Tutorial 10

Statistical Computation and Analysis

Spring 2025

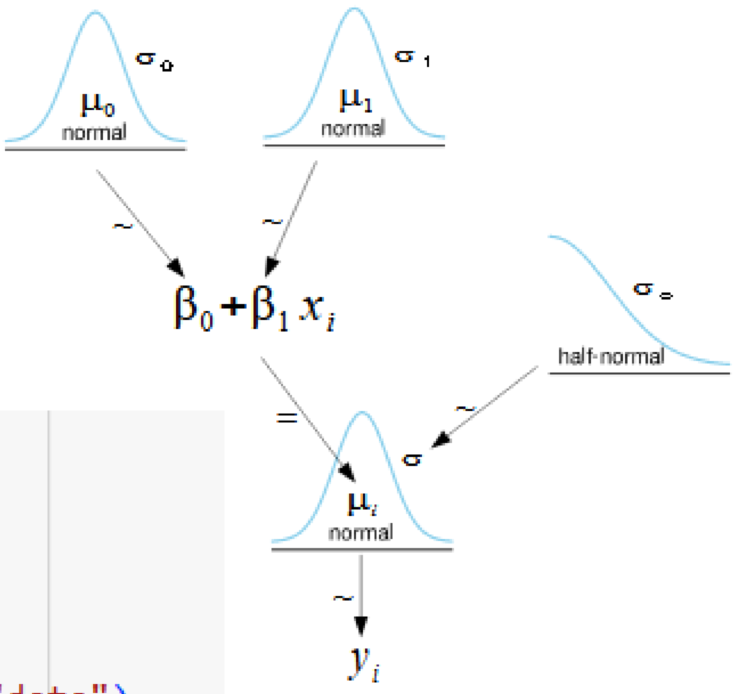
Tutorial Outline

- Bambi

Bambi

- A high-level Bayesian model-building interface written on top of PyMC.
- Designed to make it extremely easy to fit linear models, including hierarchical one.
- https://bambinos.github.io/bambi/

PyMC vs. Bambi



- What we did until now:

```
with pm.Model(coords=coords) as model_slr:
    b0 = pm.Normal("b0", mu=50, sigma=50)
    b1 = pm.Normal("b1", mu=0, sigma=50)
    sig = pm.HalfNormal("sig", 10)
    mu = pm.Deterministic("mu", b0 + b1 * data.Length, dims="data")
    y_pred = pm.Normal("y_pred", mu=mu, sigma=sig, observed=data.Force, dims="data")

    idata_slr = pm.sample(1000, chains = 4)
```

- With Bambi: (import bambi as bmb)

```
a_model = bmb.Model("y ~ x", data)
```

  - The dependent variable is on the left and the independent on the right.

Bambi

- What is bambi doing?
  - Print the model

```
       Formula: y ~ x
        Family: gaussian
          Link: mu = identity
  Observations: 117
        Priors:
    target = mu
        Common-level effects
            Intercept ~ Normal(mu: -0.055, sigma: 2.618)
            x ~ Normal(mu: 0.0, sigma: 2.3841)

        Auxiliary parameters
            sigma ~ HalfStudentT(nu: 4.0, sigma: 1.0456)
```

1. Bambi assumes the likelihood is gaussian (this can be changed).

2. We only specified how the dependent and independent variables are related.

3. Bambi automatically defines weakly informative priors.

Bambi

```
      Formula: y ~ x
       Family: gaussian
         Link: mu = identity
 Observations: 117
       Priors:
   target = mu
       Common-level effects
           Intercept ~ Normal(mu: -0.055, sigma: 2.618)
           x ~ Normal(mu: 0.0, sigma: 2.3841)

       Auxiliary parameters
           sigma ~ HalfStudentT(nu: 4.0, sigma: 1.0456)
```

Relationship between dependent and independent variables

Likelihood

Corresponding link function

Dataset size

Linearly modelling the parameter mu of the Gaussian

Common level effects

Not linearly modeled parameters

Model Structure

Bambi

- If we would like to set our own priors:
  - For example, for:
    - The coefficient of the independent variable (the slope)
    - And for sigma (the auxiliary parameter)
  - Define a dictionary and pass it to the model

```
priors = {"x": bmb.Prior("HalfNormal", sigma=3),
          "sigma": bmb.Prior("Gamma",  mu=1, sigma=2),
          }
a_model_wcp = bmb.Model("y ~ x", data, priors=priors)
```

Bambi

- If we print the model:

```
       Formula: y ~ x
        Family: gaussian
          Link: mu = identity
  Observations: 117
        Priors:
    target = mu
        Common-level effects
            Intercept ~ Normal(mu: 0.02, sigma: 2.837)
            x ~ HalfNormal(sigma: 3.0)

        Auxiliary parameters
            sigma ~ Gamma(mu: 1.0, sigma: 2.0)
```

Bambi

- We can also omit the intercept from the model in two ways:

```
no_intercept_model = bmb.Model("y ~ 0 + x", data)
```

```
no_intercept_model = bmb.Model("y ~ -1 + x", data)
```

```
       Formula: y ~ 0 + x
        Family: gaussian
          Link: mu = identity
  Observations: 117
        Priors:
    target = mu
        Common-level effects
            x ~ Normal(mu: 0.0, sigma: 2.3841)

        Auxiliary parameters
            sigma ~ HalfStudentT(nu: 4.0, sigma: 1.0456)
```

