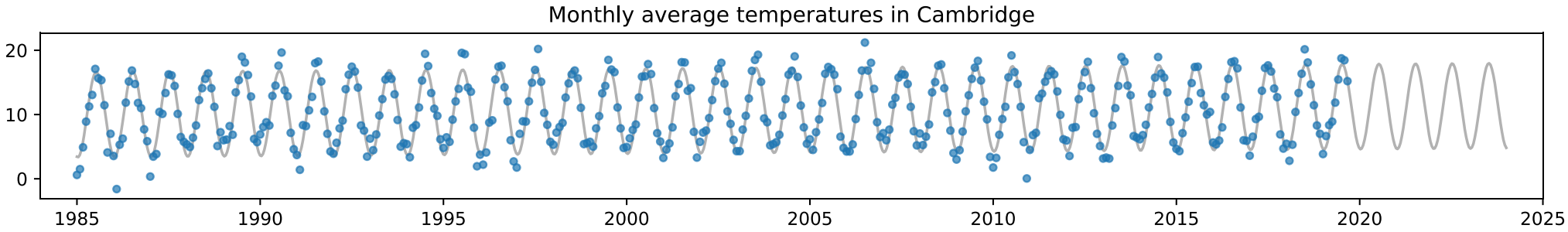
How can we estimate the rate of temperature increase from this dataset?



station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08				9	4.9	1.5
Cambridge	1985	3	1985.16				1	8.7	4.9
⋮									

er station in Cambridge

In lectures, we proposed a sinusoid, scaled and shifted, plus randomness.

```
1   df = pandas.read_csv('https://teachingfiles.blob.core.windows.net/datasets/climate.csv')
2
3   k = 10
4   phase = 0.3
5   sigma = 5
6   offset = 10
7
8   temp = numpy.random.normal(loc = k * numpy.sin(2*π*df['t']+phase) + offset, scale=sigma)
```

Today’s lecture:

Linear models are a good tools for describing data.

They let us spend our time thinking about the phenomena we’re trying to model, not about the mechanics of making the model work.

About the Pr notation

Discrete random variables:
$\Pr_X(x) = \mathbb{P}(X = x)$

called the *probability mass function*

Continuous random variables:
$\Pr_X(x) = \mathrm{pdf}(x)$

the *probability density function*

Transforms of random variables, e.g.
$\Pr_{X+Y}(0.2)$

the notation tells us both the random variable (i.e. a function) and an outcome

Pair of discrete random variables:
$\Pr_{X,Y}(x, y) = \mathbb{P}(X = x \text{ and } Y = y)$

and, if X and Y are independent,
$\Pr_{X,Y}(x, y) = \Pr_X(x)\Pr_Y(y)$

Pair of continuous random variables:
$\Pr_{X,Y}(x, y) = \cdots$

it's complicated…
but intuitively it's like the discrete case

Guess!
$\Pr(x, y, z)$
$\Pr_{X,Y,Z}(u)$

About the Pr notation

Parameterized random variables:
$\Pr_X(x \mid \text{all } X\text{'s parameters})$

Maximum likelihood estimation:
$\log \text{lik}(\text{all unknown parameters} \mid \text{all the observations you can use})$

all the parameters belong in Pr, even if we save ink by not writing them

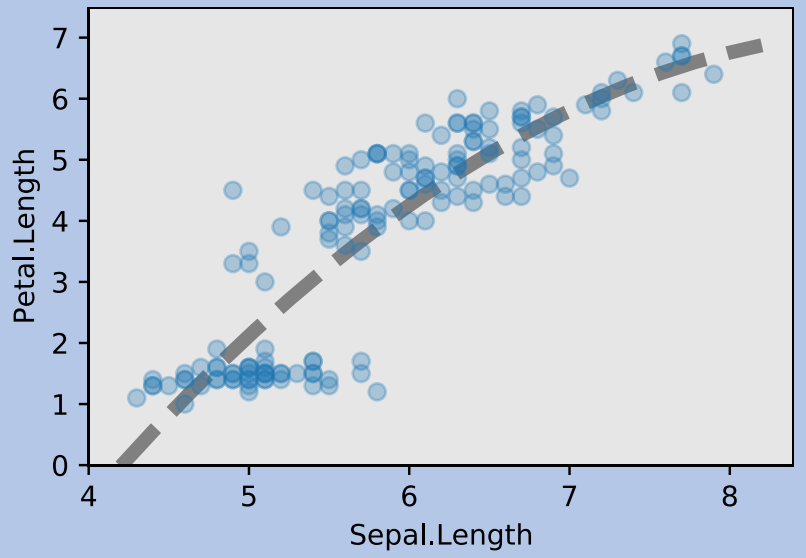
- we must put *all* unknown parameters on the left hand side
- we should use as much data as we can, whatever has any bearing on the parameters

