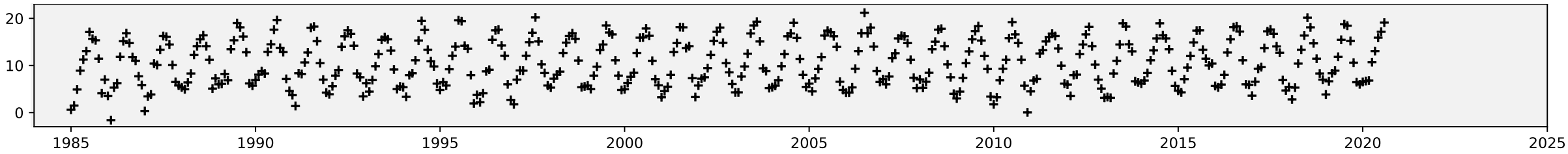
§2.1 Fitting a linear model

Monthly average temperatures in Cambridge, UK

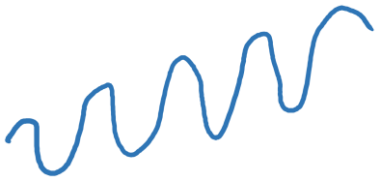
What's a good model for this dataset?



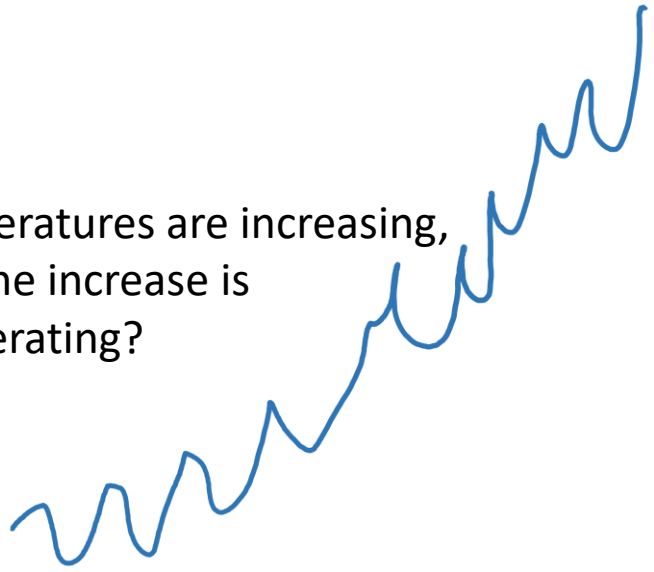
Climate is stable?

$$\text{Temp}(t) \sim a + b\sin\big(2\pi(t+\phi)\big) + N(0,\sigma^2)$$

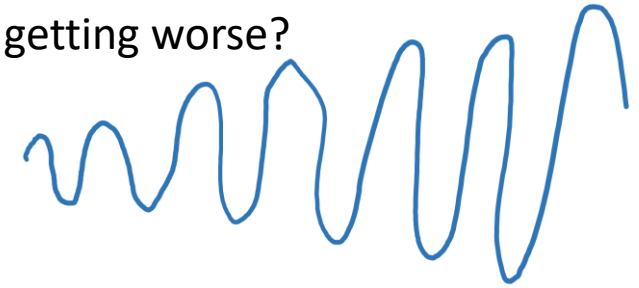
Temperatures are increasing?



Temperatures are increasing, and the increase is accelerating?



The extremes are getting worse?



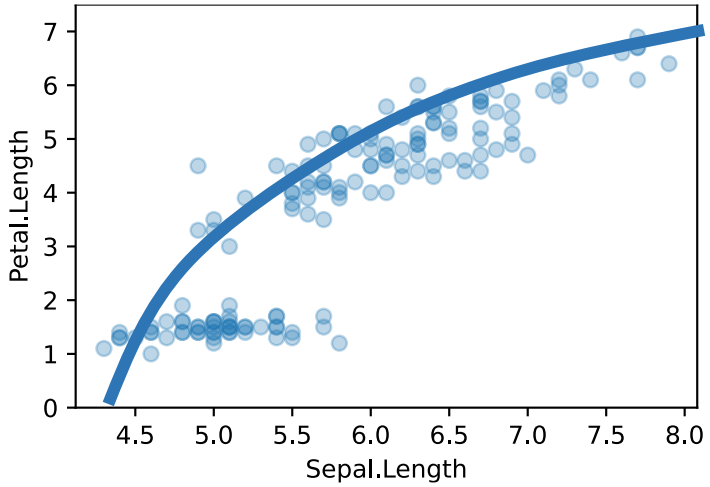
There are so many possible models. We want to make it easy to invent and fit new models, so we have time to explore all the possibilities.

Example 2.1.1

The Iris dataset has 50 records of iris measurements, from three species.

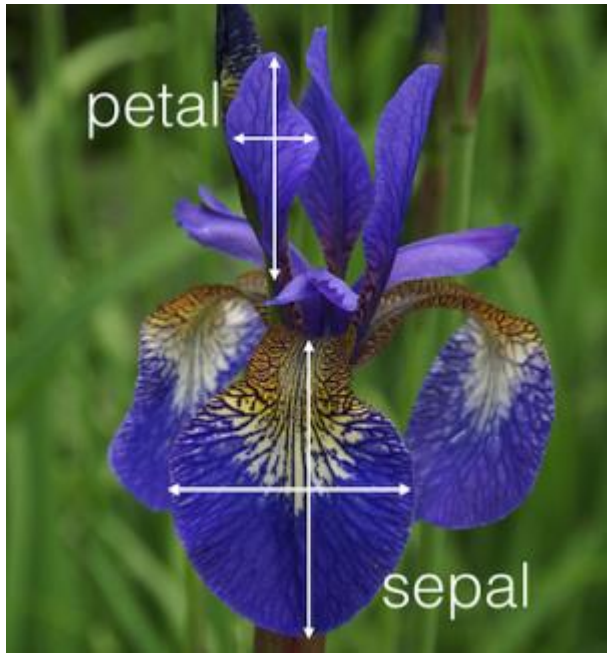
Petal. Length	Petal. Width	Sepal. Length	Sepal. Width	Species
1.0	0.2	4.6	3.6	setosa
5.0	1.9	6.3	2.5	virginica
5.8	1.6	7.2	3.0	virginica
4.2	1.2	5.7	3.0	versicolor
…				

How does `Petal.Length` depend on `Sepal.Length`?



