00
8	1954	26.98	27.03	26.90	26.64	27.12	26.80	26.11	25.43	25.12	25.23	25.57	25.26
9	1955	25.61	25.81	26.22	26.60	26.66	26.55	26.15	25.51	25.28	24.41	24.25	24.57
10	1956	25.34	25.76	26.46	26.85	27.13	26.81	26.23	25.68	25.73	25.75	25.56	25.71
11	1957	26.04	26.54	27.46	28.23	28.55	28.36	28.17	27.69	27.44	27.42	27.62	27.90
12	1958	28.33	28.24	28.27	28.27	28.31	27.99	27.32	26.85	26.40	26.45	26.75	26.62
13	1959	27.07	27.18	27.47	27.88	27.70	27.37	26.44	26.09	25.92	26.24	26.04	26.18
14	1960	26.27	26.29	26.98	27.49	27.68	27.24	26.88	26.70	26.44	26.22	26.26	26.22
15	1961	26.23	26.56	26.94	27.36	27.75	27.67	26.89	26.19	25.78	25.71	26.07	25.97
16	1962	25.96	26.19	26.80	27.13	27.05	27.08	26.76	26.33	25.94	25.97	25.75	25.67
17	1963	25.77	26.22	27.18	27.78	27.63	27.62	27.78	27.48	27.40	27.36	27.47	27.62
18	1964	27.34	27.13	27.02	26.95	26.82	26.59	26.33	25.60	25.32	25.37	25.26	25.23
19	1965	25.66	26.19	26.94	27.38	27.99	28.09	27.90	27.97	28.01	28.17	28.12	27.96
20	1966	27.67	27.55	28.21	28.16	27.55	27.64	27.33	26.48	26.27	26.22	26.23	26.03
21	1967	25.88	26.11	26.50	26.74	27.35	27.47	26.97	26.44	25.86	25.97	26.08	25.95
22	1968	25.69	25.68	26.33	27.10	27.19	27.88	27.58	27.01	26.72	26.75	27.20	27.27
23	1969	27.50	27.86	27.82	28.13	28.29	27.69	27.08	27.02	27.15	27.34	27.10	26.98

Figure: Sample content of `elnino34_sst.data`.

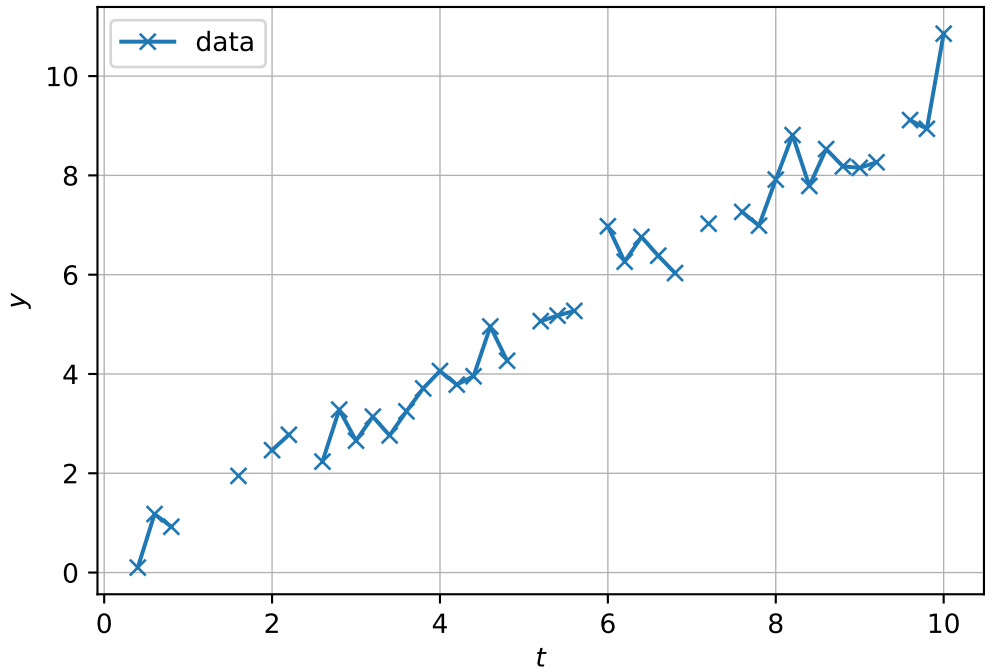
- the -99.99 denote masked/missing values

Sample plot with masked value



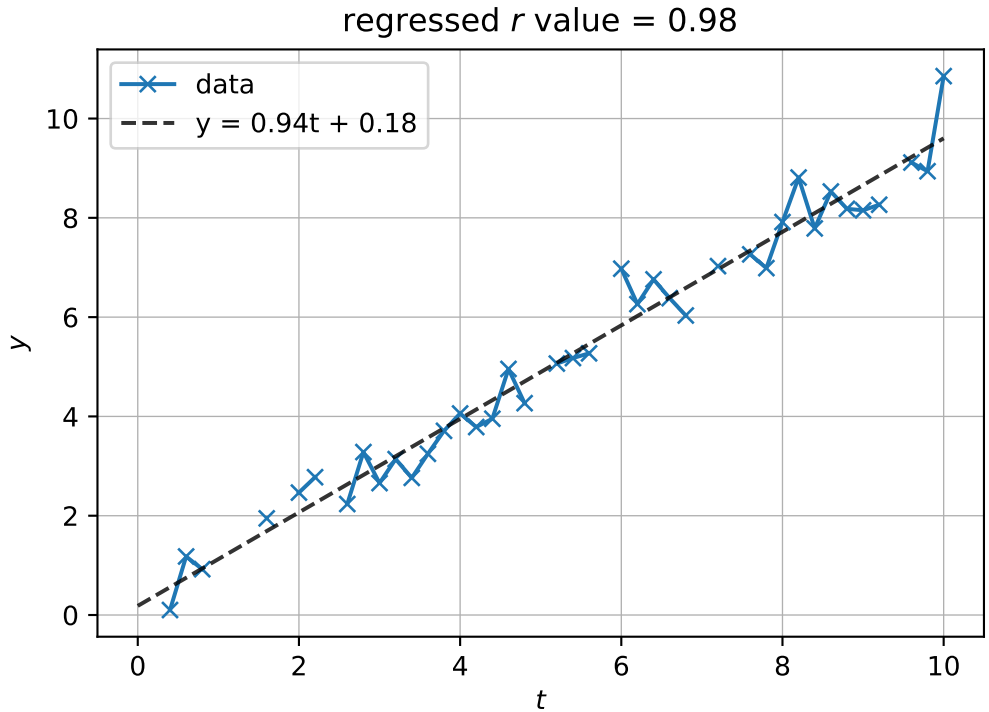
- including the masked values with screw up plots/calculations