2.1. Fitting a linear model

A *linear model* is a model of the form

$$y = \beta_1 e_1 + \cdots + \beta_K e_K + \varepsilon$$

- y is the response vector $[y_1, y_2, \ldots, y_n]$
- $e_1, \ldots, e_K$ are features, each a vector of length n
- $\beta_k \in \mathbb{R}$ is the parameter that weights the kth feature
- $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_n]$ is the *residuals* vector / error / noise

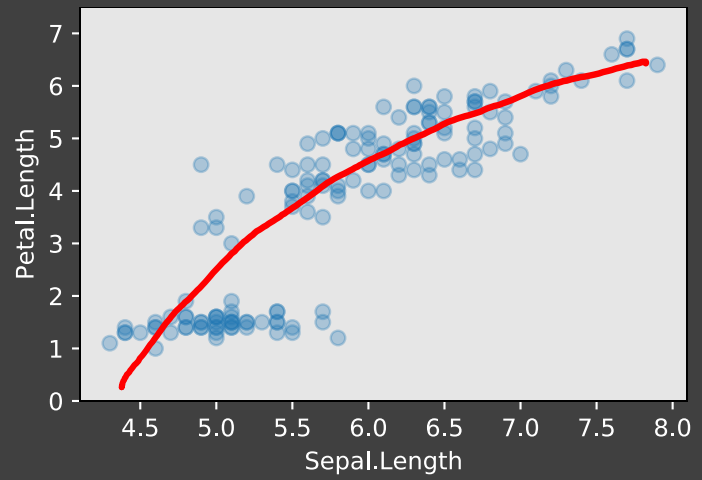


Exercise 2.1.

The Iris dataset, popularized by Ronald Fisher (a genius who almost single-handedly created the foundations for modern statistical science), 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
1.7	0.5	5.1	3.3	setosa
4.2	1.2	5.7	3.0	versicolor
…				

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



Let's guess that for parameters α, β, γ (to be estimated),

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

This is called a *linear model* because it can be written in *linear algebra* form, using vectors for the entire dataset.

The response vector is

$$\texttt{Petal.Length} = [\mathrm{PL}_1, \mathrm{PL}_2, \ldots, \mathrm{PL}_n]$$

The feature vectors are

$$\texttt{one} = [1,1,\ldots,1]$$
$$\texttt{Sepal.Length} = [\mathrm{SL}_1, \mathrm{SL}_2, \ldots, \mathrm{SL}_n]$$
$$(\texttt{Sepal.Length})^2 = [(\mathrm{SL}_1)^2, (\mathrm{SL}_2)^2, \ldots, (\mathrm{SL}_n)^2]$$

The response vector is predicted by a linear combination of feature vectors:

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

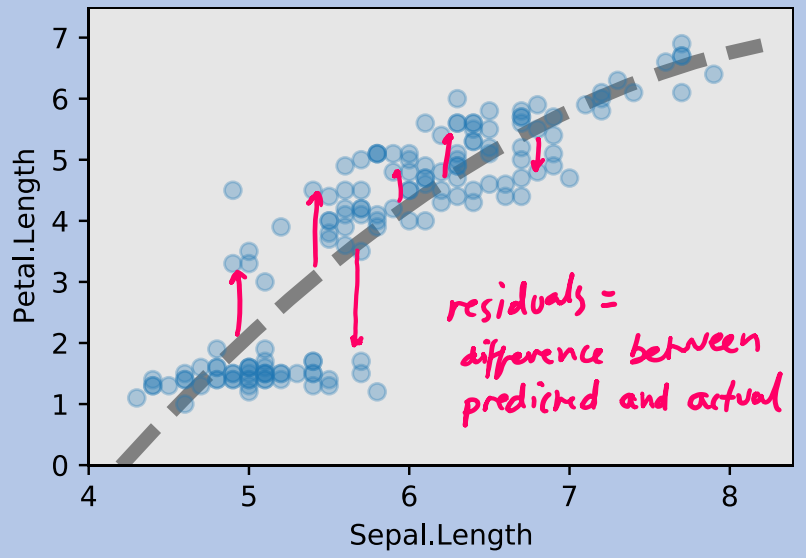
2. Feature spaces / linear regression

2.1. Fitting a linear model

A *linear model* is a model of the form

$$y = \overbrace{\beta_1 e_1 + \cdots + \beta_K e_K}^{\text{prediction}} + \overbrace{\varepsilon}^{\text{residual}}$$

