pyGAM

balancing predictive power and interpretability with generalized additive models

dani servén marín

Who are you?

- twin
- Spain + USA
- <3
  - Full-stack data (ml, r+d, deploying...)
  - Society + environment
  - learning
  - open source
  - d.i.y
- core contributor to pyGAM

github.com/dswah/

What's pyGAM?

New open-source library for estimating Generalized Additive Models.

- Modular
- Pure Python
- Plays nice with PyData tools

github.com/dswah/pygam

Compatible with sklearn

```
# do regression
gam = LinearGAM().fit(X, y)
y_pred = gam.predict(X)


# or do classification
lgam = LogisticGAM().fit(X, y)
lgam.accuracy(X, y)


# build a custom model
custom_gam = GAM(link='log', distribution='poisson').fit(X, y)
```

Why did you make pyGAM?

- No P-spline GAMs in Python
- Contribute to PyData stack

Why did you make pyGAM?

Human learning:

- Statistics
- Code quality
- Open-source

Not enough time...

Regression

Confidence intervals

Prediction intervals

Partial dependence

Model statistics

Classification

Custom GAMs

Sampling from fitted models

Extrapolation

Monotonic / Convex / Concave constraints

Histogram smoothing

Weighted data

Sklearn RandomizedSearchCV

Alternate optimization objectives

Optimizer Callbacks

Expectiles

Not enough time...

Regression

Confidence intervals

Prediction intervals

Partial dependence

Model statistics

Classification

Custom GAMs

Sampling from fitted models

Extrapolation

Monotonic / Convex / Concave constraints

Histogram smoothing

Weighted data

Sklearn RandomizedSearchCV

Alternate optimization objectives

Optimizer Callbacks

Expectiles

Welcome to PyDatatopolis

Need to generate power for our self-sufficient city

- Plan tomorrow's hourly power demand
- too low => blackouts
- too high => waste resources
- Data Driven™

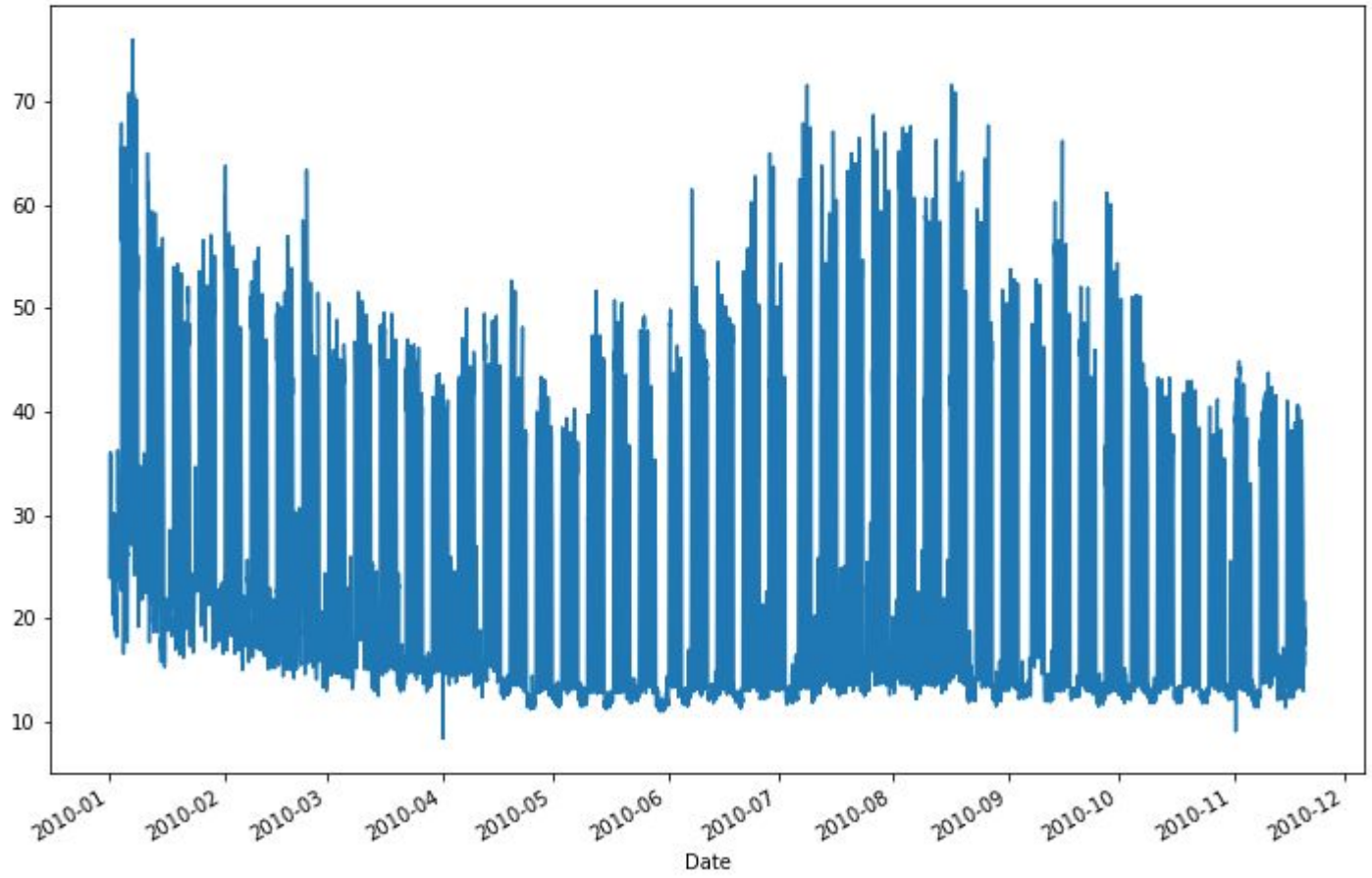
Welcome to PyDatatopolis

```
df.columns
```

```
Index(['OAT', 'Building 6 kW', 'Cloud Cover'], dtype='object')
```

```
df['Building 6 kW'].plot(figsize=(12,8))
```