No intercept

Bambi

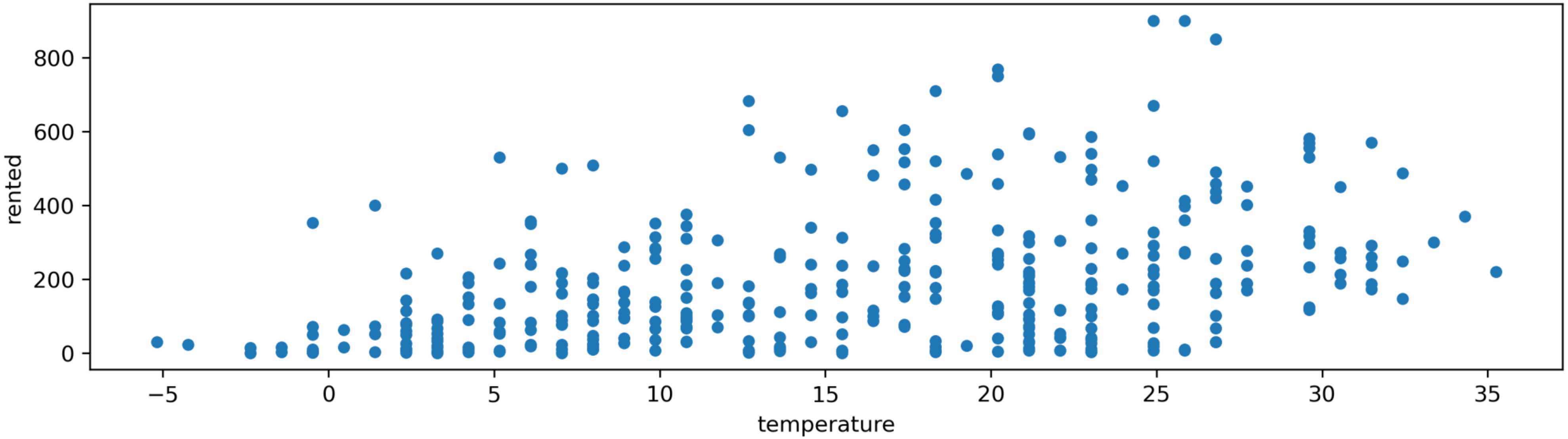
- Multiple regression in Bambi:

```
model_2 = bmb.Model("y ~ x + z", data)
```

- Bambi also allows for modelling linear hierarchical regression.

Bambi – The Bikes Example

- In a previous lecture, you examined the relation between the number of bikes rented and the temperature.
- You compared linear regression with the generalized linear model using the negative binomial distribution.



Bambi – The Bikes Example

- Model in PyMC:

```
with pm.Model() as model_neg:
    α = pm.Normal("α", mu=0, sigma=1)
    β = pm.Normal("β", mu=0, sigma=10)
    σ = pm.HalfNormal("σ", 10)
    μ = pm.Deterministic("μ", pm.math.exp(α + β * bikes.temperature))
    y_pred = pm.NegativeBinomial("y_pred", mu=μ, alpha=σ, observed=bikes.rented)
    idata_neg = pm.sample()
    idata_neg.extend(pm.sample_posterior_predictive(idata_neg))
```

- Model in Bambi:

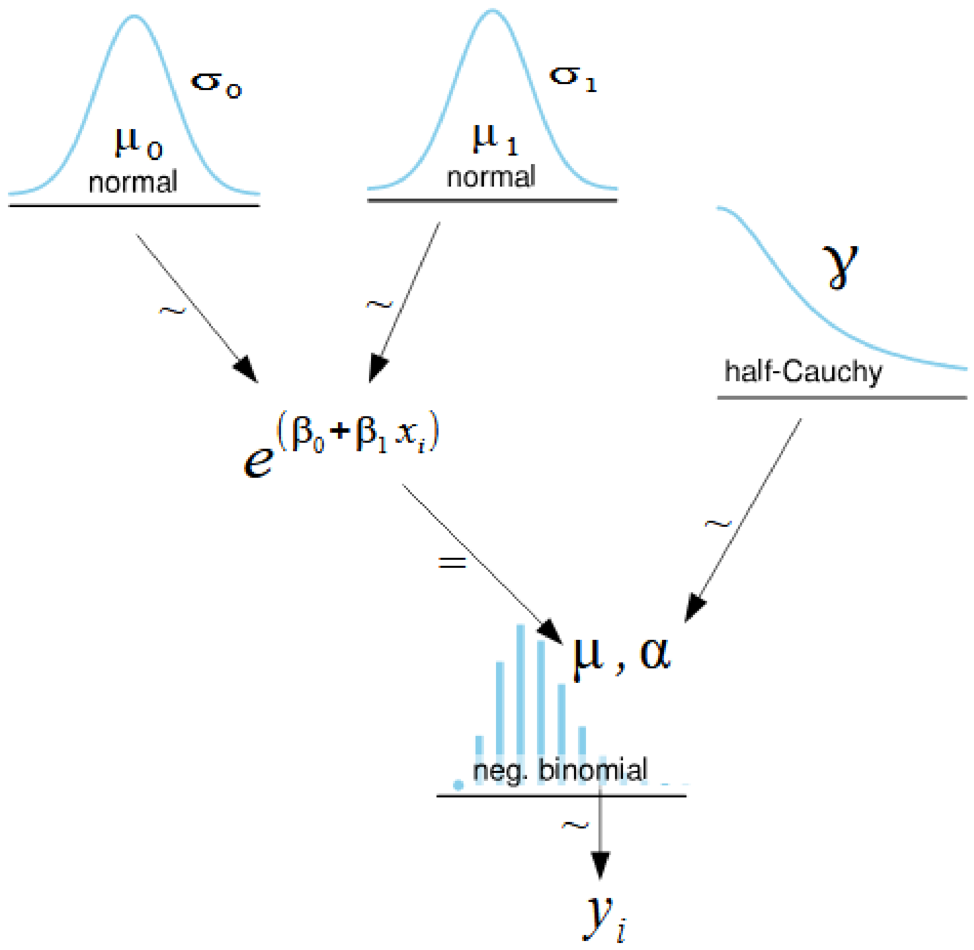
```
model_t = bmb.Model("rented ~ temperature", bikes, family="negativebinomial")
idata_t = model_t.fit()
```

Bambi – The Bikes Example

- Print the model and draw it:

```
     Formula: rented ~ temperature
      Family: negativebinomial
        Link: mu = log
Observations: 348
      Priors:
  target = mu
      Common-level effects
          Intercept ~ Normal(mu: 0.0, sigma: 4.9184)
          temperature ~ Normal(mu: 0.0, sigma: 0.2741)

      Auxiliary parameters
          alpha ~ HalfCauchy(beta: 1.0)
```



Bambi – The Bikes Example

- Using the fit function returns an inference xarray object.

arviz.InferenceData

▼ posterior

xarray.Dataset

► Dimensions: (**chain**: 4, **draw**: 1000)

▼ Coordinates:

chain	(chain)	int64	0 1 2 3
draw	(draw)	int64	0 1 2 3 4 5 ... 995 996 997 998 999

▼ Data variables:

Intercept	(chain, draw)	float64	4.52 4.407 4.448 ... 4.373 4.525
alpha	(chain, draw)	float64	0.9022 0.9849 ... 0.9876 0.9341
temperature	(chain, draw)	float64	0.04138 0.04546 ... 0.05116 0.04303

► Indexes: (2)