Let's guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),

$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$



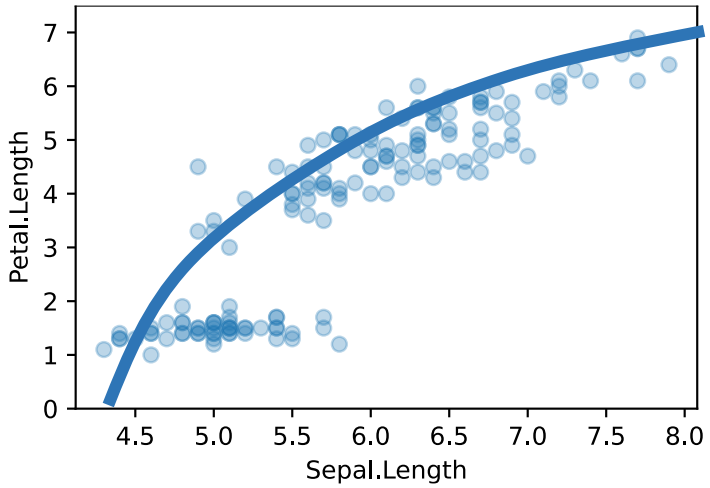
Dataset collected by Edgar Anderson and popularized by Ronald Fisher in 1936

Example 2.1.1

The Iris dataset has 50 records of iris measurements, from three species.

Petal. Length	Petal. Width	Sepal. Length	Sepal. Width	Species
1.0	0.2	4.6	3.6	setosa
5.0	1.9	6.3	2.5	virginica
5.8	1.6	7.2	3.0	virginica
4.2	1.2	5.7	3.0	versicolor
…				

How does `Petal.Length` depend on `Sepal.Length`?



Let's guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),

$\texttt{Petal.Length} \approx \alpha + \beta\,\texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$

unknown parameters to be estimated

A linear model

$$\begin{bmatrix} PL_1 \\ PL_2 \\ \vdots \\ PL_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} SL_1 \\ SL_2 \\ \vdots \\ SL_n \end{bmatrix} + \gamma \begin{bmatrix} (SL_1)^2 \\ (SL_2)^2 \\ \vdots \\ (SL_n)^2 \end{bmatrix}$$

response vector

feature vectors

Not a linear model

$$\text{Temp} \approx \alpha + \beta \sin(2\pi(t+\phi)) + \gamma t$$

$\alpha, \beta, \phi, \gamma$ unknown.

This ϕ is inside $\sin(\cdot)$, so it's not a linear model.

Dataset collected by Edgar Anderson and popularized by Ronald Fisher in 1936

Let's guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),
$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$

A linear model is

- a supervised-learning model
  [it has a response variable, and feature variables]
- in which the response and the features are numeric
- and the response vector is predicted by a linear combination of (known) feature vectors weighted by (unknown) parameters

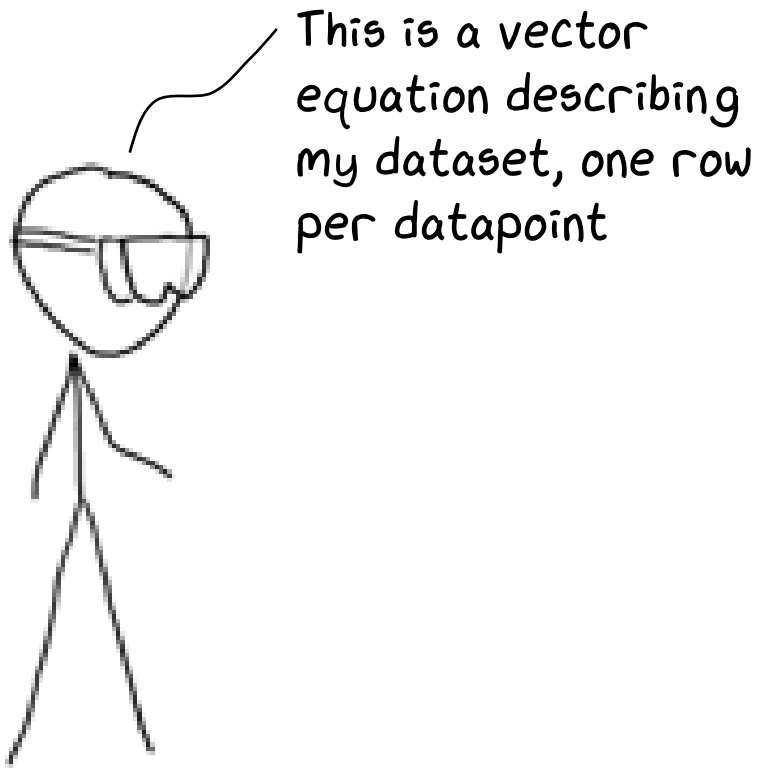
$$\begin{bmatrix} \mathrm{PL}_1 \\ \mathrm{PL}_2 \\ \vdots \\ \mathrm{PL}_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} \mathrm{SL}_1 \\ \mathrm{SL}_2 \\ \vdots \\ \mathrm{SL}_n \end{bmatrix} + \gamma \begin{bmatrix} (\mathrm{SL}_1)^2 \\ (\mathrm{SL}_2)^2 \\ \vdots \\ (\mathrm{SL}_n)^2 \end{bmatrix}$$