- y is the response vector $[y_1, y_2, \ldots, y_n]$
- $e_1, \ldots, e_K$ are features, each a vector of length n
- $\beta_k \in \mathbb{R}$ is the parameter that weights the kth feature
- $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_n]$ is the *residuals* vector / error / noise



The natural way to fit this model is to choose parameters to minimize the *mean square error*

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

This is called *least squares estimation*.

```
1  iris = pandas.read_csv('https://teachingfiles.blob.core.windows.net/datasets/iris.csv')
```

Fitting the model, using least squares estimation

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

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

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

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

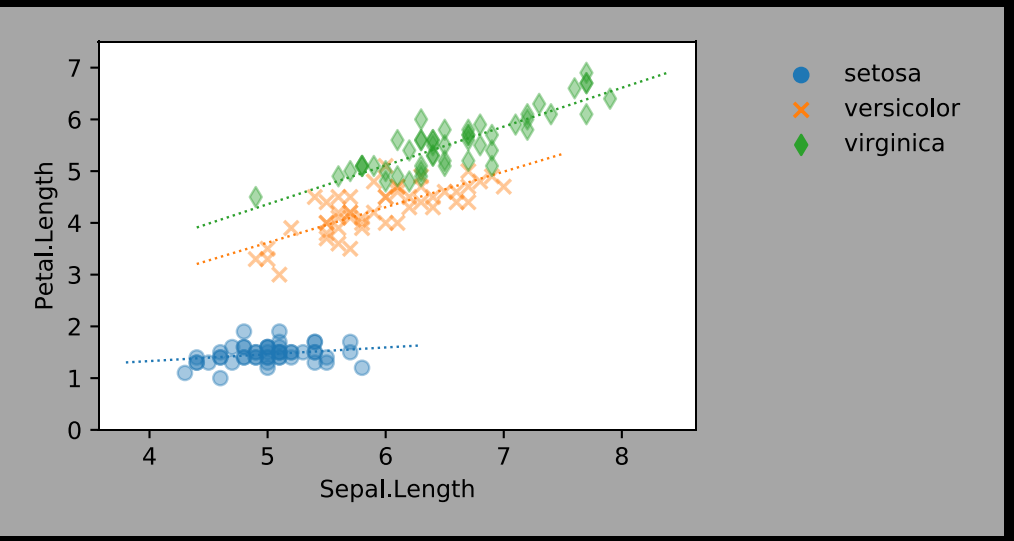
The LinearRegression() command will automatically put in the 1 feature for you, unless you explicitly tell it not to.

Making predictions / getting fitted values from the model

```
6  newx = numpy.linspace(4.2, 8.2, 20)
7  predy = α + β*newx + γ*(newx**2)
```

```
6  newx = numpy.linspace(4.2, 8.2, 20)
7  predy = model2.predict(numpy.column_stack([newx, newx**2]))
```

With well-designed features, we can ask all sorts of questions about a dataset.



Petal. length Sepal. length

$$PL \approx \alpha_{species} + \beta_{species} SL$$

This is a set of three straight-line fits, one for each species. The entire model has 6 parameters:

α_{seto} α_{vers} α_{virg}

β_{seto} β_{vers} β_{virg}.

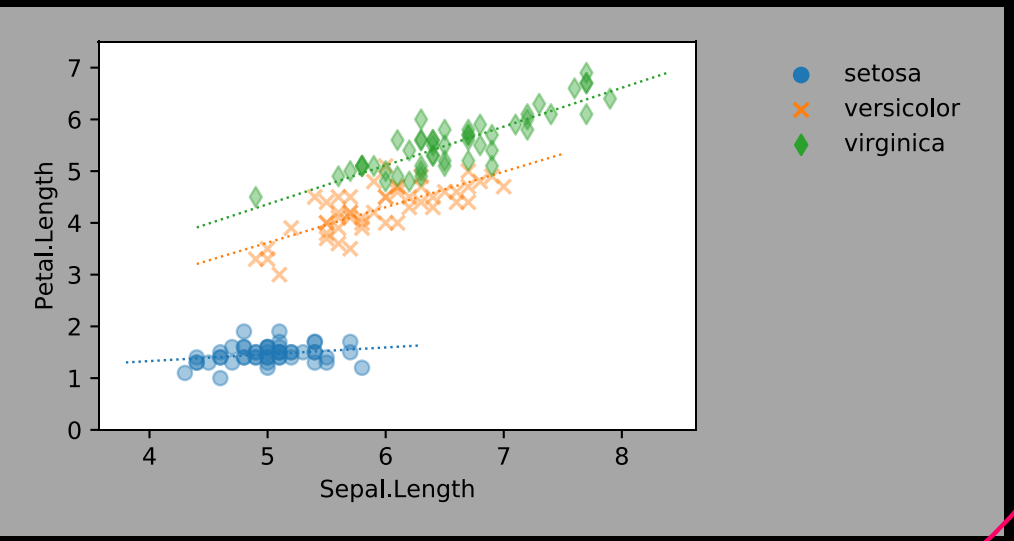
To write it as a linear model — ie in vector form, one feature vector per parameter:

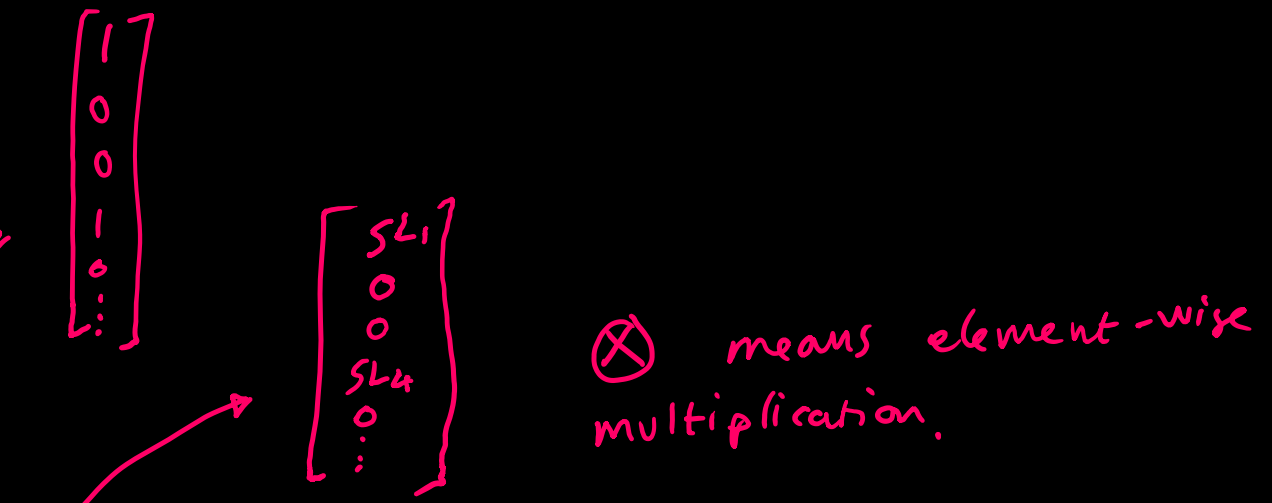
$$\begin{matrix} seto \\ virg \\ virg \\ seto \\ vers \\ \vdots \end{matrix} \begin{bmatrix} PL_1 \\ PL_2 \\ PL_3 \\ PL_4 \\ PL_5 \\ \vdots \end{bmatrix} \approx \alpha_{seto} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} + \beta_{seto} \begin{bmatrix} SL_1 \\ 0 \\ 0 \\ SL_4 \\ 0 \\ \vdots \end{bmatrix} + \alpha_{virg} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \beta_{virg} \begin{bmatrix} 0 \\ SL_2 \\ SL_3 \\ 0 \\ 0 \\ \vdots \end{bmatrix} + \alpha_{vers} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} + \beta_{vers} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ SL_5 \\ \vdots \end{bmatrix}$$

response vector, one item for each row of the dataset

feature vectors, one item for each row of the dataset

With well-designed features, we can ask all sorts of questions about a dataset.