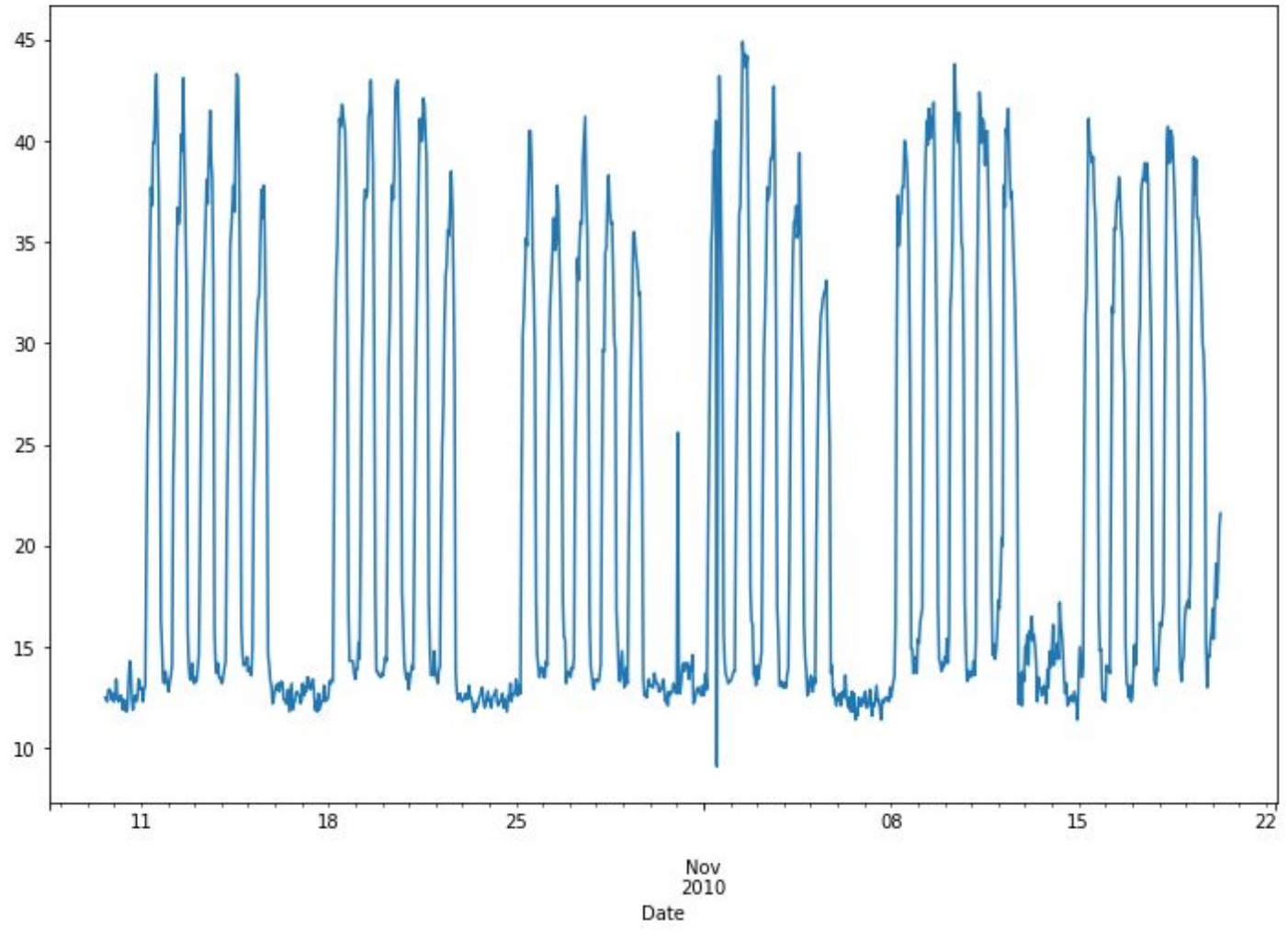
```
<matplotlib.axes. subplots.AxesSubplot at 0x7f8aa72d57b8>
```



Welcome to PyDatatopolis

```
df['Building 6 kW'][-1000:].plot(figsize=(12,8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feeaac56b00>
```



Welcome to PyDatatopolis

- Build LSTM / ARIMAX / kNN / ... model
- Good fit
- Show your comrades the prediction for tomorrow
- But they are skeptical...

Welcome to PyDatatopolis

- How certain are you?
- It's going to be a hot summer
  - how should we prepare?...
- What happens on sunny days?
- etc?

Welcome to PyDatatopolis

You survey your tool box...

- Linear models have too much bias
- Powerful models are black boxes
- ARIMAX just gives you coefficients
- ...