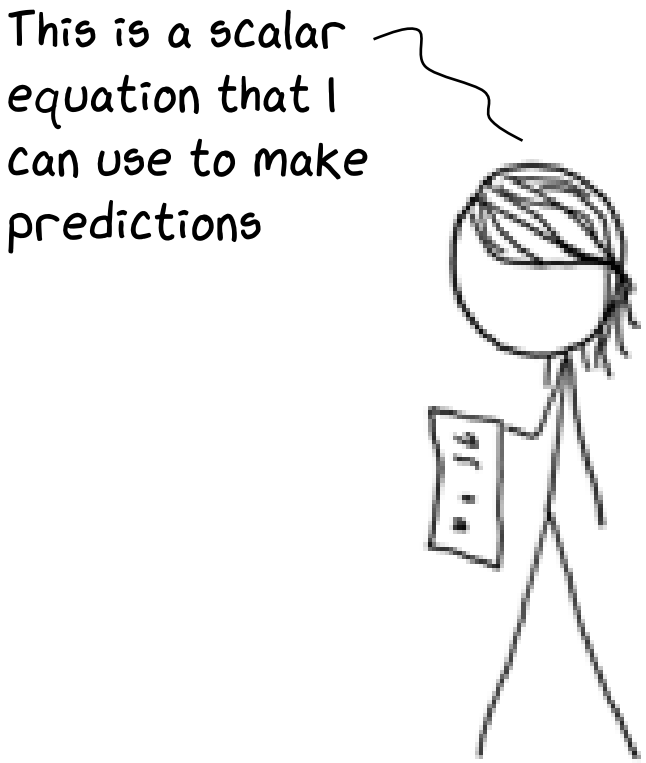
Models of this form are called *linear models* (because they're based on linear algebra).

They are flexible, and very fast to optimize.

NOTATION WARNING

$$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$$





Least squares estimation

Consider a linear model

$$y \approx \beta_1 e_1 + \cdots + \beta_K e_K$$

"All models are wrong."

The vector of prediction errors is called the *residual vector*,

$$\varepsilon = y - (\beta_1 e_1 + \cdots + \beta_K e_K)$$

We can fit the model using *least squares estimation*. This means finding parameters $\beta_1, \ldots, \beta_K$ to minimize the mean square error

$$\text{mse} = \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2$$

```
1  iris = pandas.read_csv(...)
```

$$\texttt{Petal.Length} \approx \alpha + \beta\ \texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$$

$$\begin{bmatrix} \text{PL}_1 \\ \text{PL}_2 \\ \vdots \\ \text{PL}_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} \text{SL}_1 \\ \text{SL}_2 \\ \vdots \\ \text{SL}_n \end{bmatrix} + \gamma \begin{bmatrix} (\text{SL}_1)^2 \\ (\text{SL}_2)^2 \\ \vdots \\ (\text{SL}_n)^2 \end{bmatrix}$$

Fitting the model

```
2  one, SL, PL = np.ones(len(iris)), iris['Sepal.Length'], iris['Petal.Length']
3  model = sklearn.linear_model.LinearRegression(fit_intercept=False)
4  model.fit(np.column_stack([one, SL, SL**2]), PL)
5  (α,β,γ) = model.coef_
```

Making predictions / getting fitted values from the model

```
6  newSL = np.linspace(4.2, 8.2, 20)
7  predPL = α + β*newSL + γ*(newSL**2)
```

```
1  iris = pandas.read_csv(...)
```

$$\text{Petal.Length} \approx \alpha + \beta\ \text{Sepal.Length} + \gamma(\text{Sepal.Length})^2$$

$$\begin{bmatrix} PL_1 \\ PL_2 \\ \vdots \\ PL_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} SL_1 \\ SL_2 \\ \vdots \\ SL_n \end{bmatrix} + \gamma \begin{bmatrix} (SL_1)^2 \\ (SL_2)^2 \\ \vdots \\ (SL_n)^2 \end{bmatrix}$$

Fitting the model (cleaner code)

```
2  SL, PL = iris['Sepal.Length'], iris['Petal.Length']
3  model = sklearn.linear_model.LinearRegression()
4  model.fit(np.column_stack([SL, SL**2]), PL)
5  α,(β,γ) = model2.intercept_, model2.coef_
```

`sklearn.linear_model` puts in the `[1,...,1]` feature for us by default. If we don't want it, we have to specify `fit_intercept=False`.

Making predictions / getting fitted values from the model (cleaner code)

```
6  newSL = np.linspace(4.2, 8.2, 20)
7  predPL = model.predict(np.column_stack([newSL, newSL**2]))
```

This saves us from having to explicitly code up the prediction formula – less chance we introduce bugs

Exampl
Learning with p
Data Science—I

Some of the questions ask for pseudocode. I sugg
online tester. There is a notebook with templat
the course materials webpage.

Question 1. Given a dataset $[x_1, \ldots, x_n]$, we w
random variable with a single parameter $\lambda > 0$,

$$\Pr(x\,;\,\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Show that the maximum likelihood estimator fo

Question 2. Give pseudocode to fit the mod
[Optional.] If you want to test your code using
`PoissonModel`.

In practice it'd be daft to use numerical opt
answer. But it's good to get used to numerical
problem where you what the answer should be.

Question 3. Given a dataset $[x_1, \ldots, x_n]$, we
is unknown. Show that the maximum likelihood

ex1: practical exercises for Example Sheet 1.

The example sheet asks you to implement the three classes given below: `PoissonModel`, `PiecewiseLinearModel`, and `StepPeriodModel`. The class skeletons are given, and you should fill in the missing pieces. To test your answers on Moodle, please upload either a Jupyter notebook called `ex1.ipynb` or a plain Python file called `ex1.py`.

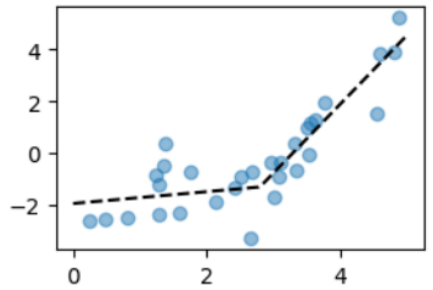
]:
```
import numpy as np
import scipy.optimize
import sklearn.linear_model
```

Poisson model. Suppose we're given a dataset $[x_1, \ldots, x_n]$. We wish to fit the model that says each x_i is an independent sample from the Poisson(λ) distribution. Estimate λ using `scipy.optimize.fmin`.

Note. If the tester reports that your answer is a little bit off, try increasing the precision that `scipy.optimize.fmin` is using by e.g. passing in the argument `xtol=0.00001`.