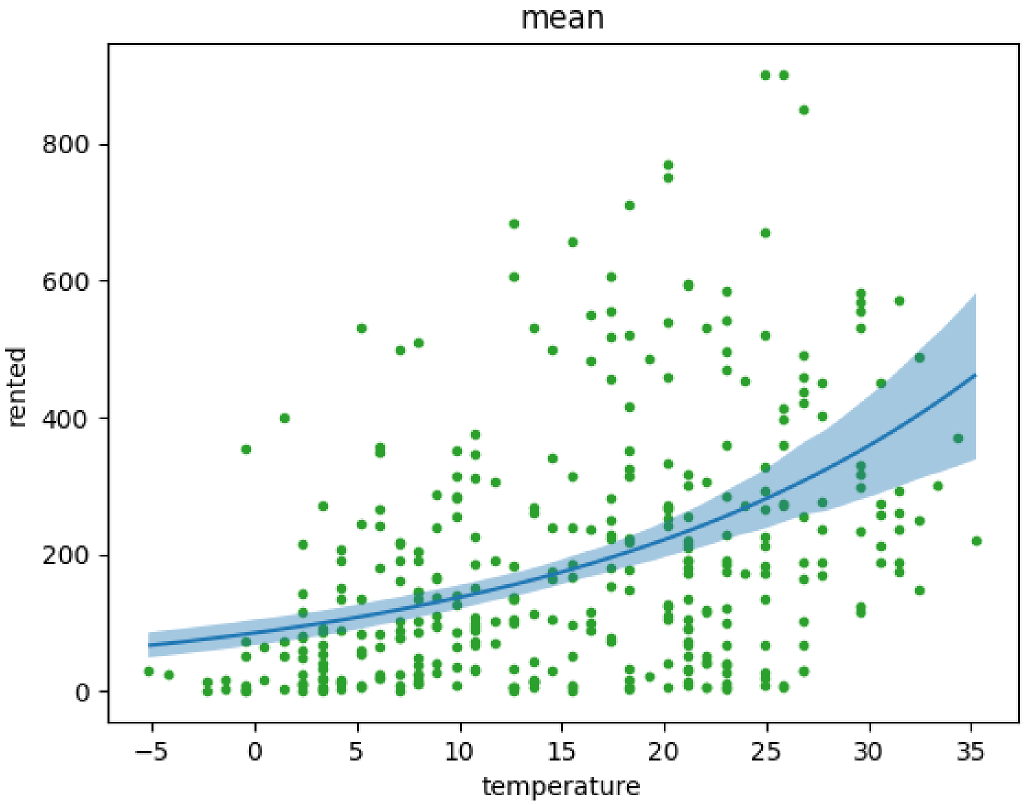
► Attributes: (8)

► sample_stats

► observed_data

Bambi – The Bikes Example

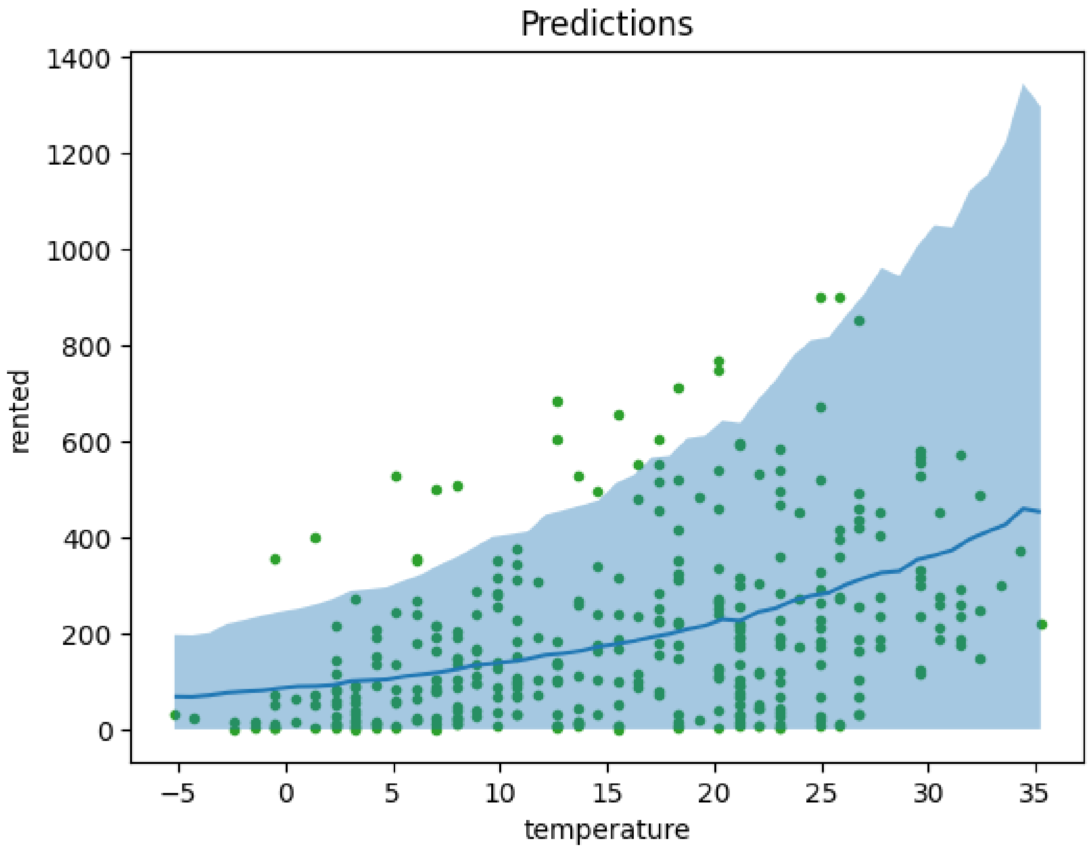
- We can also use bambi functions to look at the posterior.
  - Look at the posterior mean and 94% HDI.
  - Only accounts for the uncertainty in the intercept and slope parameters.



Bambi – The Bikes Example

- Posterior predictive distribution.

- Accounts for the uncertainty in the model parameters and the data.