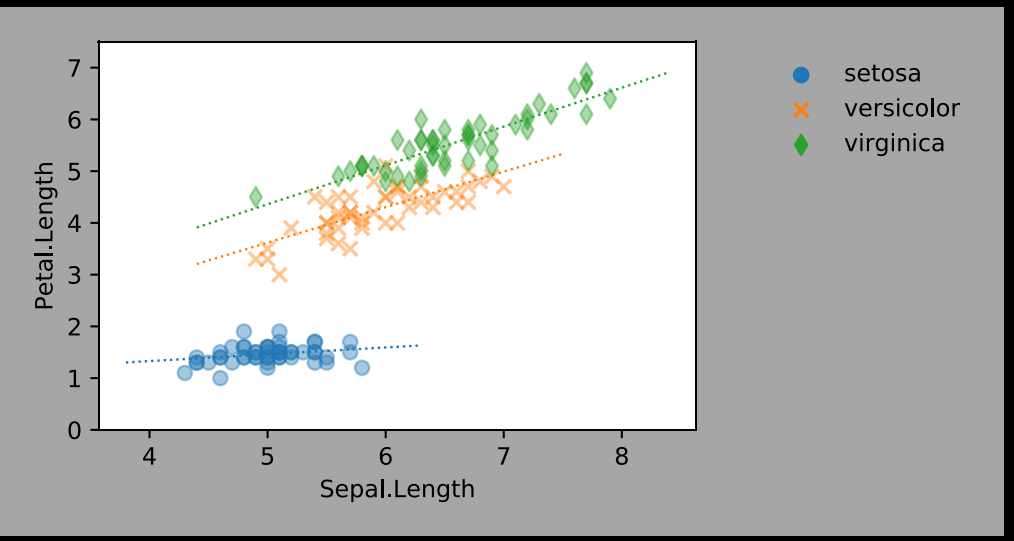
In vector algebraic notation,

$$\text{PL} \approx \alpha_{\text{seto}} 1_{\text{spec}=\text{"seto"}} + \alpha_{\text{vers}} 1_{\text{spec}=\text{"vers"}} + \alpha_{\text{virg}} 1_{\text{spec}=\text{"virg"}}$$
$$+\beta_{\text{seto}} \left(1_{\text{spec}=\text{"seto"}} \otimes \text{SL}\right) + \beta_{\text{vers}} \left(1_{\text{spec}=\text{"vers"}} \otimes \text{SL}\right) + \beta_{\text{virg}} \left(1_{\text{spec}=\text{"virg"}} \otimes \text{SL}\right)$$

When writing linear models, everything we do is taken to be vectorized:

```
1   spec = ["seto", "virg", "virg", "seto", "vers", …]
2   1_spec="seto" = [1 if s=="seto" for s in spec]
3   α_spec = [αdict[s] for s in spec],   αdict = {"seto":α_seto, "vers":α_vers, "virg":α_virg}
```

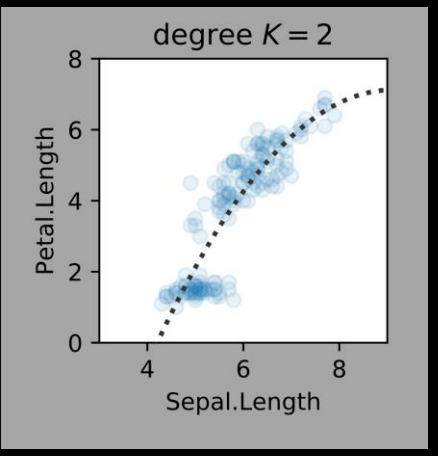
With well-designed features, we can ask all sorts of questions about a dataset.