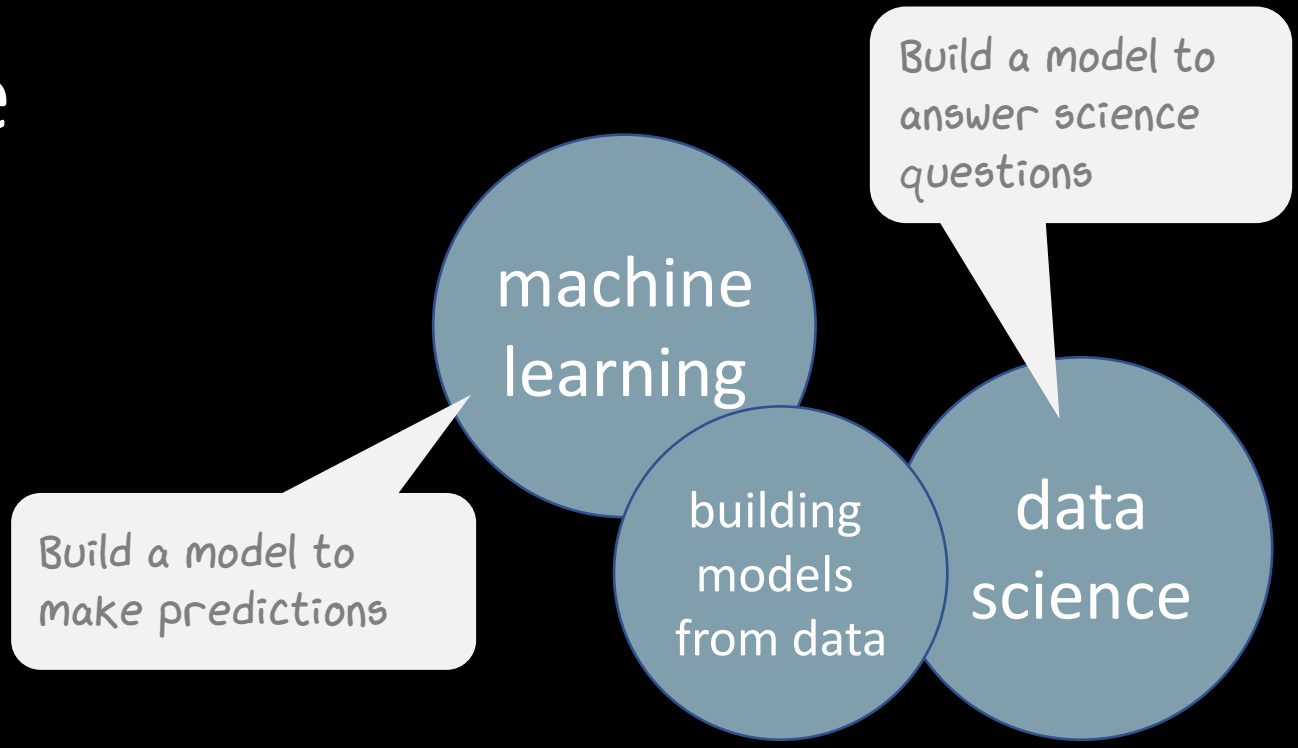
```
class PoissonModel():
    def __init__(self):
        self.λ_ = np.nan
    def fit(self, x):
        # Input: x is a numpy vector of integers
        # TODO: set self.λ_
```

Piecewise linear response. Suppose we're given a dataset of (x_i, y_i) pairs. We wish to fit a model for y as a function of x, made up of two straight lines. The function must be continuous, i.e. the two straight lines must meet at an inflection point. The x-coordinate of the inflection point is given.



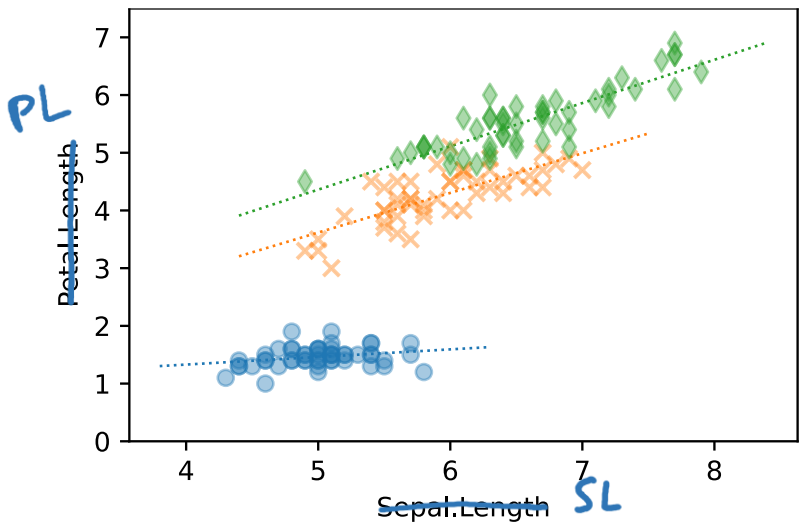
§2.2 Feature design

How do we design features, so that linear models answer the questions we want answered?



ONE-HOT CODING

EXERCISE. Fit three straight lines, one per species.



we want to fit $PL \approx \alpha_{species} + \beta_{species} \times SL$

This has 6 unknown parameters to be estimated:

α_{seto}, β_{seto}, α_{vers}, β_{vers}, α_{virg}, β_{virg}.

So, in linear-model form, there must be 6 features vectors one per unknown parameter.