Bambi – Multiple Regression (Bikes Example)

- Use both the temperate and the humidity as independent variables.

```
model_th = bmb.Model("rented ~ temperature + humidity", bikes, family="negativebinomial")
idata_th = model_th.fit(1000, chains = 4)
```

```
     Formula: rented ~ temperature + humidity
      Family: negativebinomial
        Link: mu = log
Observations: 348
      Priors:
   target = mu
      Common-level effects
          Intercept ~ Normal(mu: 0.0, sigma: 9.6708)
          temperature ~ Normal(mu: 0.0, sigma: 0.2741)
          humidity ~ Normal(mu: 0.0, sigma: 13.2417)

      Auxiliary parameters
          alpha ~ HalfCauchy(beta: 1.0)
```

arviz.InferenceData

▾ posterior

xarray.Dataset

▸ Dimensions: (**chain**: 4, **draw**: 1000)

▸ Coordinates: (2)

▾ Data variables:

Intercept	(chain, draw)	float64	5.576 6.227 5.31 ... 5.468 5.468
alpha	(chain, draw)	float64	0.9507 0.9634 1.073 ... 1.051 1.051
humidity	(chain, draw)	float64	-1.722 -2.576 ... -1.598 -1.598
temperature	(chain, draw)	float64	0.04152 0.03936 ... 0.04469 0.04469

▸ Indexes: (2)

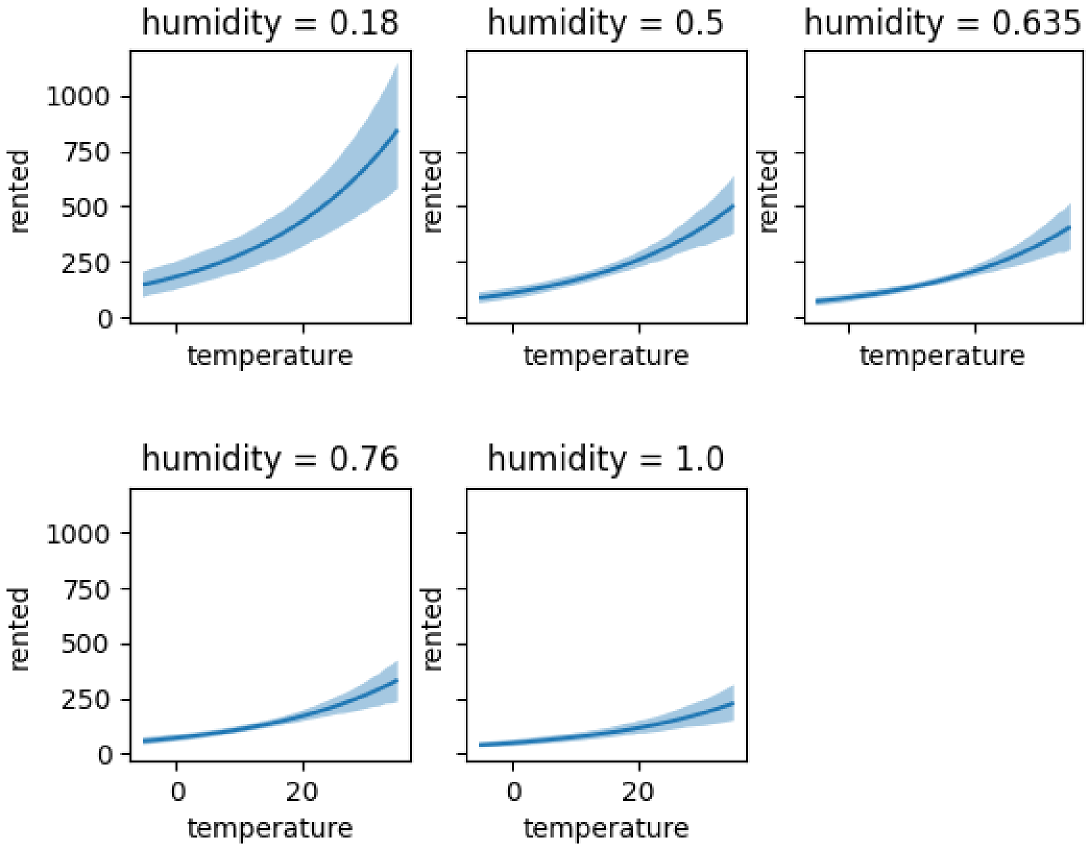
▸ Attributes: (8)

▸ sample_stats

▸ observed_data

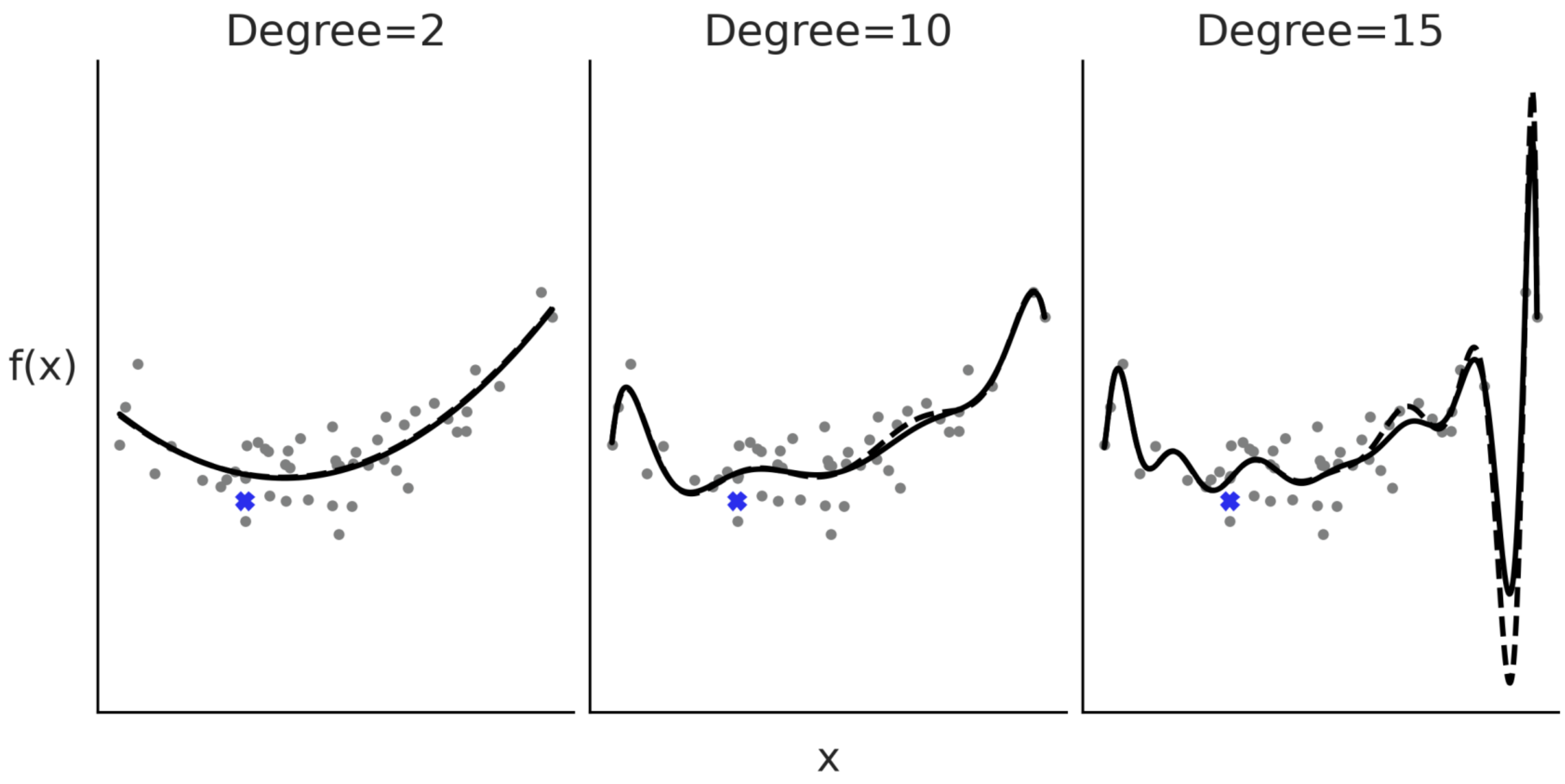
Bambi – Multiple Regression (Bikes Example)