Welcome to PyDatatopolis

You take a break...

read a blog post about GAMs

Welcome to PyDatatopolis

You take a break...

read a blog post about GAMs

so you try it out…

```
pip install pygam
```

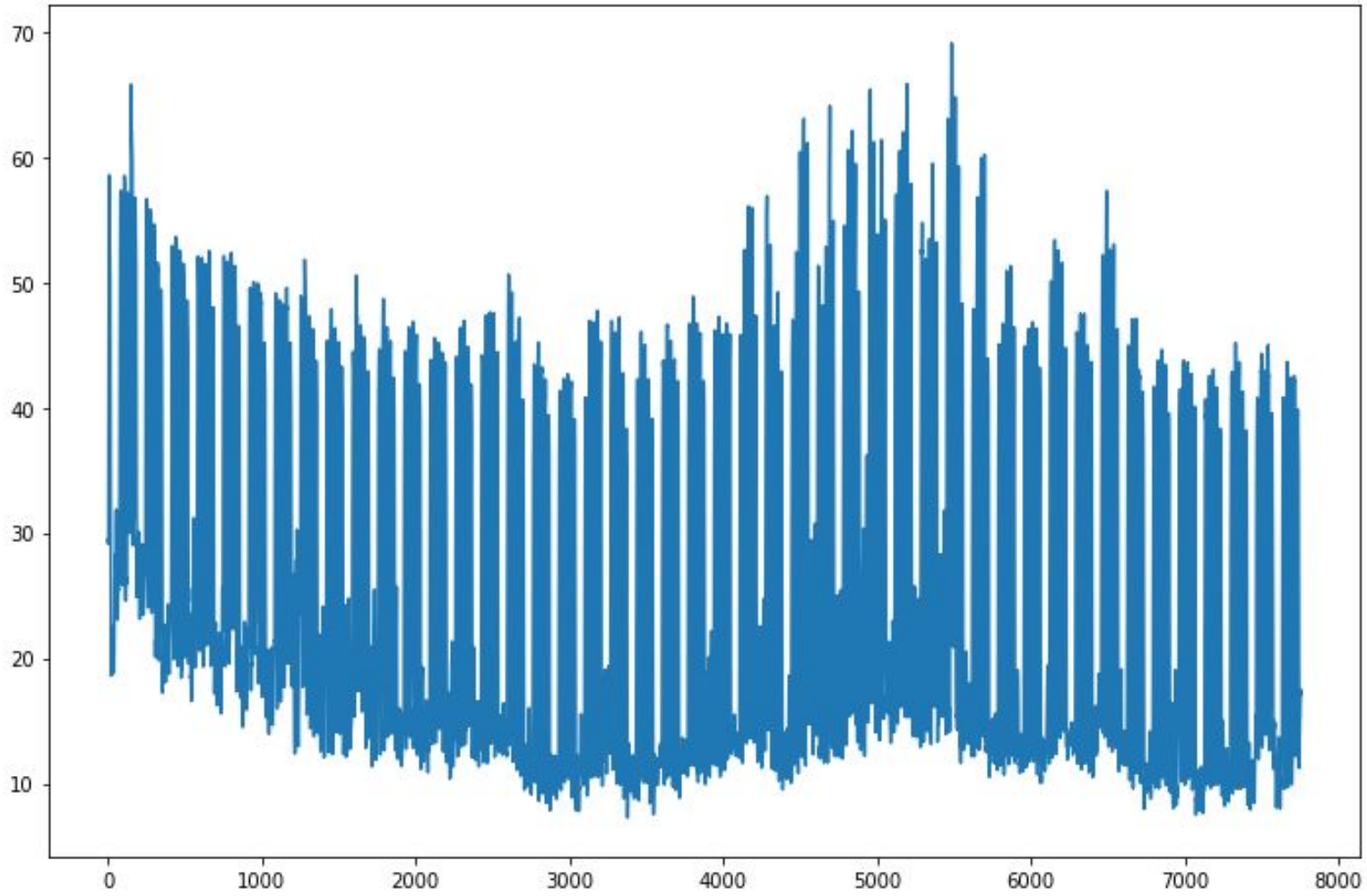
Welcome to PyDatatopolis

```
X = df[['day_of_week', 'time_of_year', 'OAT', 'Cloud Cover']].values
y = df['Building 6 kW'].values

from pygam import LinearGAM
gam = LinearGAM().fit(X, y)
```