Row 1: $PL_1 \approx \alpha_{seto} + \beta_{seto} SL_1$

Row 2: $PL_2 \approx \alpha_{virg} + \beta_{virg} SL_2$

Row 3: $PL_3 \approx \alpha_{virg} + \beta_{virg} SL_3$ etc.

$$\begin{matrix} \text{seto} \\ \text{virg} \\ \text{virg} \\ \text{seto} \\ \text{vers} \\ \vdots \end{matrix} \begin{bmatrix} PL_1 \\ PL_2 \\ PL_3 \\ PL_4 \\ PL_5 \\ \vdots \end{bmatrix} \approx \alpha_{\text{seto}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \alpha_{\text{virg}} \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \end{bmatrix} + \alpha_{\text{vers}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \beta_{\text{seto}} \begin{bmatrix} SL_1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \beta_{\text{virg}} \begin{bmatrix} 0 \\ SL_2 \\ SL_3 \\ \vdots \end{bmatrix} + \beta_{\text{vers}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

$$\vec{PL} \approx \alpha_{seto} \vec{1}_{spec=seto} + \alpha_{virg} \vec{1}_{spec=virg} + \alpha_{vers} \vec{1}_{spec=vers} + \beta_{seto} (SL \times \vec{1}_{spec=seto}) + \cdots$$

I’m using numpy’s notation for handling vectors:

- $\vec{1}$ is the constant vector [1,1,1,1,1]
- $\overrightarrow{\text{spec}}$ is a vector from the dataset, `["seto", "virg", "virg", "seto", ...]`

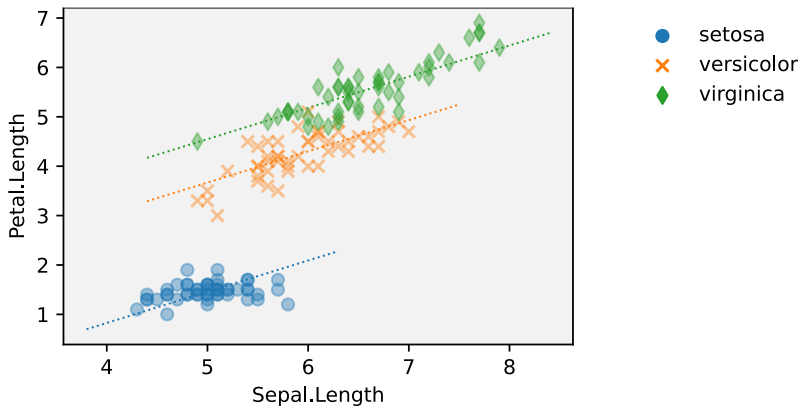
- $f(\vec{x})$ means “apply the function to each element of $\vec{x}$”
- $1_{\overrightarrow{\text{spec}}=\text{"seto"}}$ means “apply the indicator to each element of $\overrightarrow{\text{spec}}$”

- $\overrightarrow{\text{SL}} * 1_{\overrightarrow{\text{spec}}=\text{"seto"}}$ uses element-wise multiplication

```
import sklearn.linear_model

species, SL, PL = iris['Species'], iris['Petal.Length'], iris['Sepal.Length']

species_levels = ['setosa', 'virginica', 'versicolor']
i1,i2,i3 = [np.where(species==k, 1, 0) for k in species_levels]

X = np.column_stack([i1, i2, i3, i1*SL, i2*SL, i3*SL])
m = sklearn.linear_model.LinearRegression(fit_intercept=False)
m.fit(X, PL)
m.coef_
```

EXERCISE

Fit the model with three parallel straight lines.

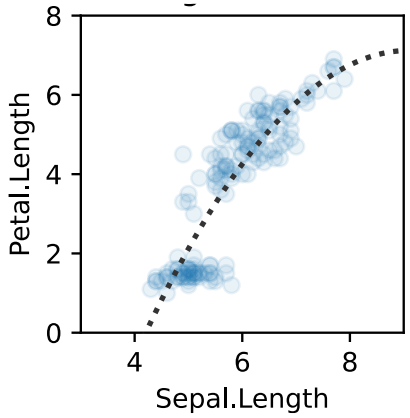


$$\vec{PL} \approx \alpha_{species} + \beta \vec{SL}$$

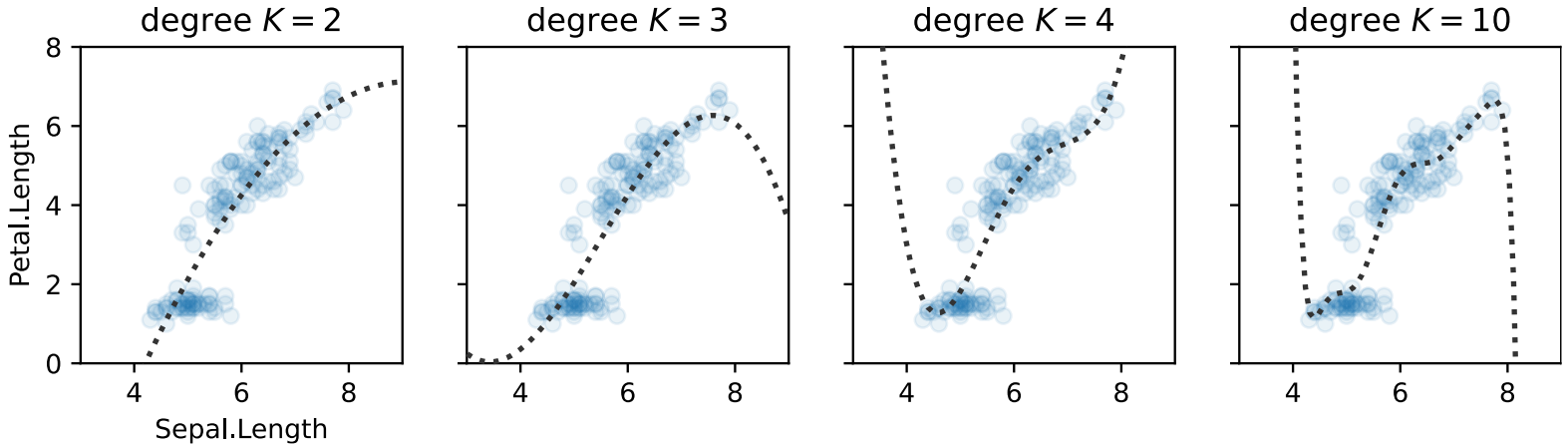
$$= \alpha_{seto} 1_{\vec{spec}=seto} + \alpha_{viry} 1_{\vec{spec}=viry} + \alpha_{vers} 1_{\vec{spec}=vers} + \beta SL.$$

"one-hot coding"

NON-LINEAR RESPONSE



$$\texttt{Petal.Length} \approx \alpha + \beta\,\texttt{Sepal.Length} + \gamma(\texttt{Sepal.Length})^2$$



$$\texttt{Petal.Length} \approx \beta_0 + \sum_{k=1}^{K} \beta_k (\texttt{Sepal.Length})^k$$