Sample plot with NaN'ed value



- ...replacing with `NaN` usually fixes plots, but not calculations (e.g. regression routine will 'fail')

Sample plot with NaN'ed value

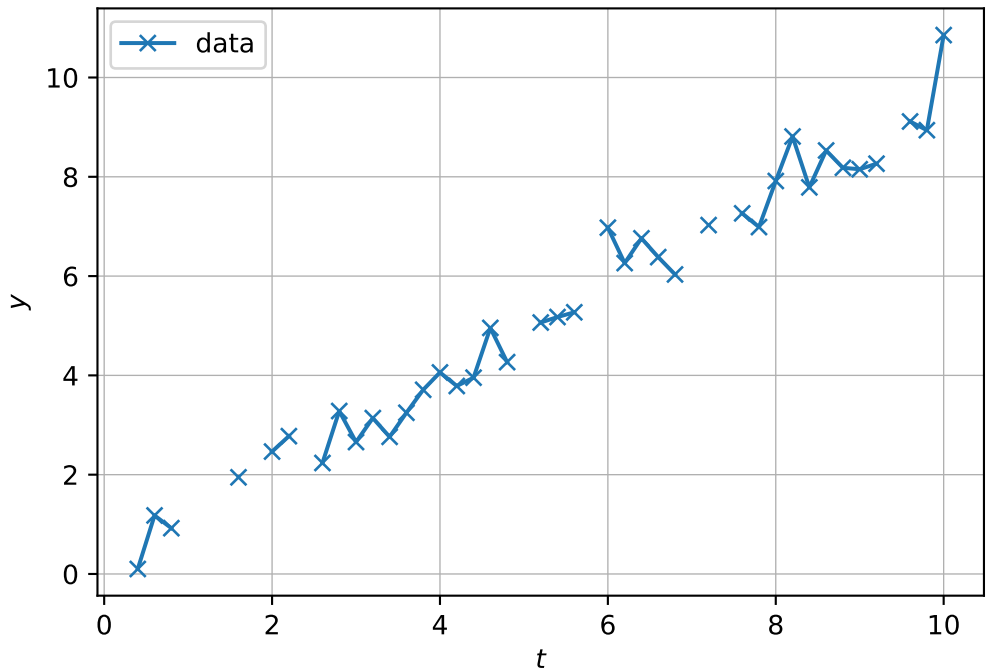


- just don't throw the NaNs in

  $\rightarrow$ find index corresponding to NaNs, pick out the not NaNs, and only throw those into the routine

  $\rightarrow$ recall `~` is NOT in python (from first notebook)

Interpolation



- could fill out the missing numbers too?

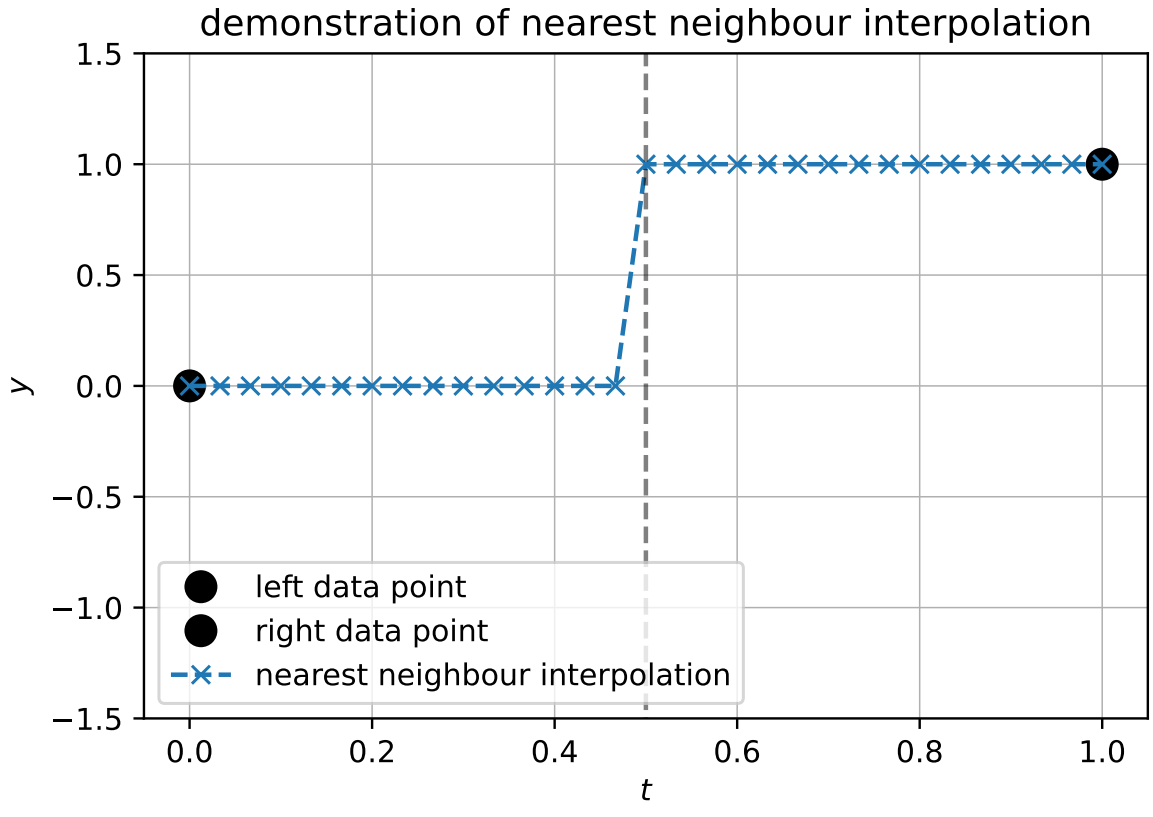
  $\rightarrow$ validity depends on context

Interpolation

- nearest neighbour

  $\rightarrow$ fills out data with point closest to it

  $\rightarrow$ depends on choice of **distance** (cf. choice of mismatch in regression)