- We can look at the mean posterior for different humidity levels.
  - The number of rented bikes increases with temperature, but the slope is larger when humidity is low.



Polynomial Regression

- $\mu = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots$
- The higher the degree of the polynomial, the more flexible the curve can be.



Polynomial Regression

- There are two ways to define in Bambi:

```
"y ~ x + I(x ** 2) + I(x ** 3) + I(x ** 4)"
```

  - where I is the identity function.

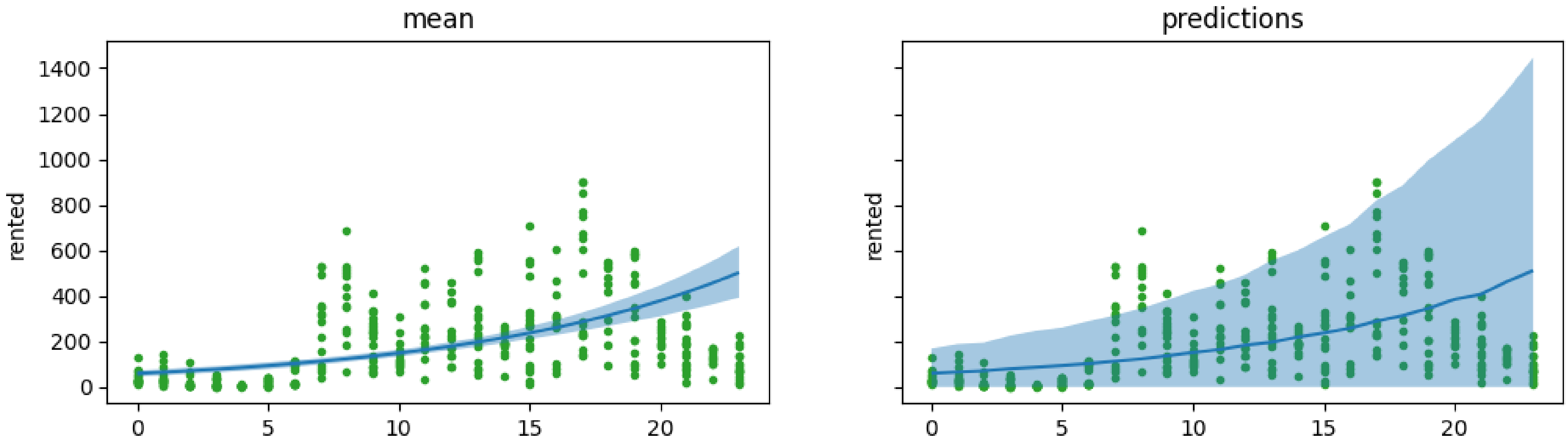
- ```
  "y ~ poly(x, 4)"
  ```
  ```

Polynomial Regression

- The two definitions should lead to the same predictions but are not actually the same.
- Using the poly syntax ensures that the polynomial terms will be orthogonal to each other.
  - Can be numerically more stable.
  - Orthogonal polynomials allow you to interpret the effect of each term more clearly, as they are independent of each other.

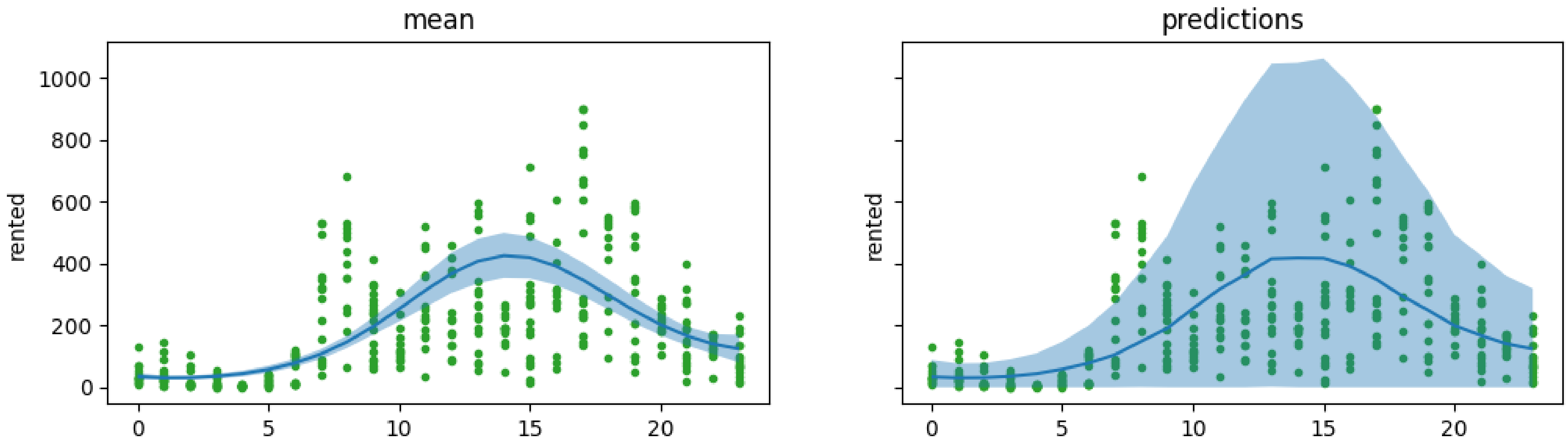
Polynomial Regression – Bikes Example