Q. Should we just keep adding more and more features to our model?

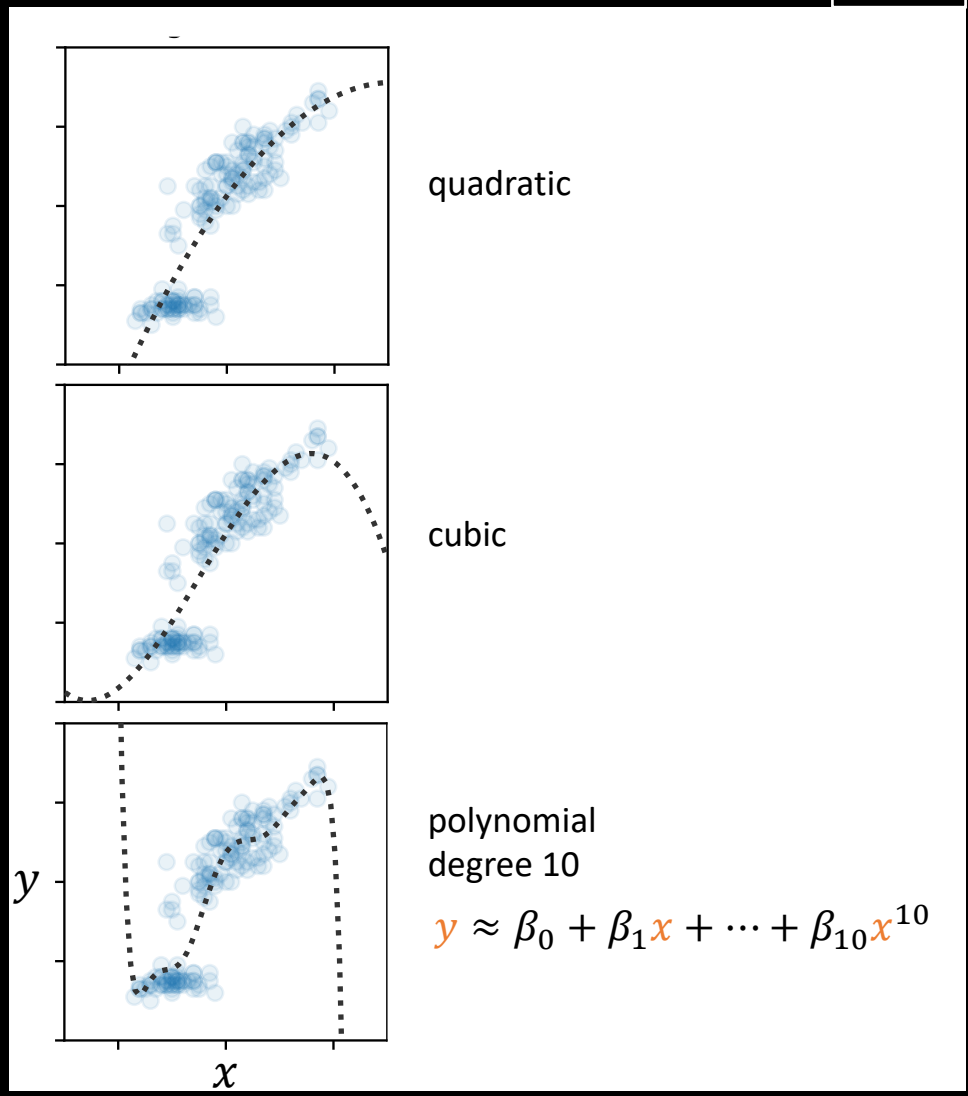
(seeing as the more features we add, the better we can fit the dataset)

A. No. If we did, we'd *overfit*.

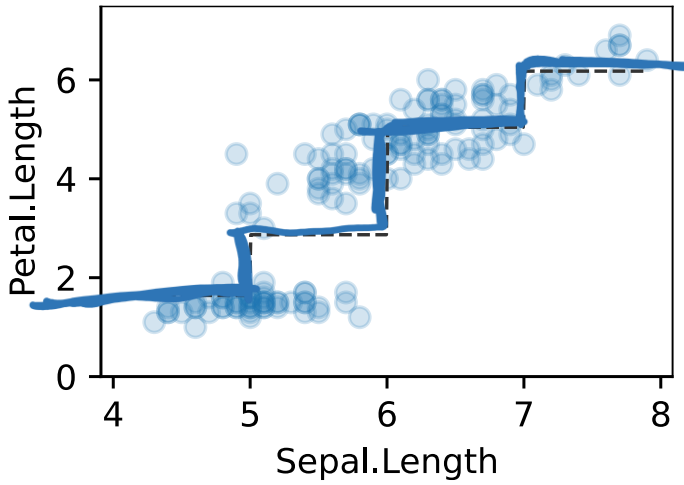
Only add in features that you (as a scientist) believe are relevant.

Or, evaluate your model on a holdout set.
If your model is overfitted to the *training* data, it'll perform poorly on *holdout* data.

[§4, 8–10]



NON-LINEAR RESPONSE via one-hot coding

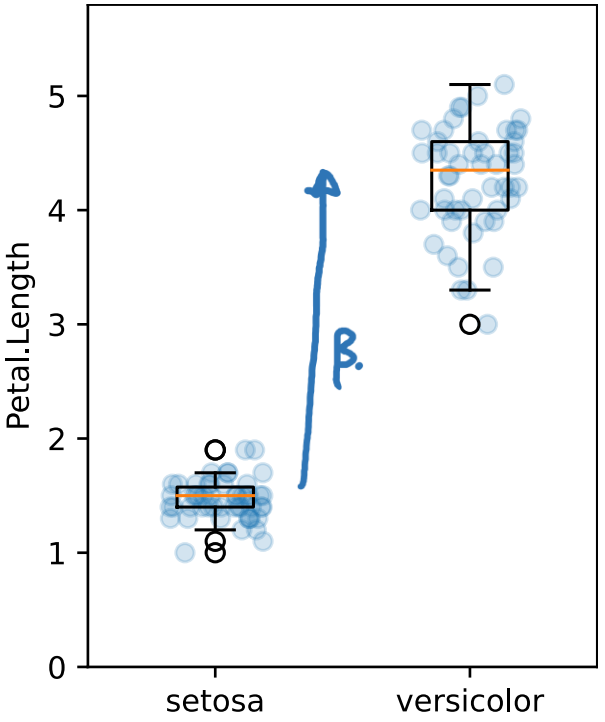


$$\mathrm{PL} \approx \alpha_4 1_{\lfloor \mathrm{SL} \rfloor \leq 4} + \alpha_5 1_{\lfloor \mathrm{SL} \rfloor = 5} + \alpha_6 1_{\lfloor \mathrm{SL} \rfloor = 6} + \alpha_7 1_{\lfloor \mathrm{SL} \rfloor \geq 7}$$