In vector algebraic notation,

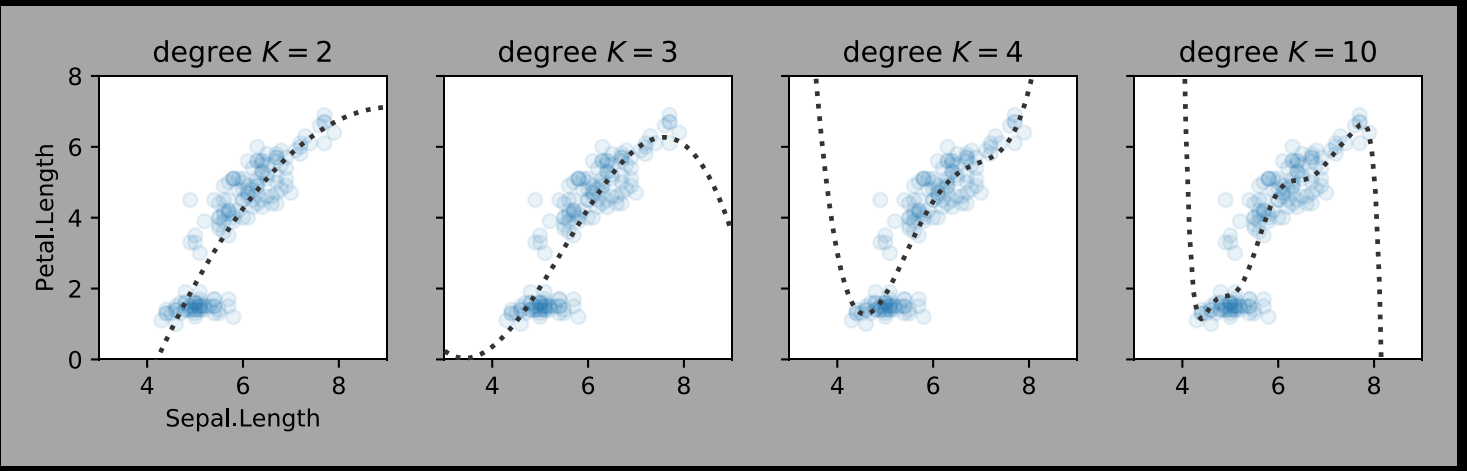
$$\text{PL} \approx \alpha_{\text{seto}} 1_{\text{spec}=\text{"seto"}} + \alpha_{\text{vers}} 1_{\text{spec}=\text{"vers"}} + \alpha_{\text{virg}} 1_{\text{spec}=\text{"virg"}} + \beta_{\text{seto}} (1_{\text{spec}=\text{"seto"}} \otimes \text{SL}) + \beta_{\text{vers}} (1_{\text{spec}=\text{"vers"}} \otimes \text{SL}) + \beta_{\text{virg}} (1_{\text{spec}=\text{"virg"}} \otimes \text{SL})$$

we're representing the string vector spec = ["seto", "virg", "virg", "seto", "vers", ...]
with three binary vectors $(1_{\text{spec}=\text{"seto"}}, 1_{\text{spec}=\text{"vers"}}, 1_{\text{spec}=\text{"virg"}})$.
This is called "one-hot encoding".



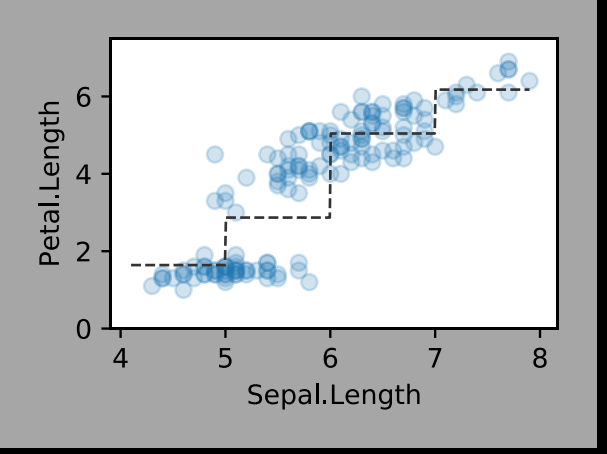
Quadratic response:

$$\text{PL} \approx \alpha + \beta\ \text{SL} + \gamma(\text{SL})^2$$



Polynomial response:

$$\text{PL} \approx \beta_0 + \beta_1\ \text{SL} + \cdots + \beta_K(\text{SL})^K$$

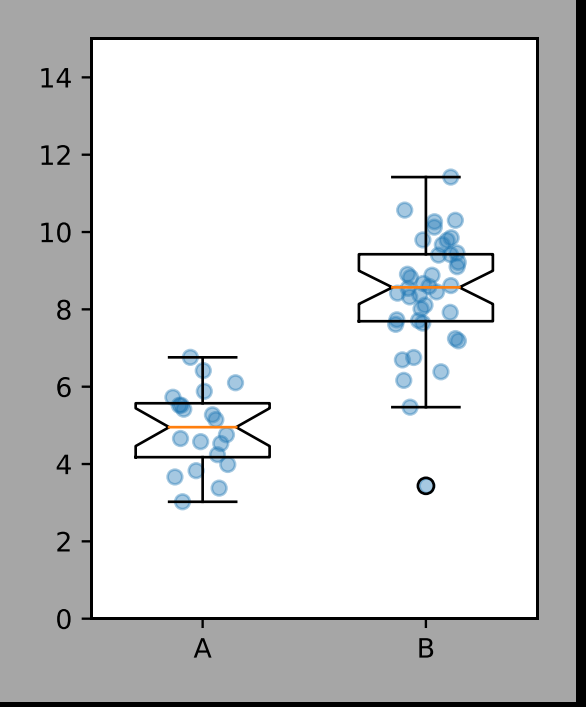


$$PL = \alpha_4 1_{SL \in [4,5)} + \alpha_5 1_{SL \in [5,6)} + \alpha_6 1_{SL \in [6,7)} + \alpha_7 1_{SL \in [7,8)}$$