- Model the number of bikes rented on the hour of the day.
- Let's start with simple regression (only as a function of hour).



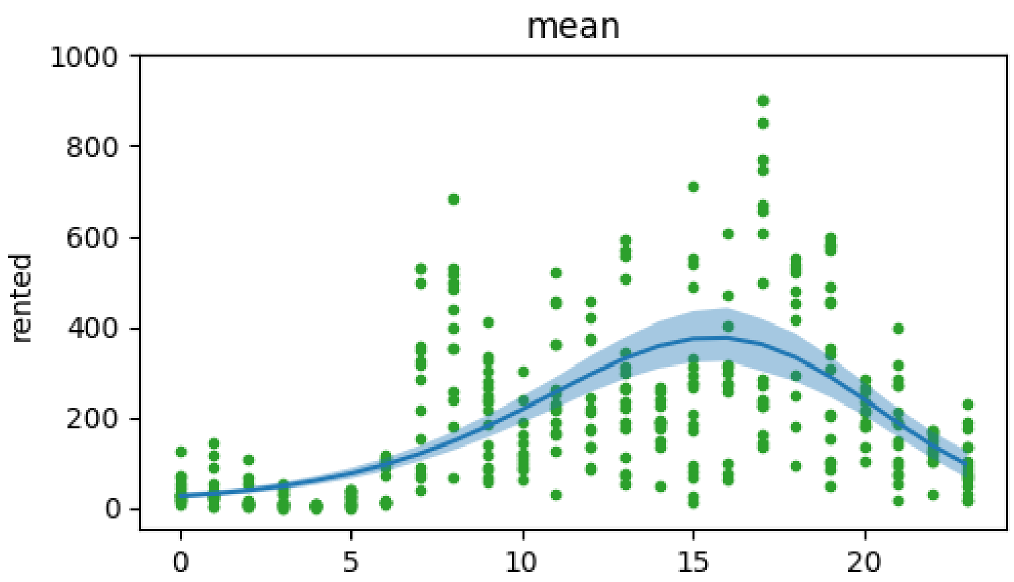
Polynomial Regression – Bikes Example

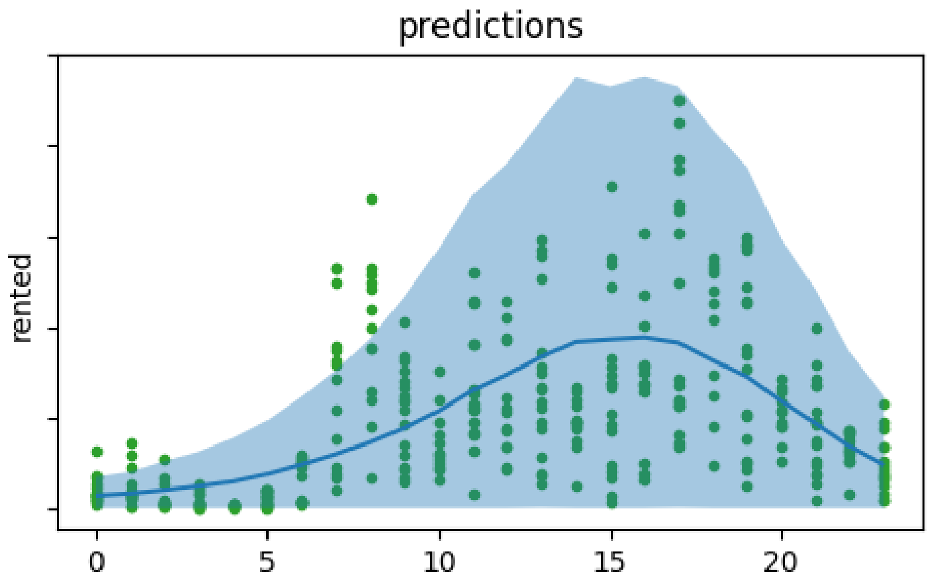
- Polynomial regression with orthogonal fourth-degree polynomial.



Polynomial Regression – Bikes Example

- Polynomial regression with standard fourth-degree polynomial.





Polynomial Regression – Bikes Example

- When we apply a polynomial of degree m, we are saying that the relationship between the independent and dependent variables is of degree m for the entire dataset.
- As the number of degrees of freedom increases, the model becomes more prone to overfitting.

Splines

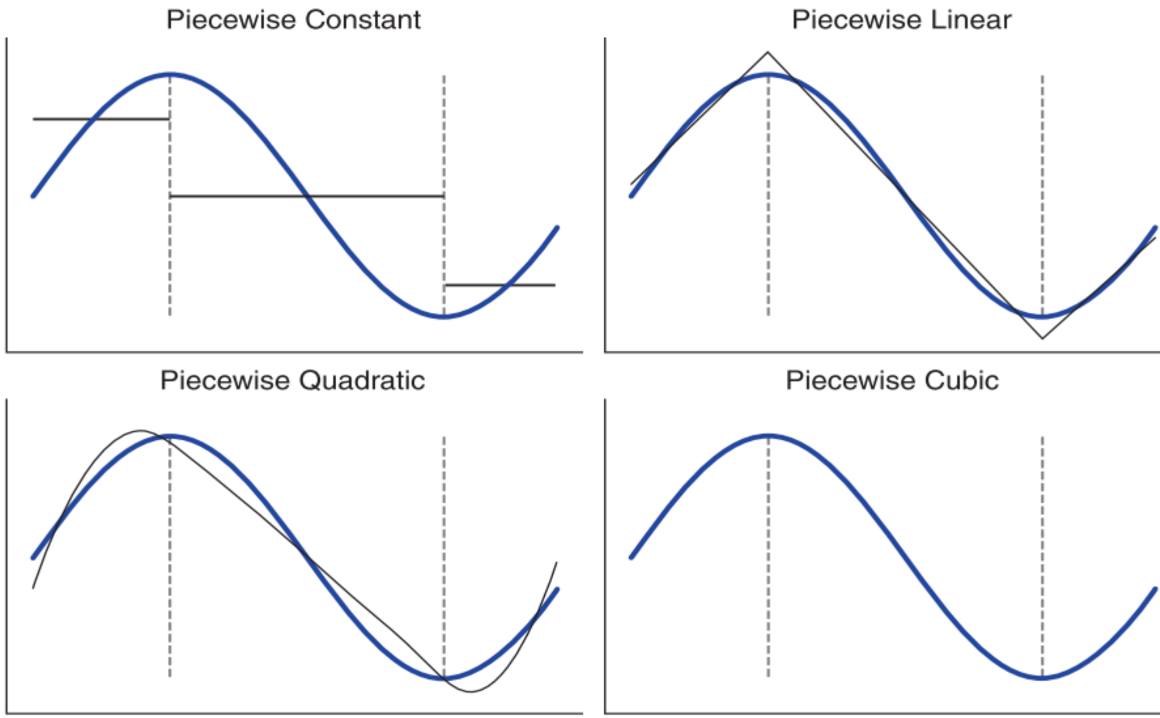
- A general way to write very flexible models is to apply functions B_m to X_m and then multiply them by coefficients β_m.

$$\mu = \beta_0 + \beta_1 B_1(X_1) + \beta_2 B_2(X_2) + \cdots + \beta_m B_m(X_m)$$

- A common choice is to use B-splines.
- Piece-wise polynomials.