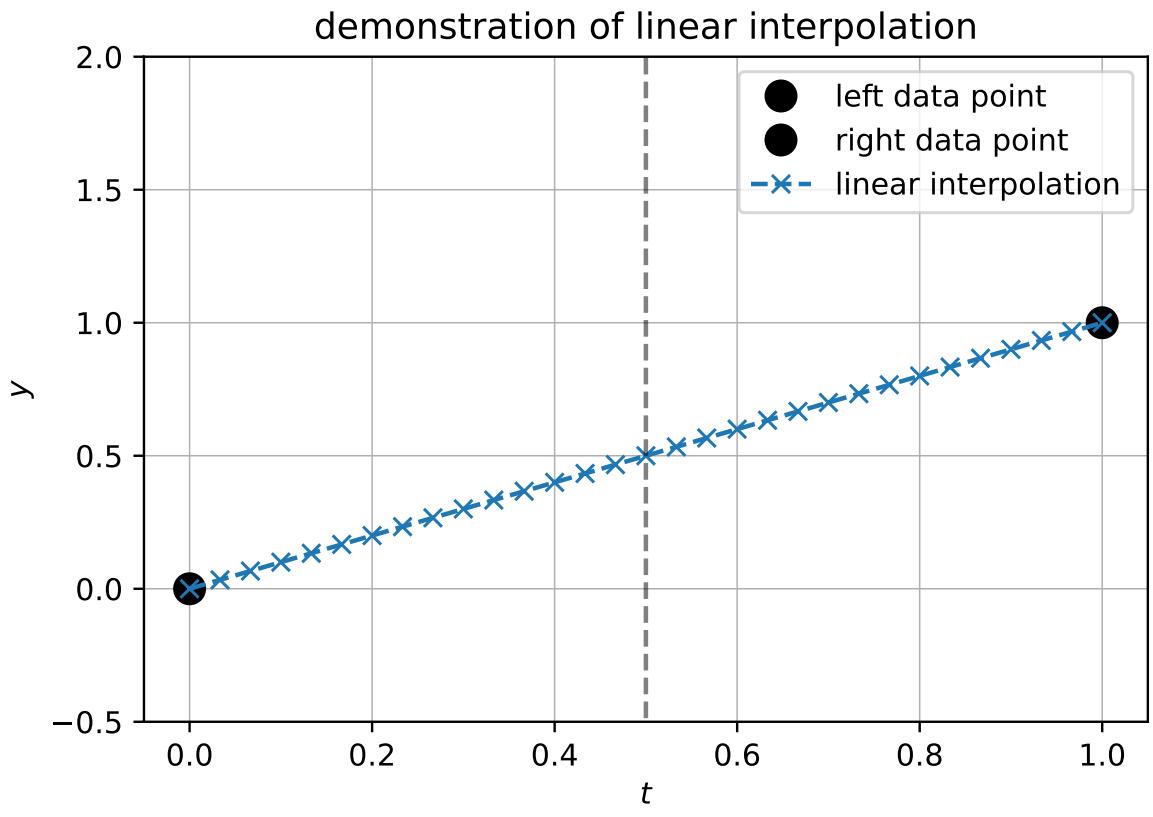


Interpolation

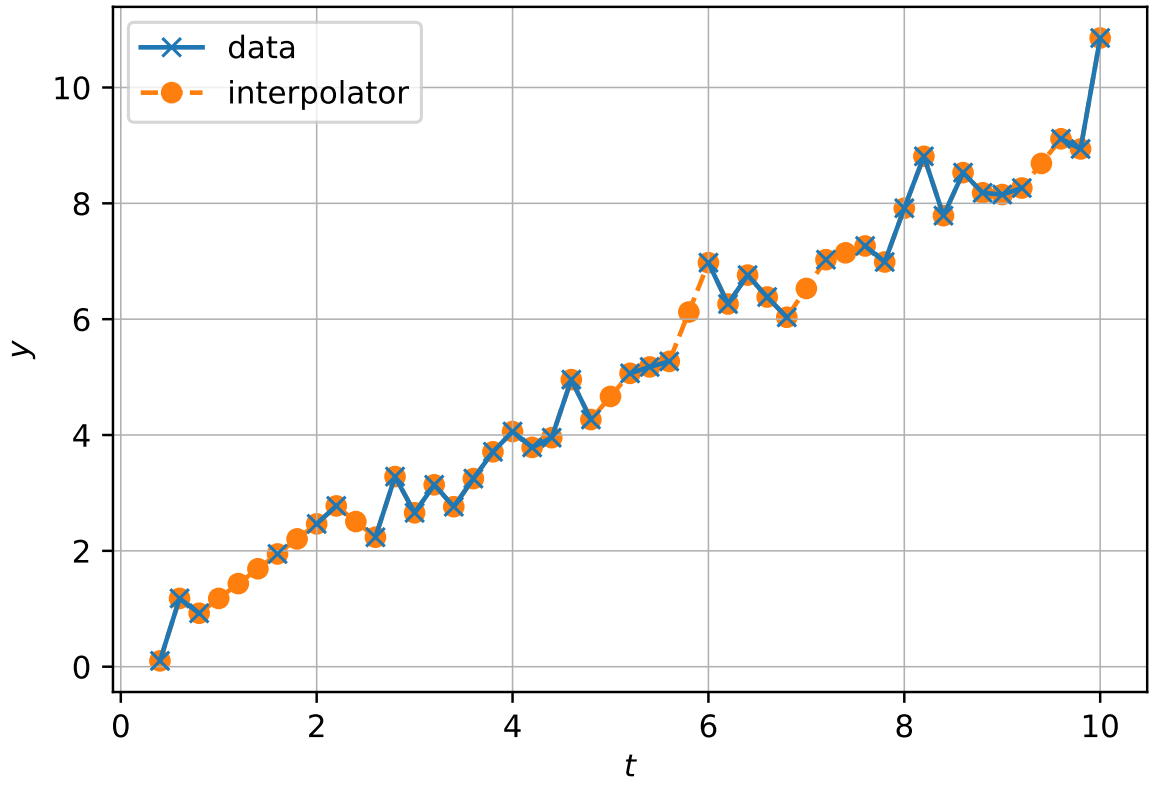
- linear interpolation

  $\rightarrow$ draw a straight line between two points and fill it out with the expected value

  $\rightarrow$ basically draw triangles, or use $y = mt + c$



Interpolation



- easy enough to do by hand, or use `scipy.interpolate.interp1d`

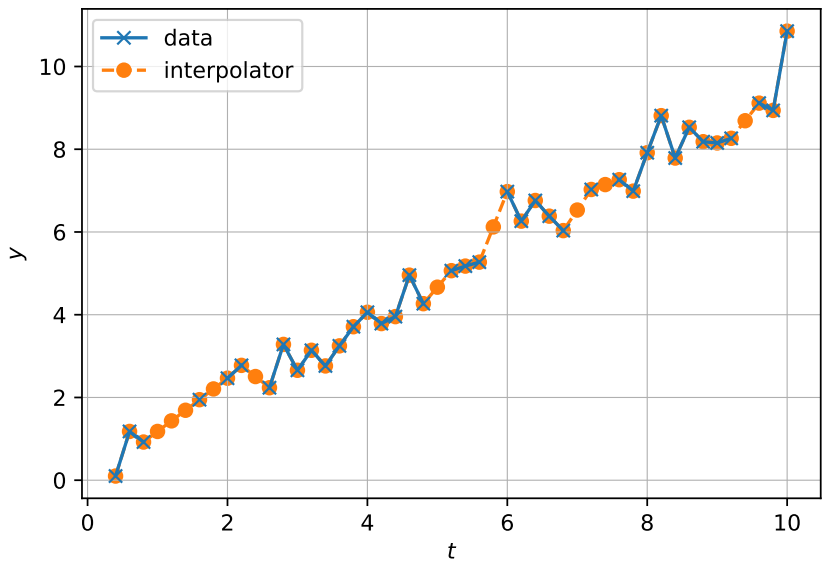
  → extrapolation can be done similarly, but see notebook for how it can break (for good reasons)