Welcome to PyDatatopolis

```
plt.plot(gam.predict(X))
```

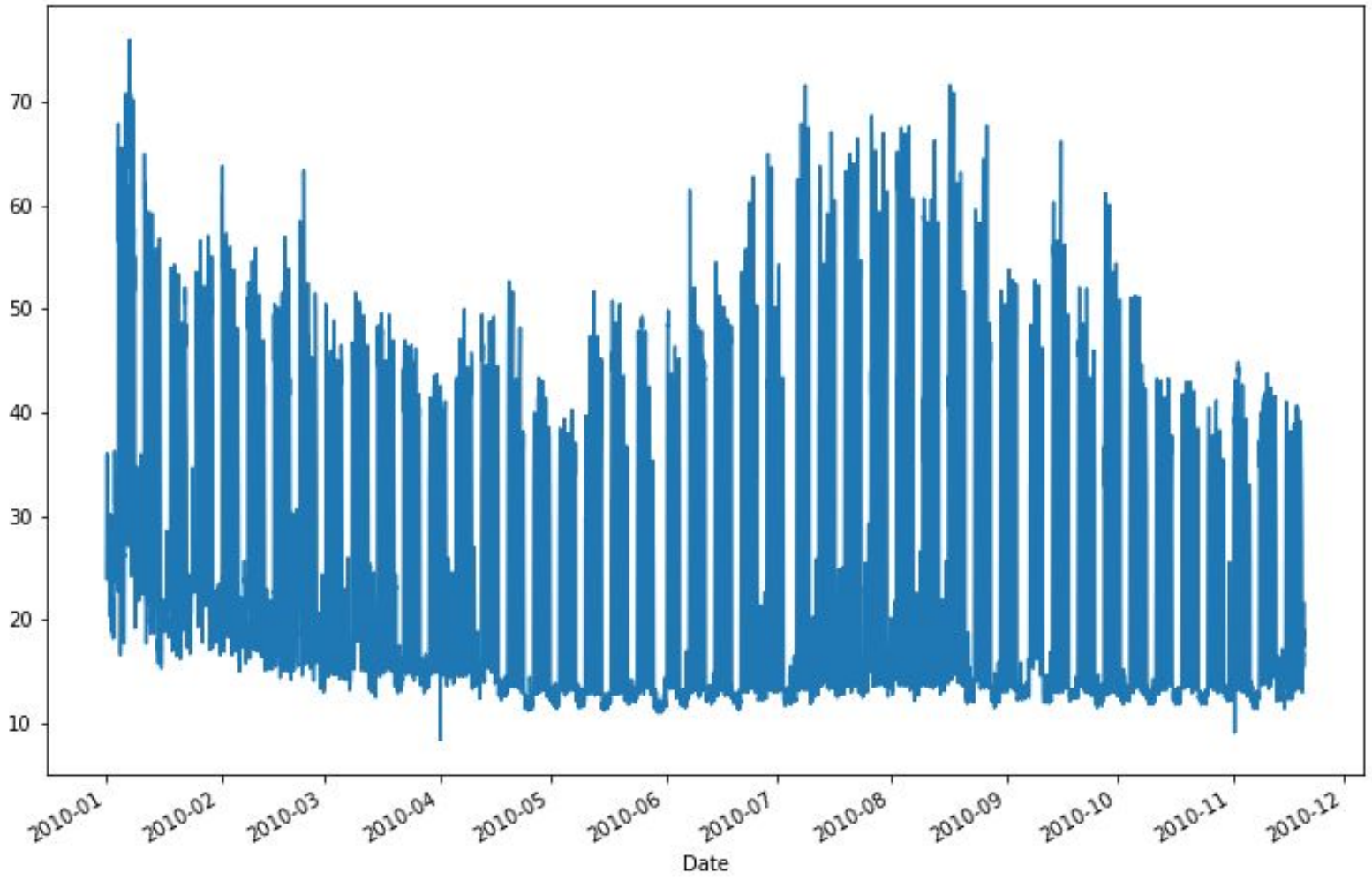
```
[<matplotlib.lines.Line2D at 0x7feea8128ba8>]
```



Welcome to PyDatatopolis

```
df['Building 6 kW'].plot(figsize=(12,8))
```

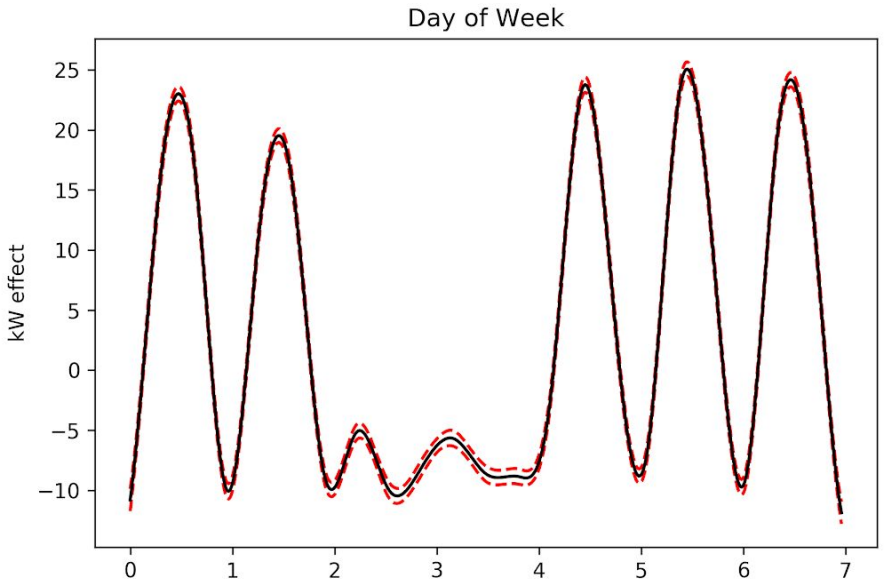
```
<matplotlib.axes. subplots.AxesSubplot at 0x7f8aa72d57b8>
```