  - Polynomials that are restricted to affect only a portion of the data.

Splines

- Let’s say we are approximate the blue curve.
- The black lines show the fit according to the degree of the polynomial.
- The dashed black lines show the knots - the points used to restrict the regions.



Splines

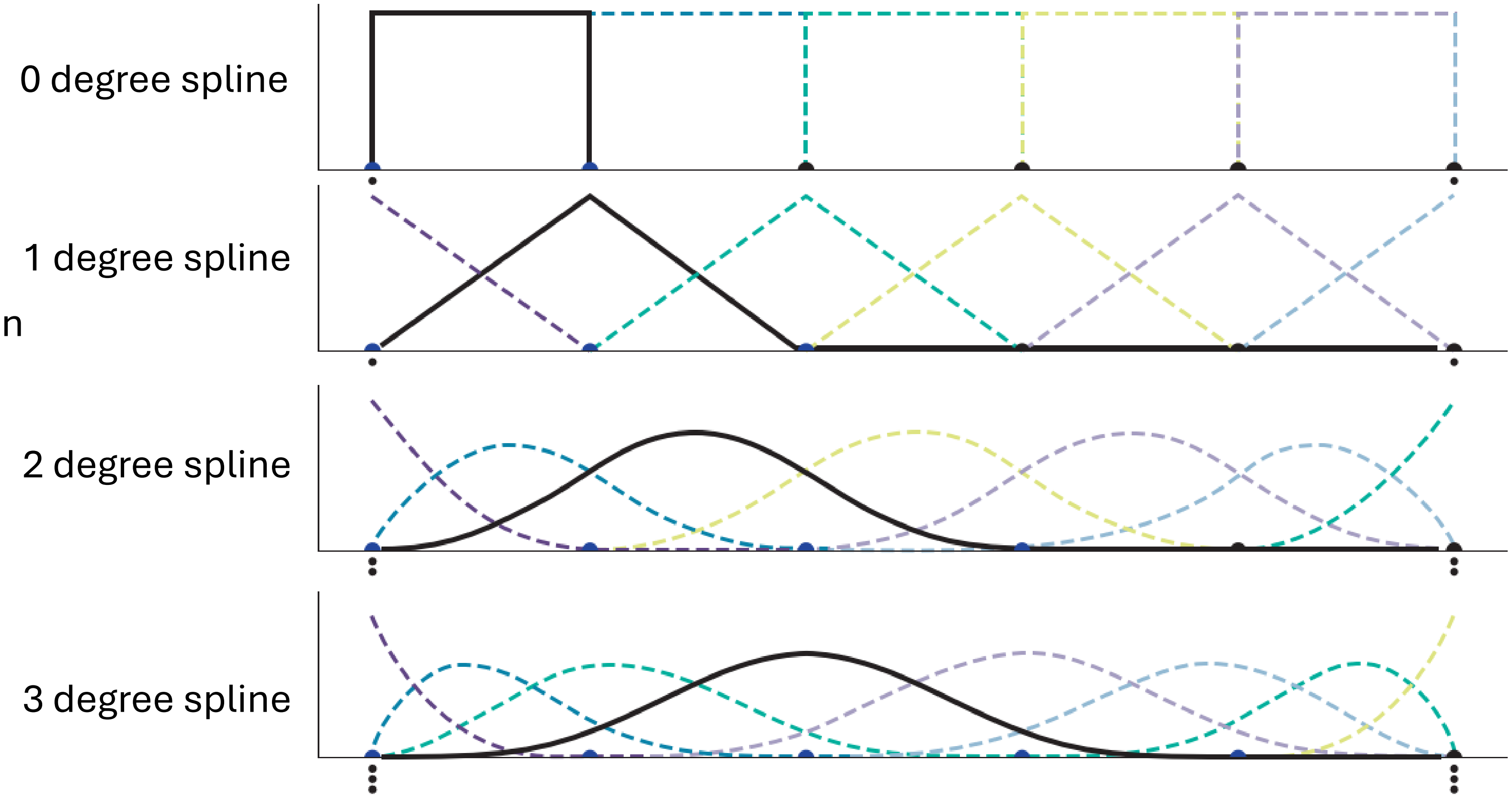
- How do we fit the spline?
  - We fit the polynomial of the relevant degree between each two knots (B-spline).
  - This leads to polynomials that are continuous, overlapping but also restricted to local areas.
  - At the beginning, each B-spline is weighted equally.
  - Then we multiply each by the coefficient β_m.
  - Lastly, we compute the weighted sum of the B-splines, giving us the spline.
  - We can use Bayesian statistics to find the proper weights for the B-splines.

Splines

- The B-splines.

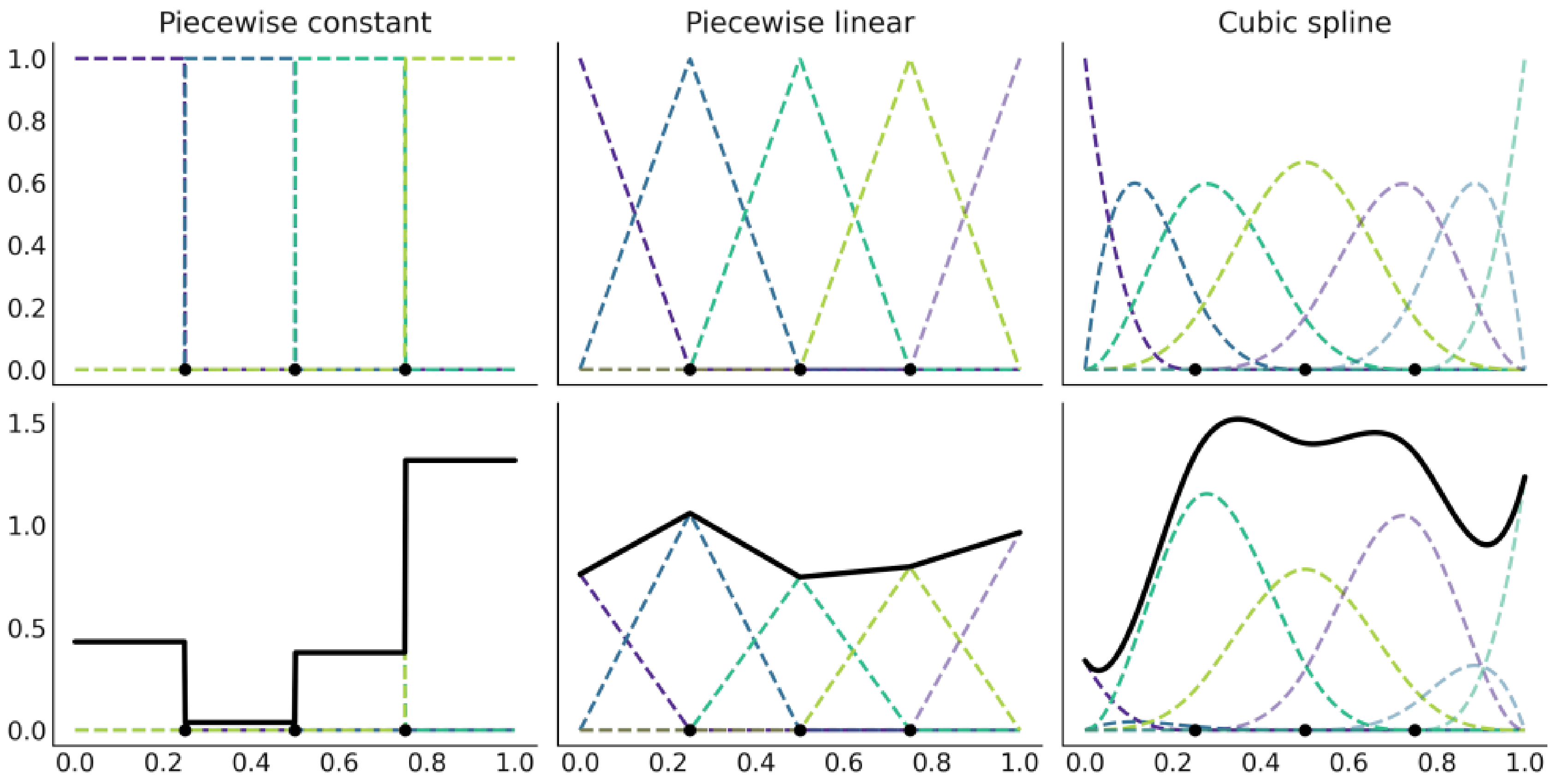
In a spline of degree n:

- The bases a polynomials of degree n
- n+1 bases overlap between knots
- n bases overlap at each knot



Splines