e.g. for an obs with SL = 5.6,
this model predicts PL ≈ α_5

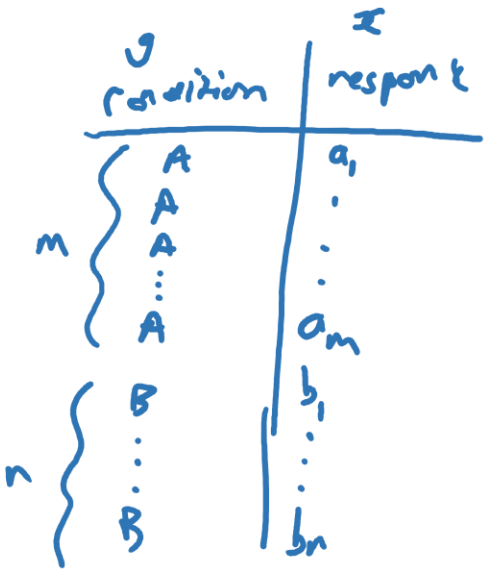
e.g. for an obs with SL = 7.7
this model predicts PL ≈ α_4

COMPARING GROUPS



Measurements for condition A: $a = [a_1, a_2, \ldots, a_m]$
Measurements for condition B: $b = [b_1, b_2, \ldots, b_n]$

Can we use a linear model to compare A and B?



$$\vec{x} \approx \alpha_A 1_{\vec{y}=A} + \alpha_B 1_{\vec{y}=B}$$

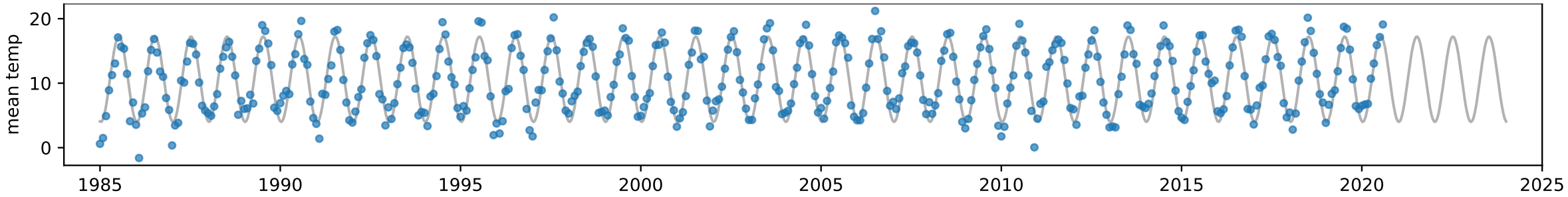
OR:

$$\vec{x} = \alpha + \beta 1_{\vec{y}=B.}$$

For a person of group A, $x \approx \alpha$
B, $x \approx \alpha + \beta$.

} This, β is meaning the difference between the groups.

PERIODIC PATTERN



We'd like to fit the model:

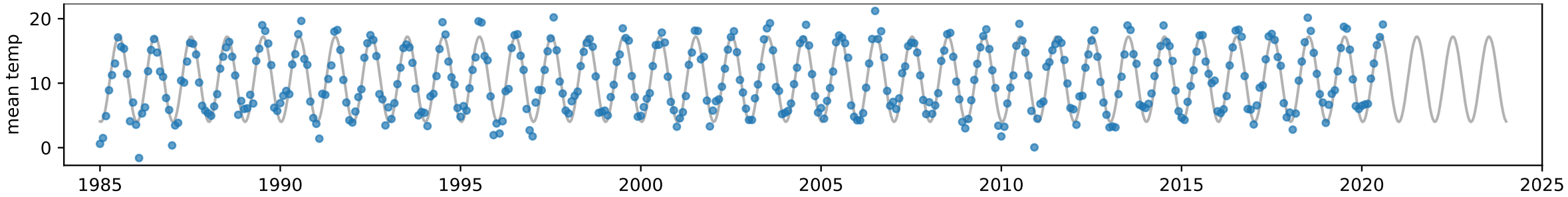
$$\texttt{temp} \approx \alpha + \beta \sin(2\pi(\texttt{t}+\varphi))$$

It looks like we can't use `sklearn.LinearRegression`. That's only for linear models, e.g.

$$\texttt{temp} \approx \alpha + \beta\texttt{e} + \gamma\texttt{f}$$

Instead, we could just brute-force optimize it with `scipy.optimize.fmin`.

PERIODIC PATTERN



We'd like to fit the model:

$$\texttt{temp} \approx \alpha + \beta \sin(2\pi \texttt{t} + \varphi)$$

$$\approx \alpha + \beta \{\sin(2\pi \texttt{t}) \cos\phi + \cos(2\pi \texttt{t}) \sin\phi\}$$

$$= \alpha + (\beta \cos\phi) \sin(2\pi \texttt{t}) + (\beta \sin\phi) \cos(2\pi \texttt{t})$$