Interpolation

- piecewise linear interpolation a good default to try first

Q. can go beyond linear?

Q. can control the 'roughness'?

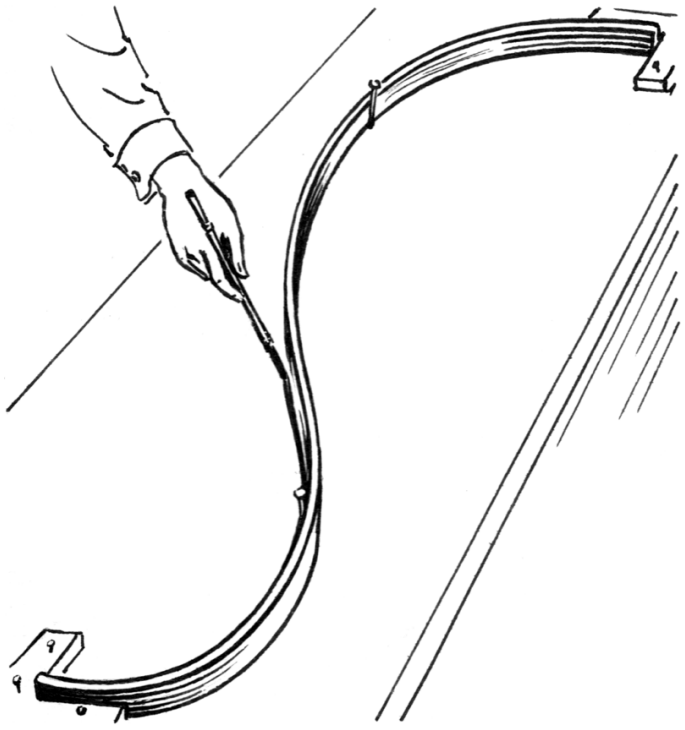


Interpolation

- spline

  → used for ship design etc.

  → pin a few places down, let it relax, draw the curve

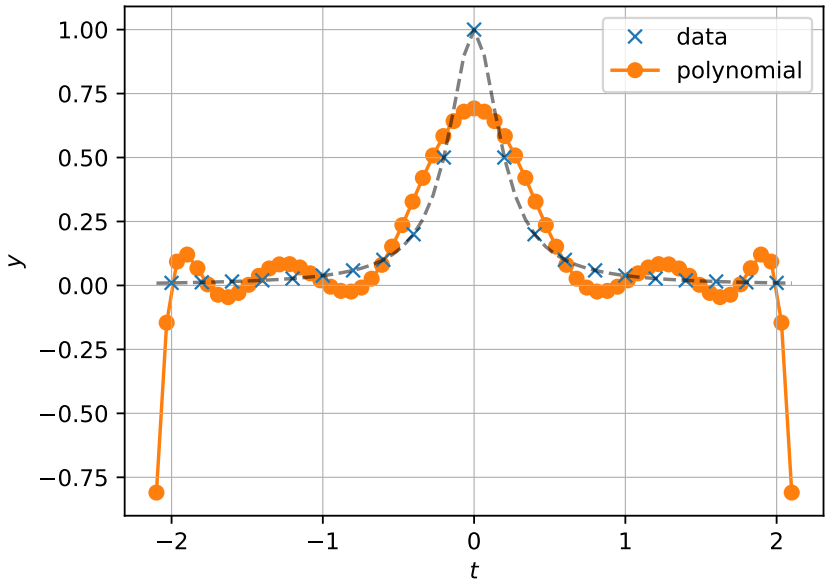


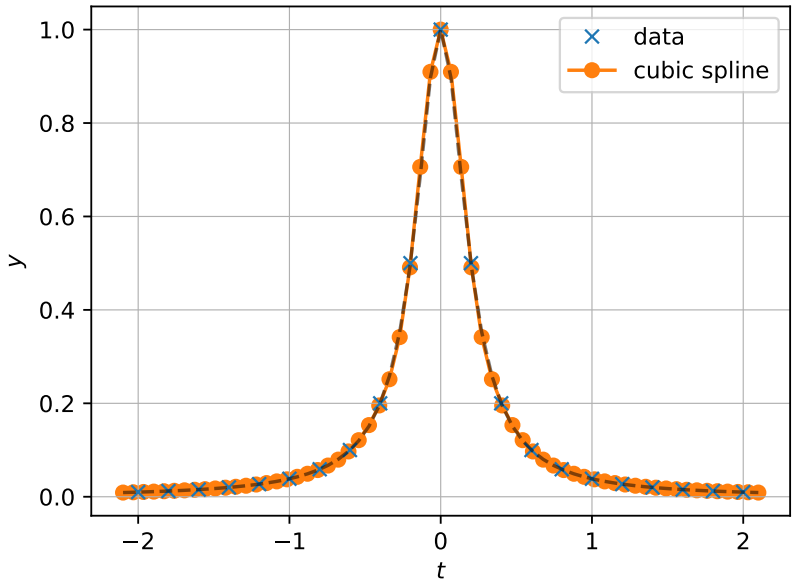
- "pinned place" = data points
- "relax" = constraints on curvature

- many of these, standard is (piecewise) **cubic spline**

  → result is differentiable (i.e. no kinks)