- The spline.

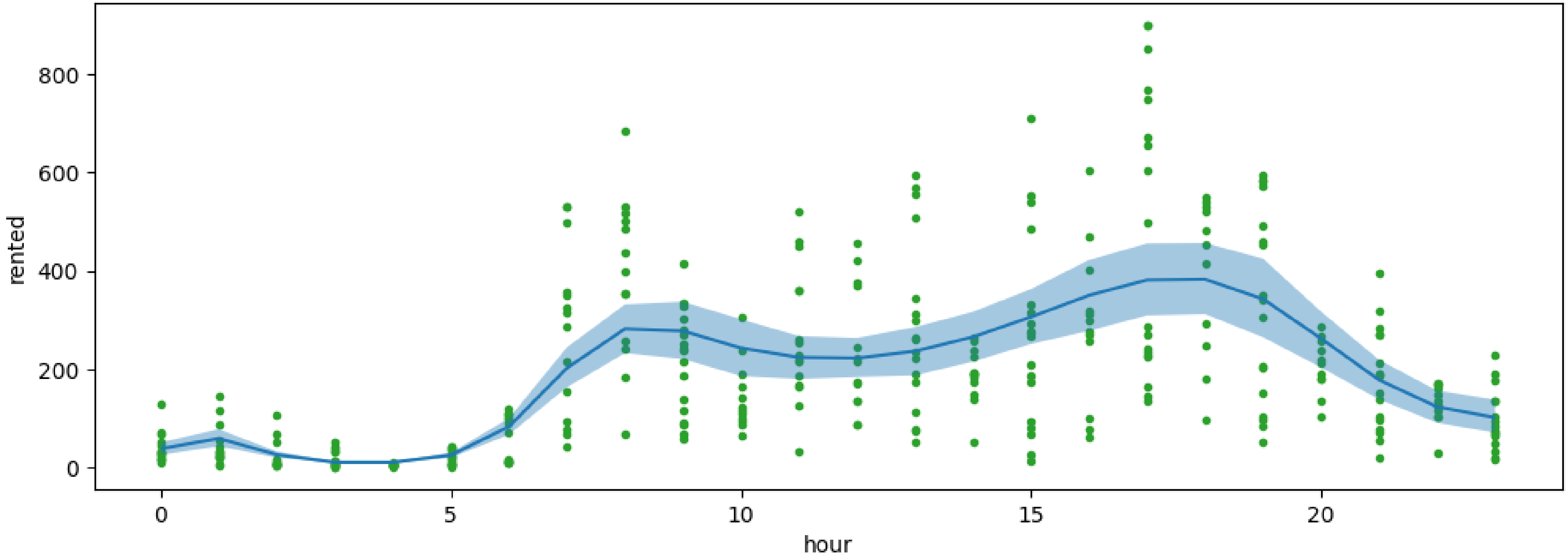


Splines – Bike Example

- Using Bambi to fit a spline to the bike data:

```
num_knots = 6
knots = np.linspace(0, 23, num_knots+2)[1:-1]
model_spline = bmb.Model("rented ~ bs(hour, degree=3, knots=knots)", bikes, family="negativebinomial")
idata_spline = model_spline.fit(1000, chains = 4)
```

- This yields:



Splines – Bike Example

- We can see that the number of rental bikes is at the lowest number late at night.
- There is then an increase, probably as people wake up and go to work or school, or do other activities.
- We have a first peak at around hour 8, then a slight decline, followed by the second peak at around hour 18, probably because people commute back home, after which there is a steady decline.

Splines – Bike Example

- How should we choose where to place the knots?
  - We can choose based on quantiles rather than uniformly.
  - This positions more knots in areas where we have a greater amount of data, while placing fewer knots in areas with less data.