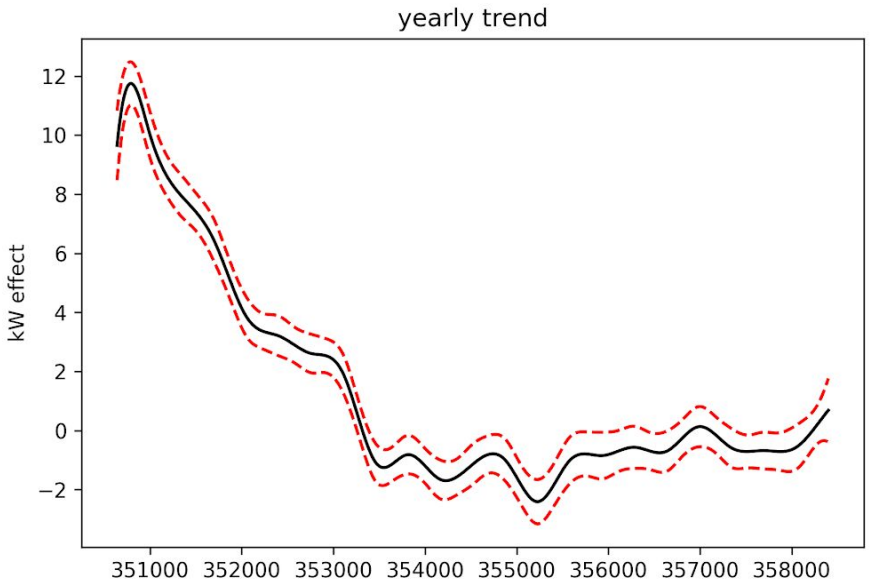


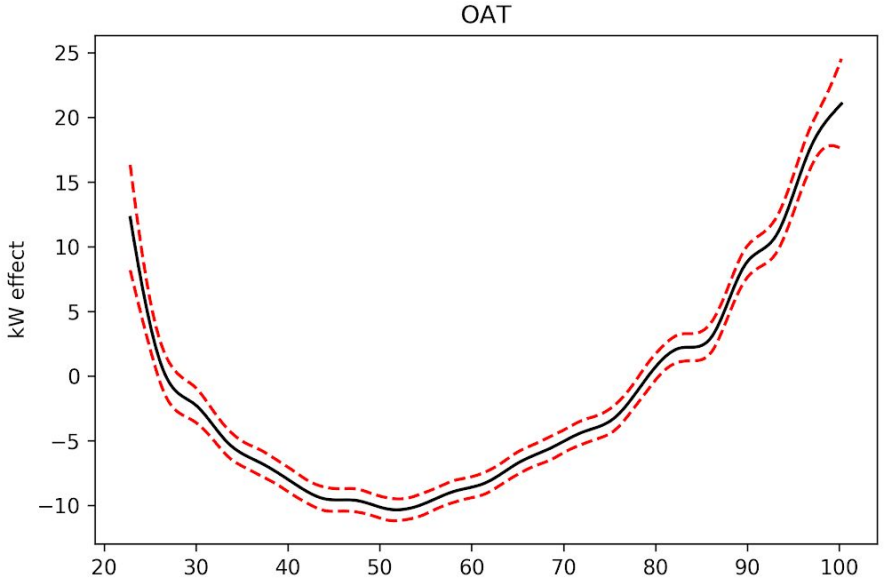
What's pyGAM?

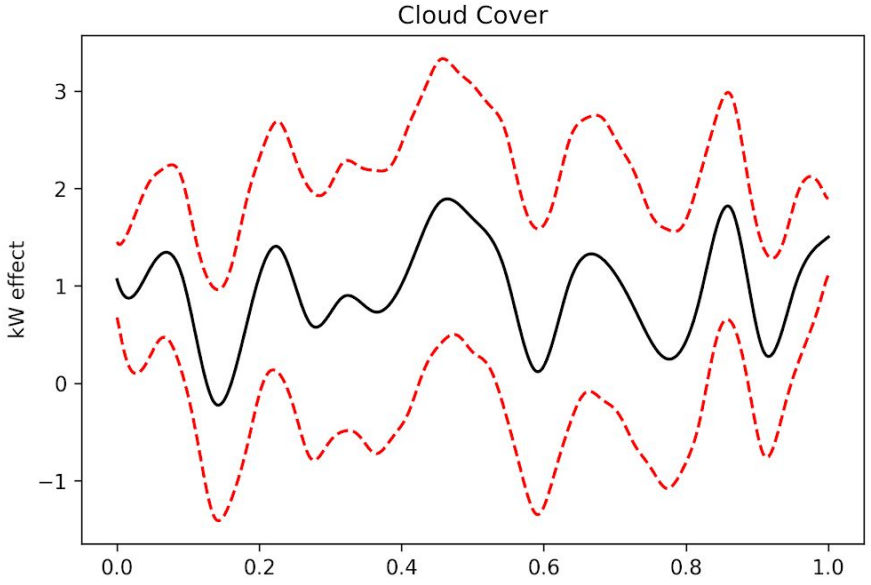
```
pdeps, confi = gam.partial_dependence(width=0.95)
```