So, for example, if row i has $SL_i = 6.2$, the model formula says

$$PL_i \approx \alpha_4 \times 0 + \alpha_5 \times 0 + \alpha_6 \times 1 + \alpha_7 \times 0$$

$$= \alpha_6.$$



Measurements for condition A:

$x = [x_1, x_2, \ldots, x_m]$

Measurements for condition B:

$y = [y_1, y_2, \ldots, y_n]$

What's a linear model for the entire set of measurements?

cond	meas
A	x_1
A	x_2
⋮	⋮
A	x_m
B	y_1
B	y_2
⋮	⋮
B	y_n

First think of how you'd store all the data in a spreadsheet / database table.

Then, the measurements can be described with one-hot coding:

$$meas \approx \alpha_A 1_{cond = \text{"A"}} + \alpha_B 1_{cond = \text{"B"}}$$

COMPUTER SCIENCE TRIPOS Part IB – mock – Paper 6

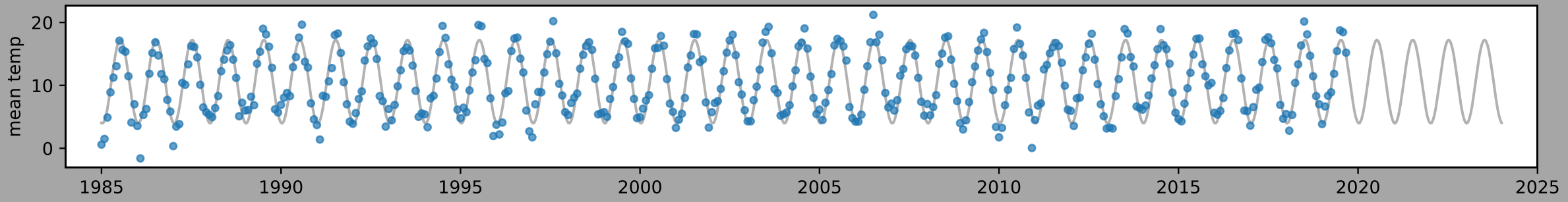
2. Foundations of Data Science (DJW)

A group of three friends want to find out how well they know each other. They settle on a questionnaire about tastes (favourite film, favourite food, favourite mustelid, etc.) and then each tries to guess what each other will answer. Let x_{ij} be the number of correct answers when i tries to guess j's answers, and consider the model

$$x_{ij} \approx \alpha_i + \beta_j.$$

Here α_i represents the perceptiveness of i, and β_j represents the openness of j.

a) Write the model for x_{ij} as a linear model, and identify the feature vectors.



$$\text{temp} \approx \alpha + \beta \sin(2\pi t + \theta)$$

θ = phase
β = amplitude
α = offset

A linear model is

$$\text{response} \approx \sum \beta_k e_k$$

β_k = weights to be estimated
e_k = feature vectors

From school trigonometry:

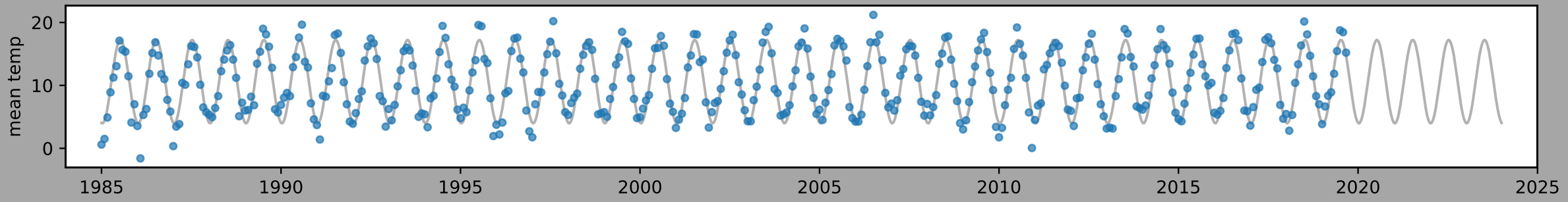
$$\sin(A+B) = \sin A \cos B + \cos A \sin B$$

$$\text{temp} \approx \alpha + \beta\left(\sin(2\pi t)\cos\theta + \cos(2\pi t)\sin\theta\right)$$

$$= \alpha + (\beta\cos\theta)\sin 2\pi t + \beta\sin\theta\cos(2\pi t)$$

$$= \alpha + \beta_1 \sin 2\pi t + \beta_2 \cos 2\pi t$$

The feature vectors here are $1, \sin 2\pi t, \cos 2\pi t$.