Interpolation





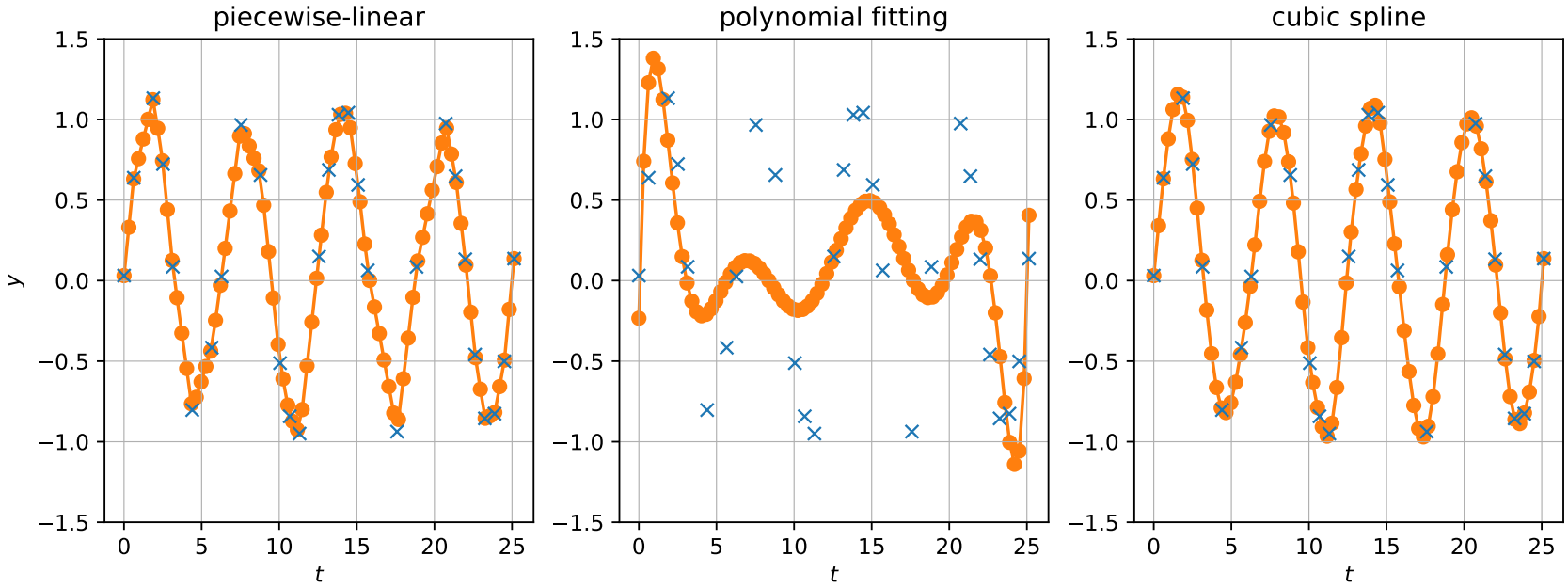
- example with sharp peak (witch of Agnessi)

  $\rightarrow$ high order polynomial (10?) struggles

  $\rightarrow$ cubic spline works pretty well

(former does global optimisation, latter is local since it is piecewise)

Interpolation



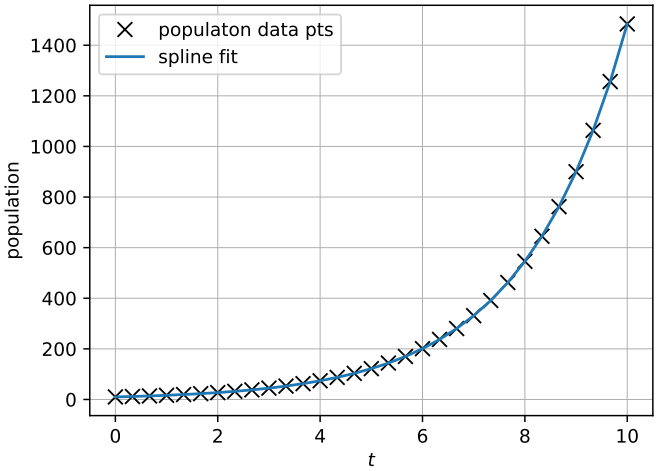
- oscillatory signal (with noise)

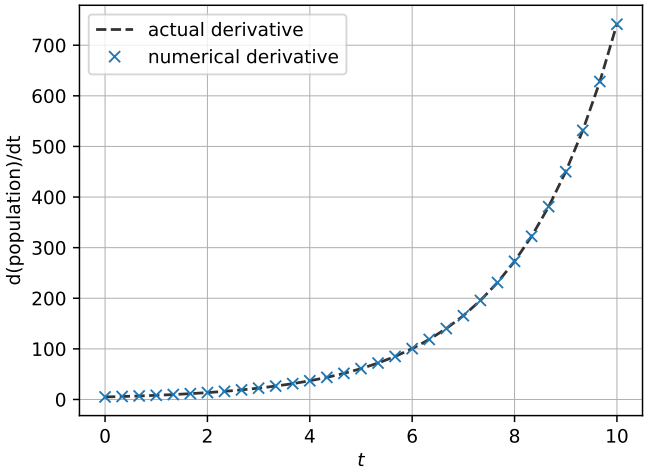
  → piecewise linear is non-smooth

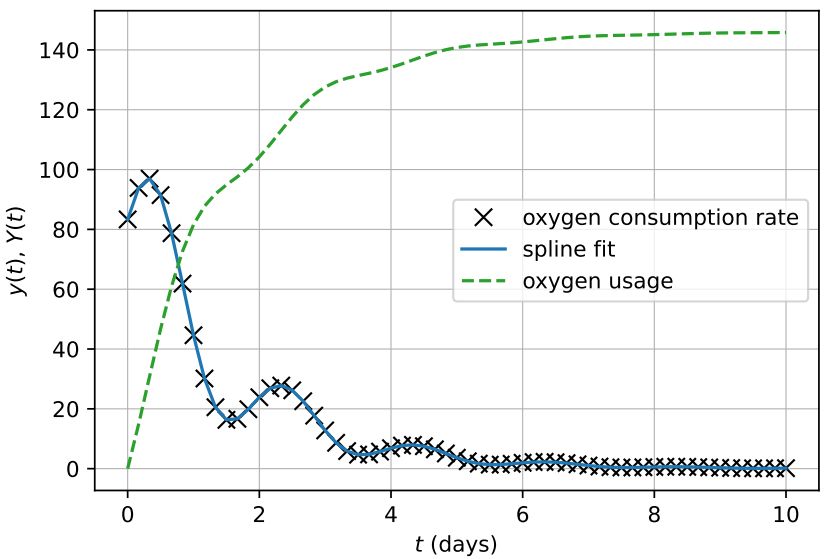
  → high order polynomial fitting struggles

  → cubic spline does pretty well

Interpolation







- smooth ⇒ can do derivatives and integrals
- ↑ exponential growth, derivative to get e-folding factor
- ← total oxygen consumption of model fish, as an integral

A real example: tides (see OCES 2003, lec 18)

Figure: High (or flood) and low (or ebb) tide at Tobermory, Isle of Mull, Scotland, using the pastel pink and red house as references. Modified images from `www.thechaoticscot.com` (left) and from myself (right).