Welcome to PyDatatopolis





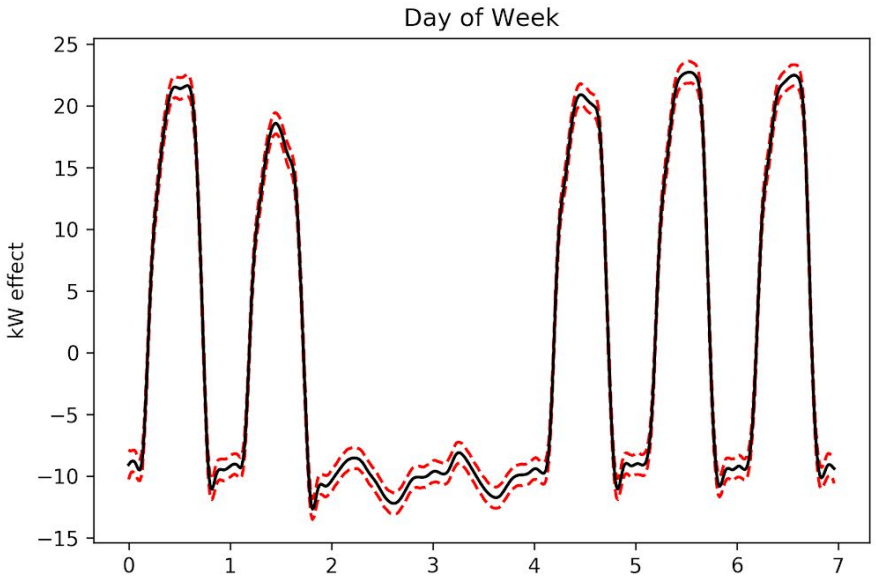


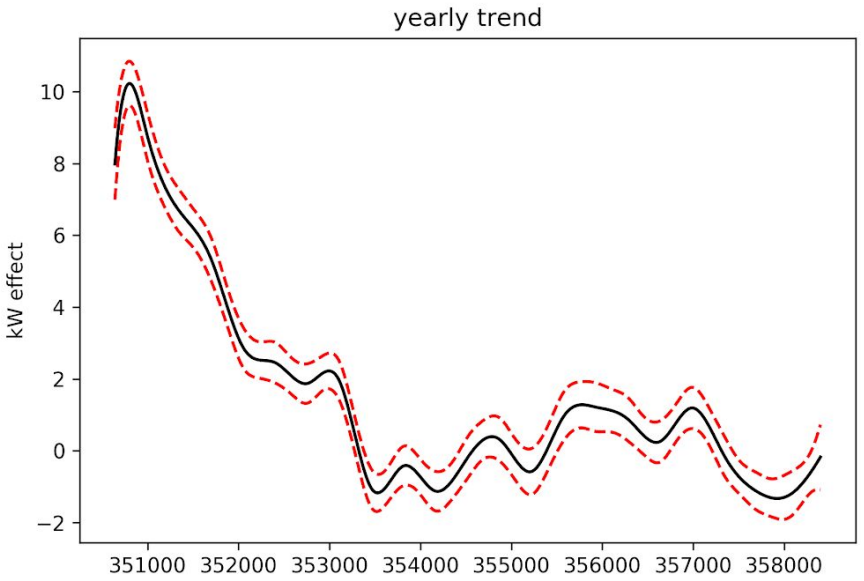


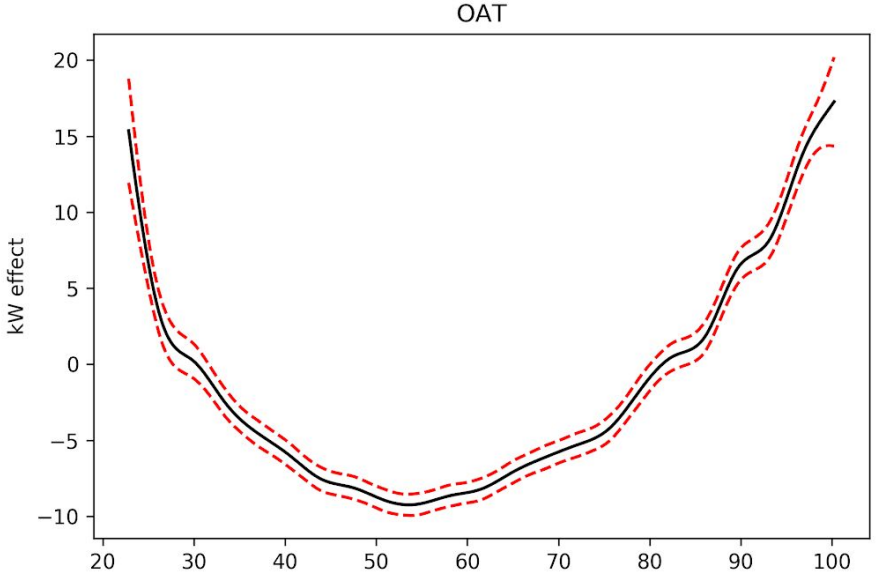
Welcome to PyDatatopolis

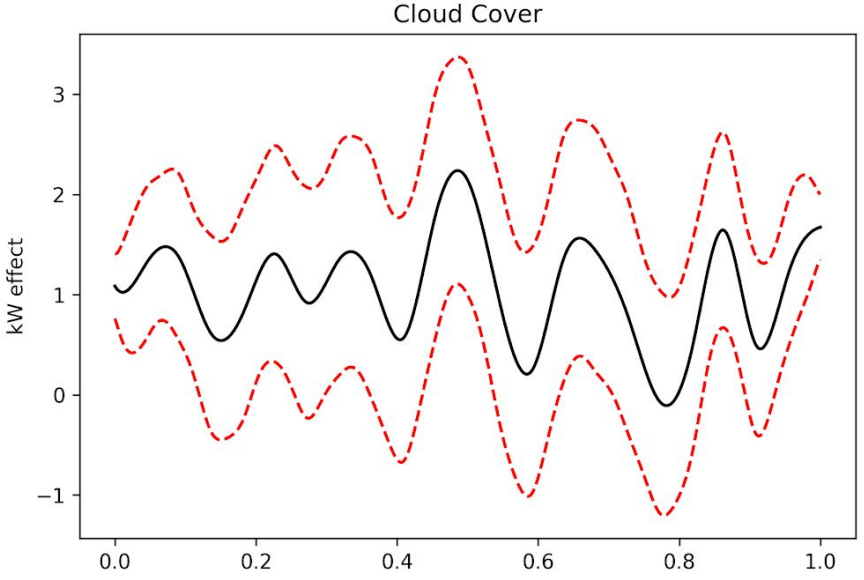
```
gam = LinearGAM(n_splines=[100, 25, 25, 25]).fit(X, y)
```