$$= \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t})$$

$$= \alpha + \beta_1 e + \beta_2 f$$

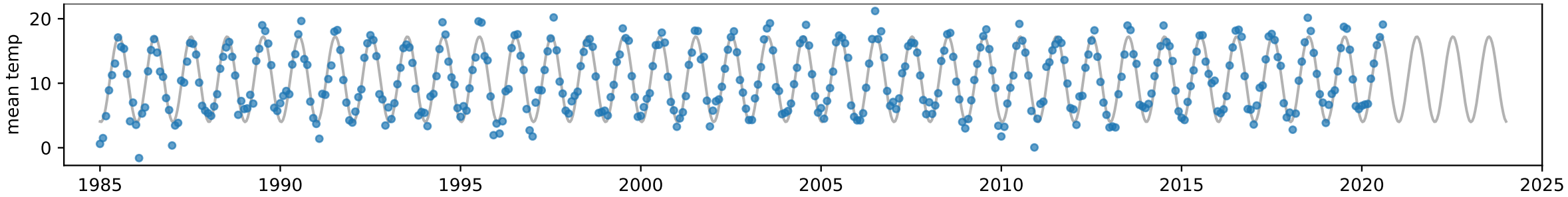
From secondary school trigonometry,

$$\sin(A + B) = \sin(A)\cos(B) + \cos(A)\sin(B)$$

a linear model with feature vectors 1, $\sin(2\pi \texttt{t})$, $\cos(2\pi \texttt{t})$

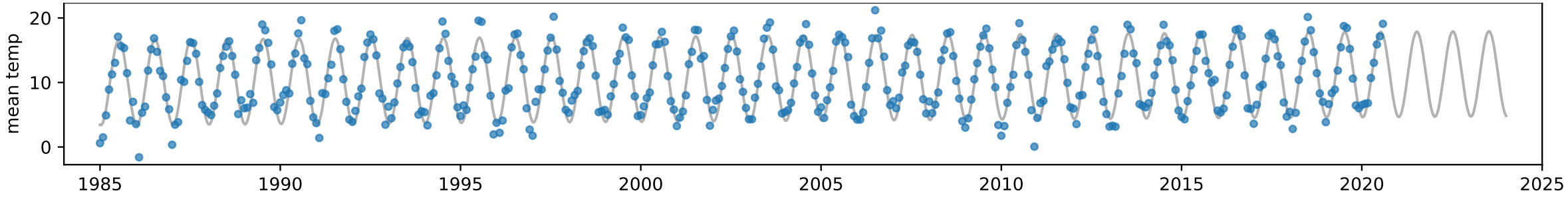
PERIODIC PATTERN

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t})$$



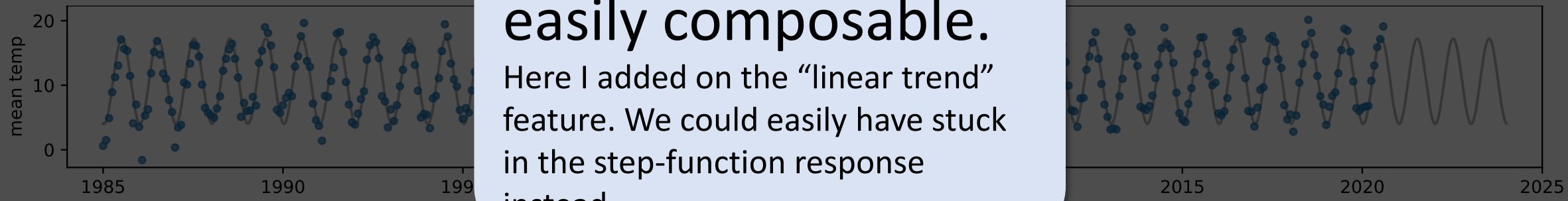
PERIODIC PATTERN + SECULAR TREND

$$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi \texttt{t}) + \beta_2 \cos(2\pi \texttt{t}) + \gamma \texttt{t}$$



PERIODIC PATTERN

$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi\texttt{t}) + \beta$

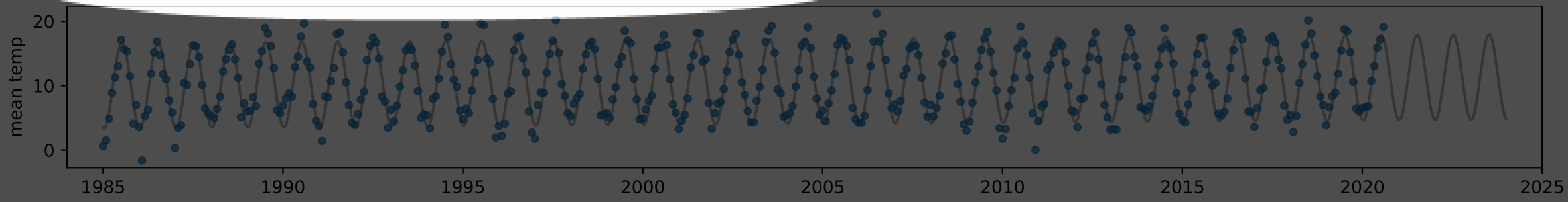


Linear models are easily composable.

Here I added on the "linear trend" feature. We could easily have stuck in the step-function response instead.

PERIODIC PATTERN + SECULAR TREND

$\texttt{temp} \approx \alpha + \beta_1 \sin(2\pi\texttt{t}) + \beta_2 \cos(2\pi\texttt{t}) + \gamma\texttt{t}$



climate

cl.cam.ac.uk/teaching/2223/DataSci/datasci/ex/climate.html

In [1]:
```
import numpy as np
import pandas
import matplotlib.pyplot as plt
import sklearn.linear_model
π = np.pi
```

Climate dataset challenge

- What is the rate of temperature increase in Cambridge?
- Are temperatures increasing at a constant rate, or has the increase accelerated?
- How do results compare across the whole of the UK?

Your task is to answer these questions using appropriate linear models, and to produce elegant plots to communicate your findings. Please submit a Jupyter notebook, or a pdf. Include explanations of what your models are, and of what your plots show.

The dataset is from https://www.metoffice.gov.uk/pub/data/weather/uk/climate/. Code for retrieving the dataset is given at the bottom.

Upload your answers to Moodle by Sunday for presentation / discussion next week

You've got to have models in your head. And you've got to array your experience – both vicarious and direct – on this latticework of models.

You may have noticed students who just try to remember and pound back what is remembered. Well, they fail in school and in life. You've got to hang experience on a latticework of models in your head.

Charlie Munger (business partner of Warren Buffet),
A lesson on elementary, worldly wisdom as it relates to investment management & business.