A real example: tides (see OCES 2003, lec 18)

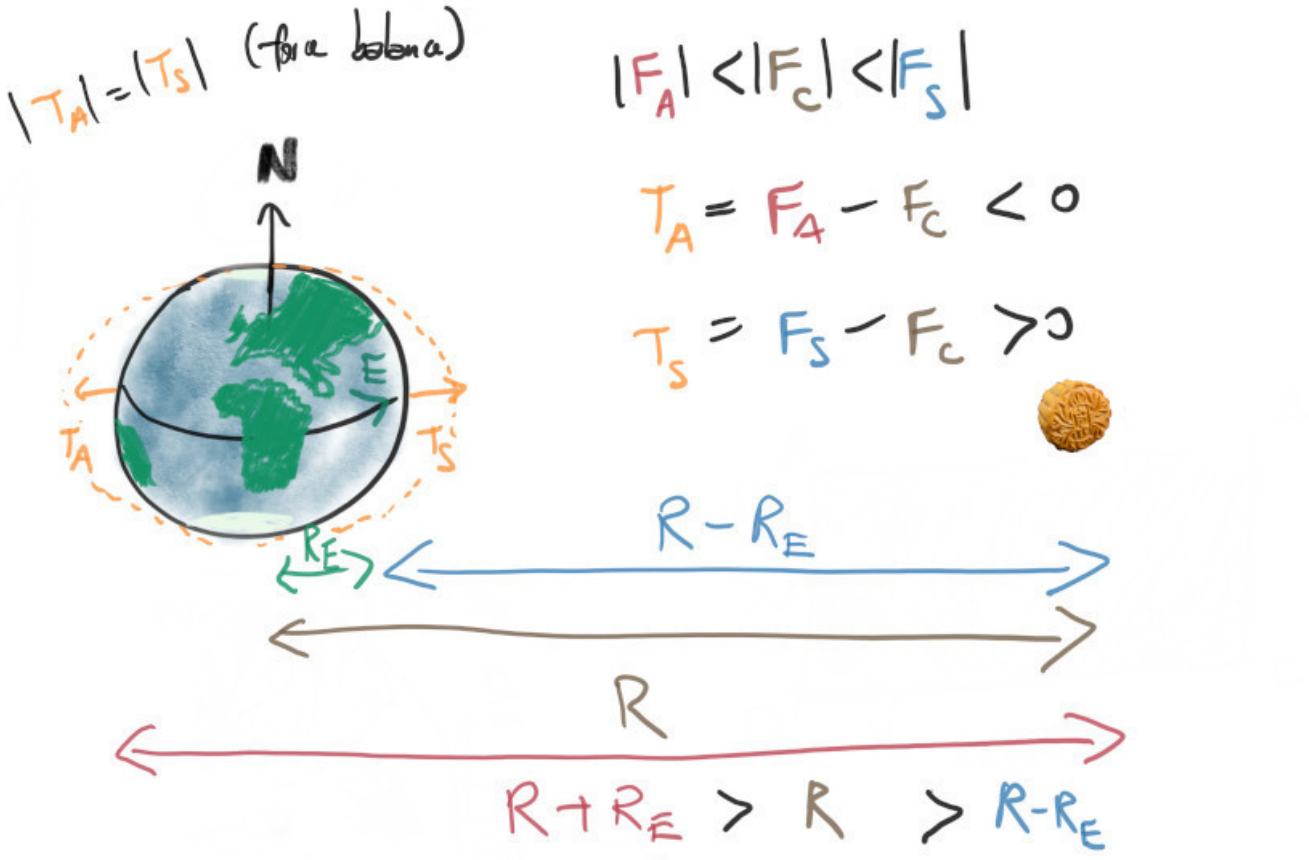


Figure: Schematic of tidal forcing by an astronomical body. Assume instantaneous response ("equilibrium theory"). No rotation is assumed here.

A real example: tides (see OCES 2003, lec 18)

symbol	period (in solar hrs)	rel. amp (to M_2)	name
M_2	12.42	1	principal lunar (semi-diurnal)
K_1	23.93	0.58	luni-solar (diurnal)
S_2	12.00	0.47	principal solar (semi-diurnal)
O_1	25.82	0.42	principal lunar (diurnal)
N_2	12.66	0.19	larger lunar elliptic (semi-diurnal)
⋮	⋮	⋮	⋮
Mf	327.85 ($\approx$ 14 days)	0.09	lunar fortnightly
Mm	661.30 ($\approx$ 28 days)	0.05	lunar monthly
SSa	4382.86	0.04	solar semi-annual

Table: Some sample tidal forcings sorted by relative amplitude to the M_2 tide (which is the largest forcing for Earth). Subset of Table 6.2 given in Wunsch (2015). The last few entries are weak and long term but they are there.

- M_2 and K_1 the dominant ones

  $\rightarrow$ usually do include these two in numerical models
- notice the periods are close to multiples of 12 hours

A real example: tides (see OCES 2003, lec 18)

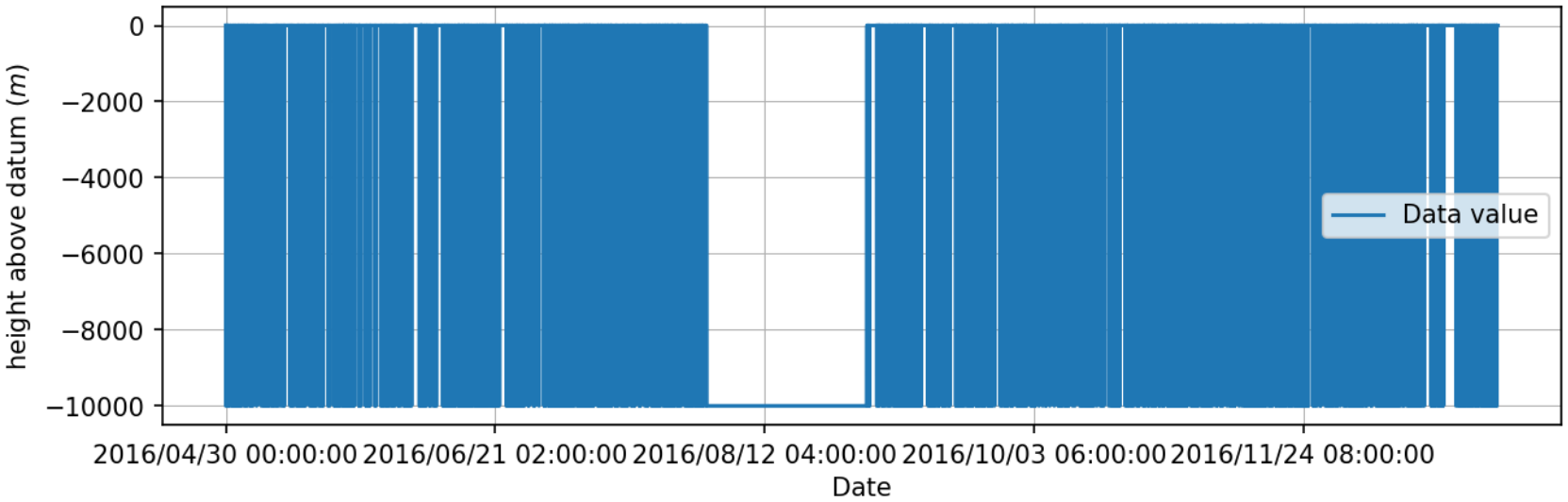


Figure: Data segment from BODC of sea level above datum at Tobermory.