Welcome to PyDatatopolis





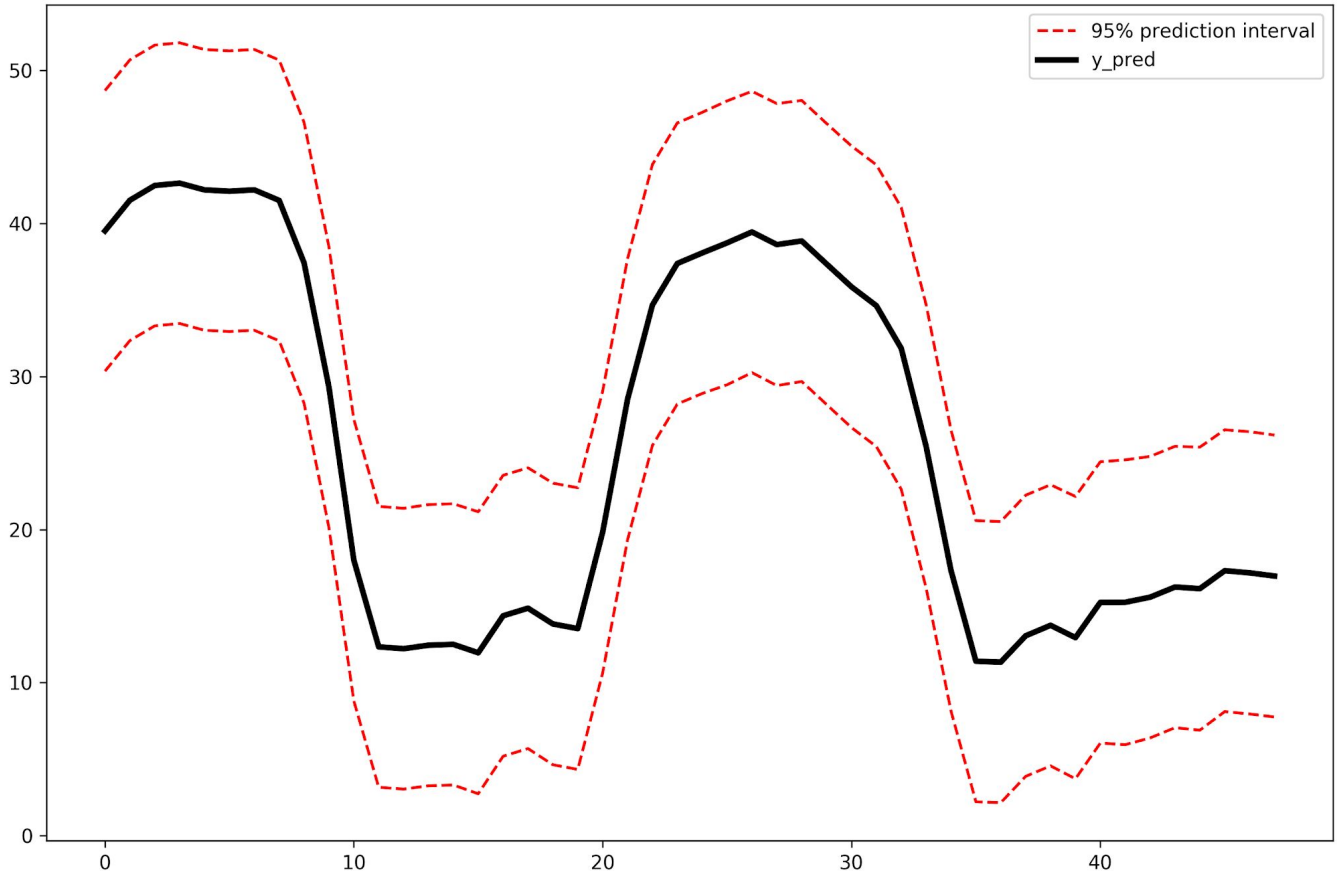




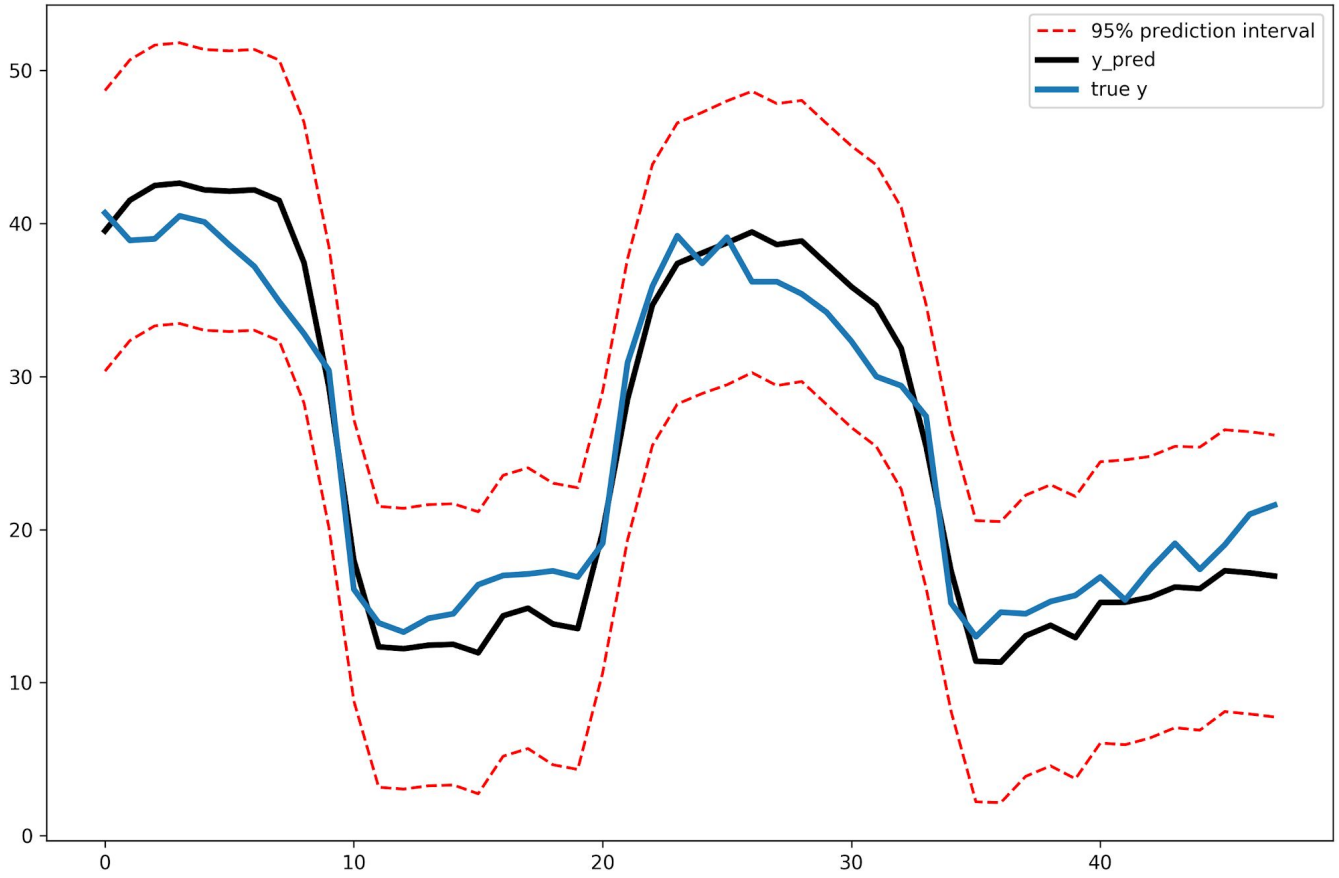
Welcome to PyDatatopolis

```
y_pred = gam.predict(X_pred)
predi = gam.prediction_intervals(X_pred, width=0.95)

plt.plot(predi, c='r', ls='--')
plt.plot(y_pred, c='k')
```