$$temp \approx \alpha + \beta_1 \sin 2\pi t + \beta_2 \cos 2\pi t$$

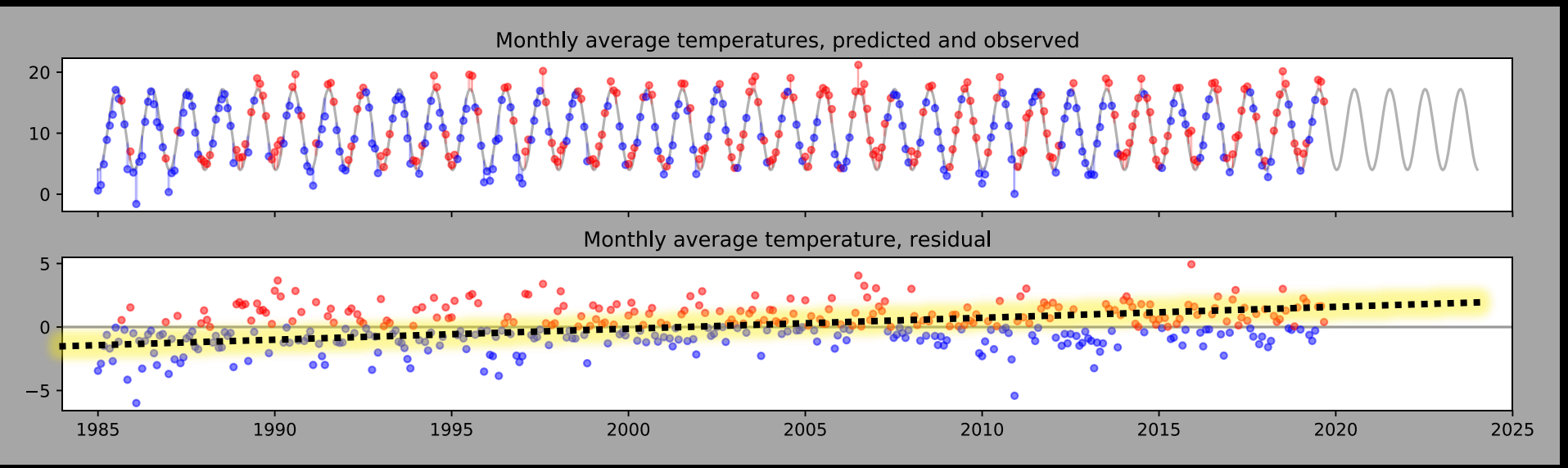
If we believe there's some systematic rate of temperature increase, we can include it by simply adding a term:

$$temp \approx \alpha + \beta_1 \sin 2\pi t + \beta_2 \cos 2\pi t + \gamma t$$

γ is the annual rate of temperature increase.

How do we know we've got all the right features?

1. Fit our model, and get estimated coefficients:
   $y = \hat{\beta}_1 e_1 + \cdots + \hat{\beta}_K e_K + \varepsilon$
2. Compute the residuals ε
   `ε = y - model.predict()`
3. Plot ε against any other features we think might be relevant. If there's a systematic pattern, decide on a formula for it, and add a term to our model.



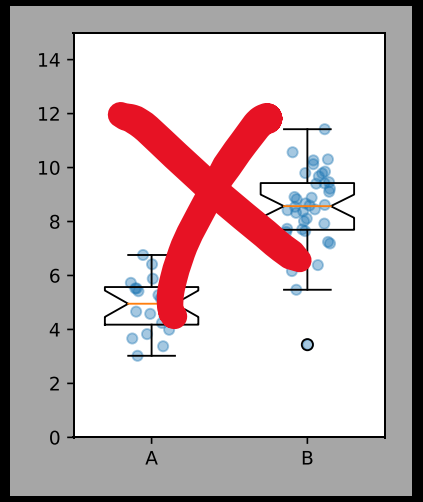
This plot suggests $\varepsilon \approx \alpha + \beta t$.

So we can refine our model:

$y \approx \beta_1 e_1 + \cdots + \beta_K e_K + \alpha + \beta t$.

- There's no reason to believe linear models are *true*—but they're useful because they are flexible and interpretable.
- George Box, one of the greatest statistical minds of the 20th century, wrote that

  "All models are wrong—but some are useful"



- Box didn't invent the Box plot—Mary Eleanor Spear did.