- broken data and unformatted

  $\rightarrow$ plot of data with masked value left as is

A real example: tides (see OCES 2003, lec 18)

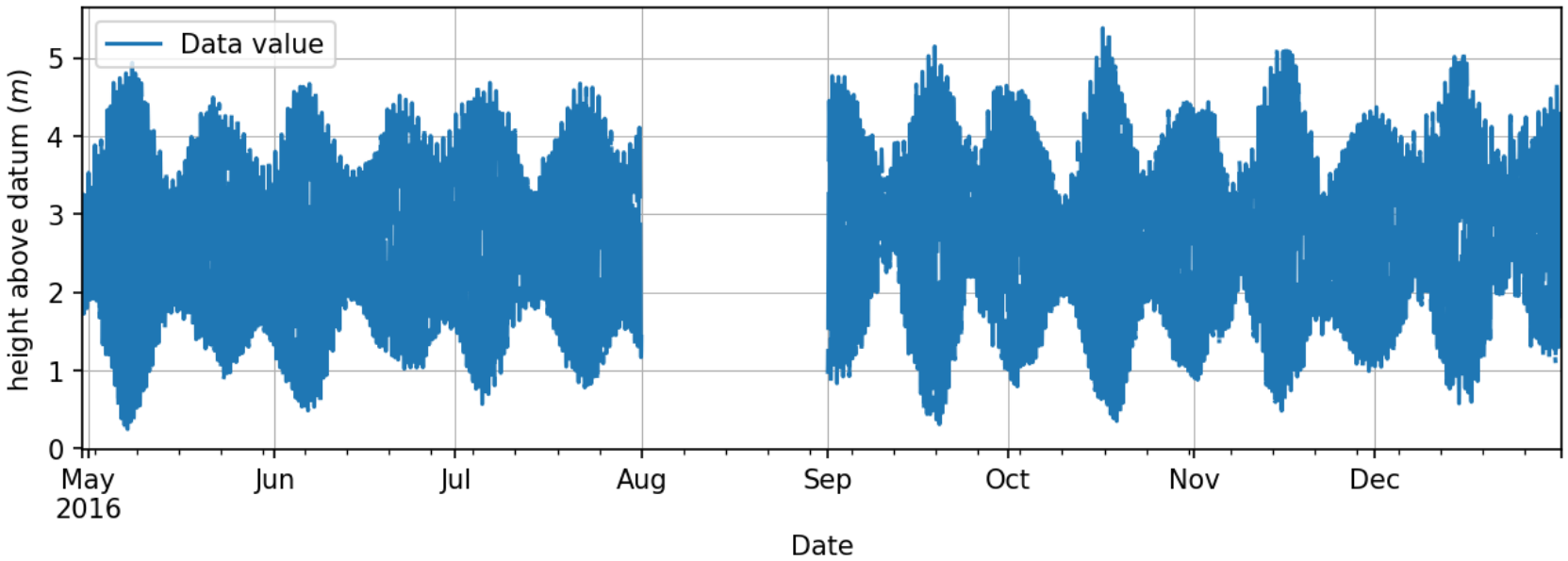


Figure: Data segment from BODC of sea level above datum at Tobermory.

- NaN out the missing data and better formatted (see notebook)
  → some spots missing, with a big chunk of data missing

A real example: tides (see OCES 2003, lec 18)

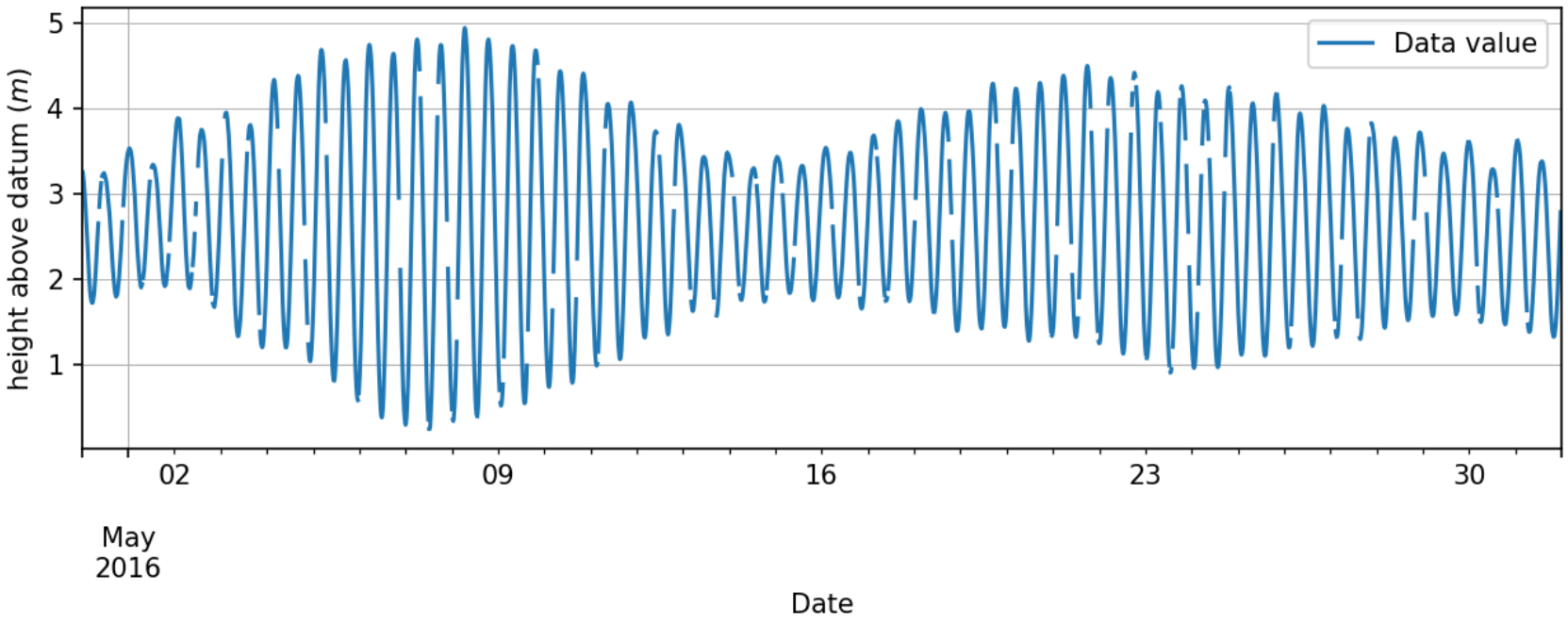


Figure: Data segment from BODC of sea level above datum at Tobermory.

- zoomed into month of May

  → some spots missing, probably ok to fill those out

A real example: tides (see OCES 2003, lec 18)

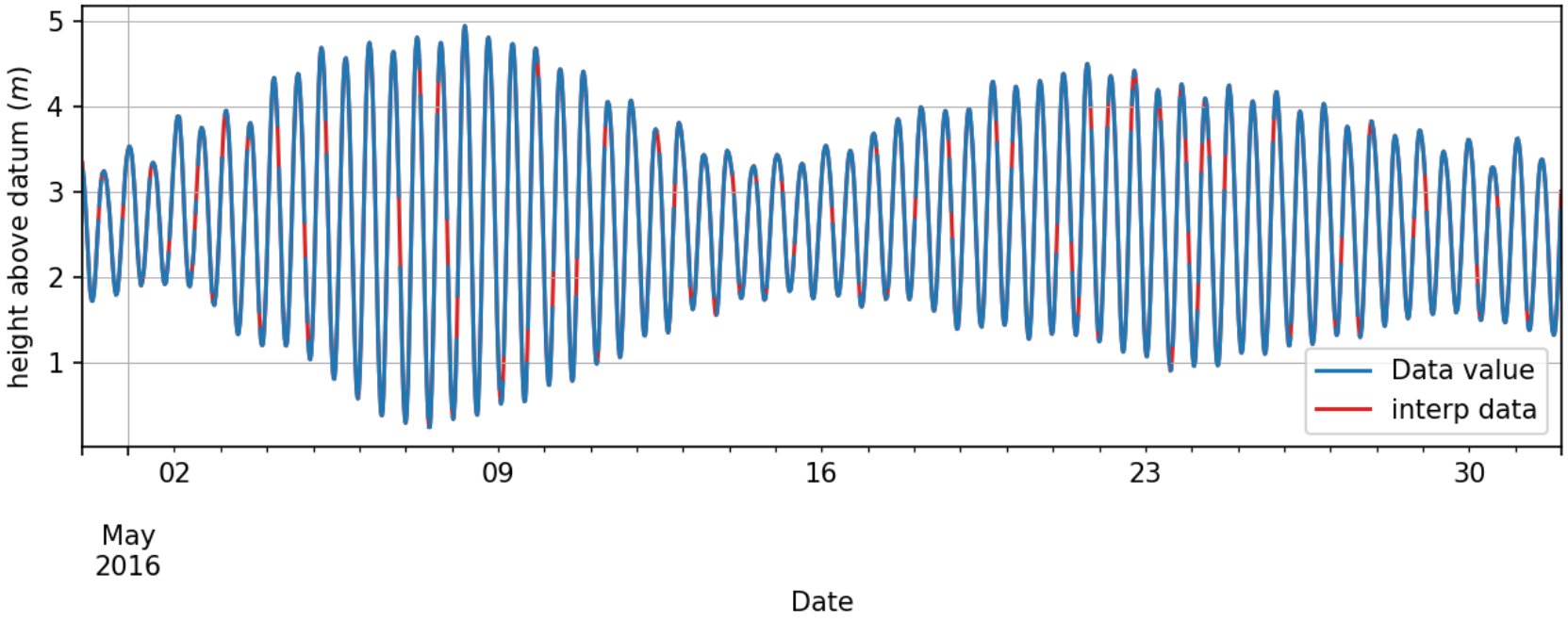


Figure: Data segment from BODC of sea level above datum at Tobermory.

- filling out with cubic spline

  $\rightarrow$ smoother than linear interpolation, probably better choice since data is clearly oscillatory

A real example: tides (see OCES 2003, lec 18)

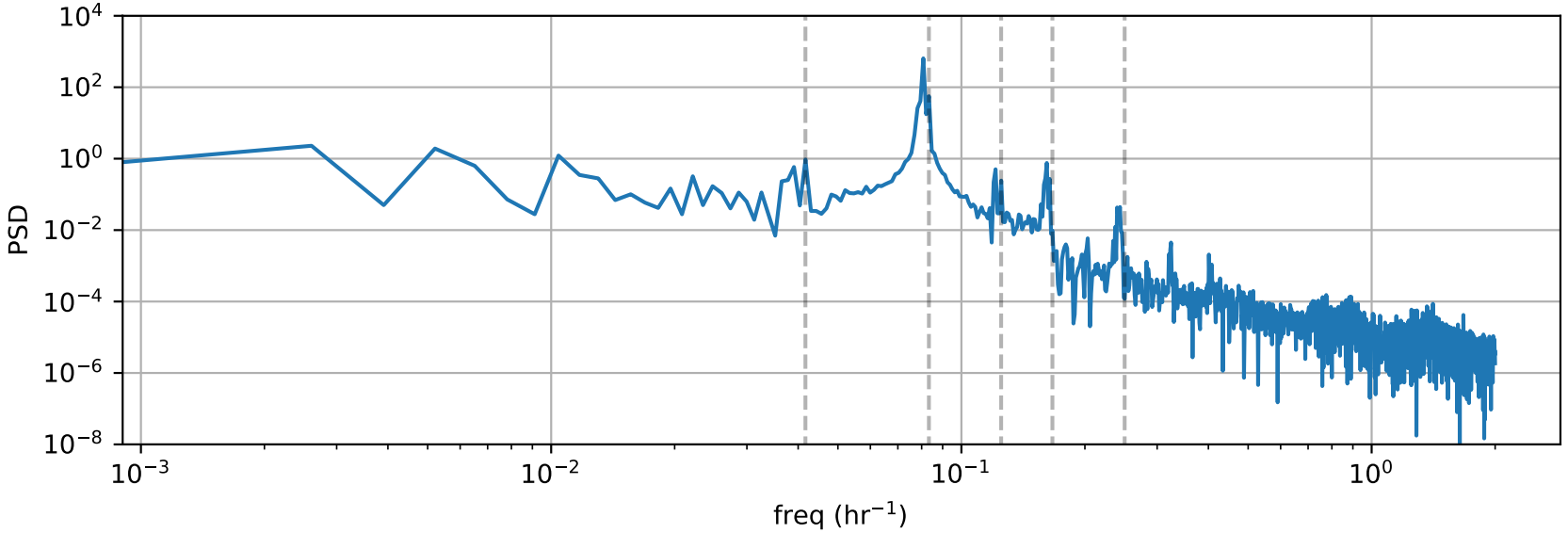


Figure: Data segment from BODC of sea level above datum at Tobermory.

- whole series used to generate a PSD (being careful with units etc.)

  $\rightarrow$ peaks can be shown to correspond to various tidal harmonics

Jupyter notebook

go to 08 Jupyter notebook to get some code practise

- try something similar for the El-Niño 3.4 data

  $\rightarrow$ be careful of trends

  $\rightarrow$ be careful of units (time units is in **years**)

Note: none of the content I introduced in 'times series' are exclusive to 'time', and works just as well for 'space' too