Welcome to PyDatatopolis



Welcome to PyDatatopolis



Welcome to PyDatatopolis

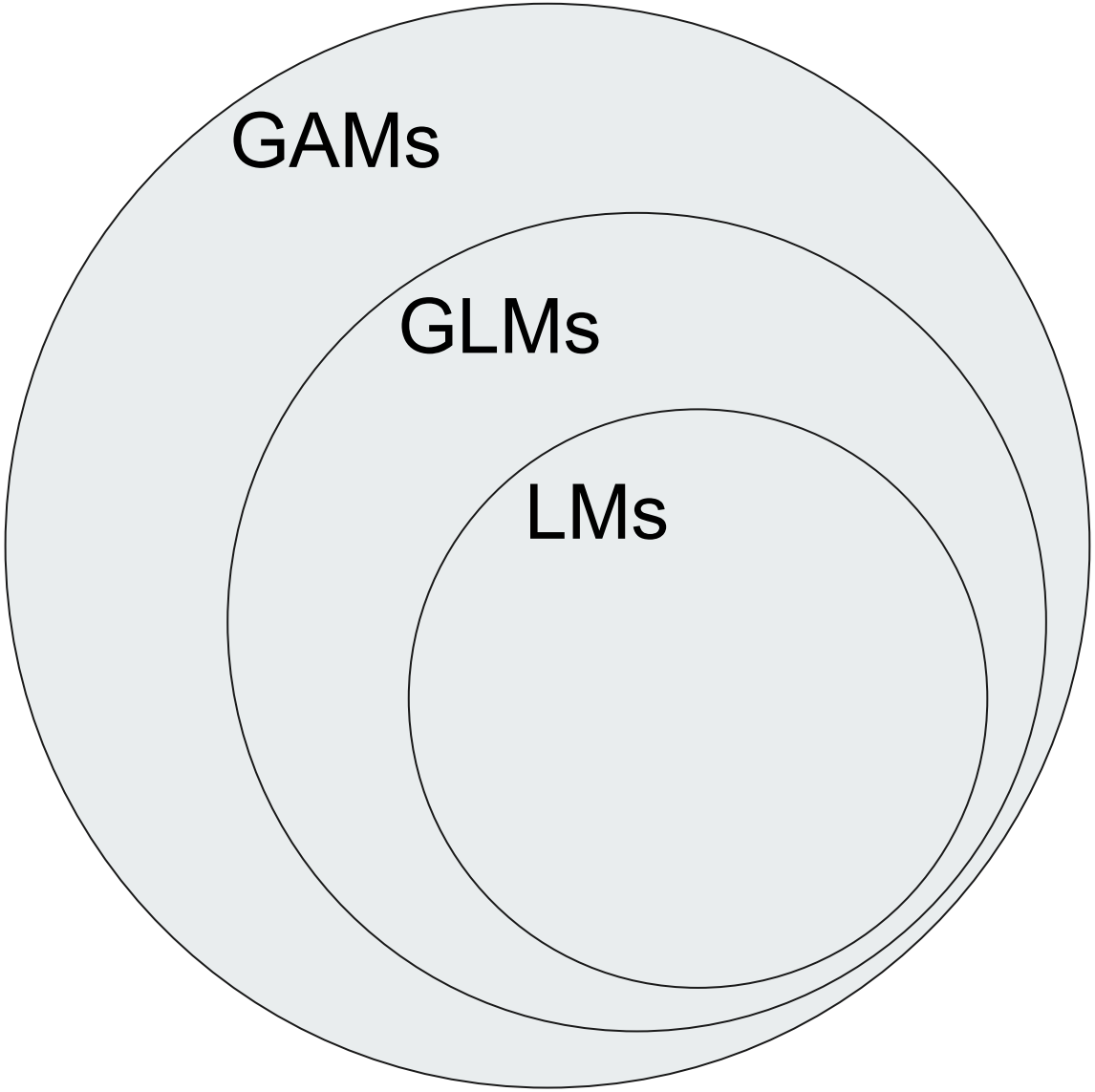
Open questions…

- How do we choose the smoothness?
- Feature selection?

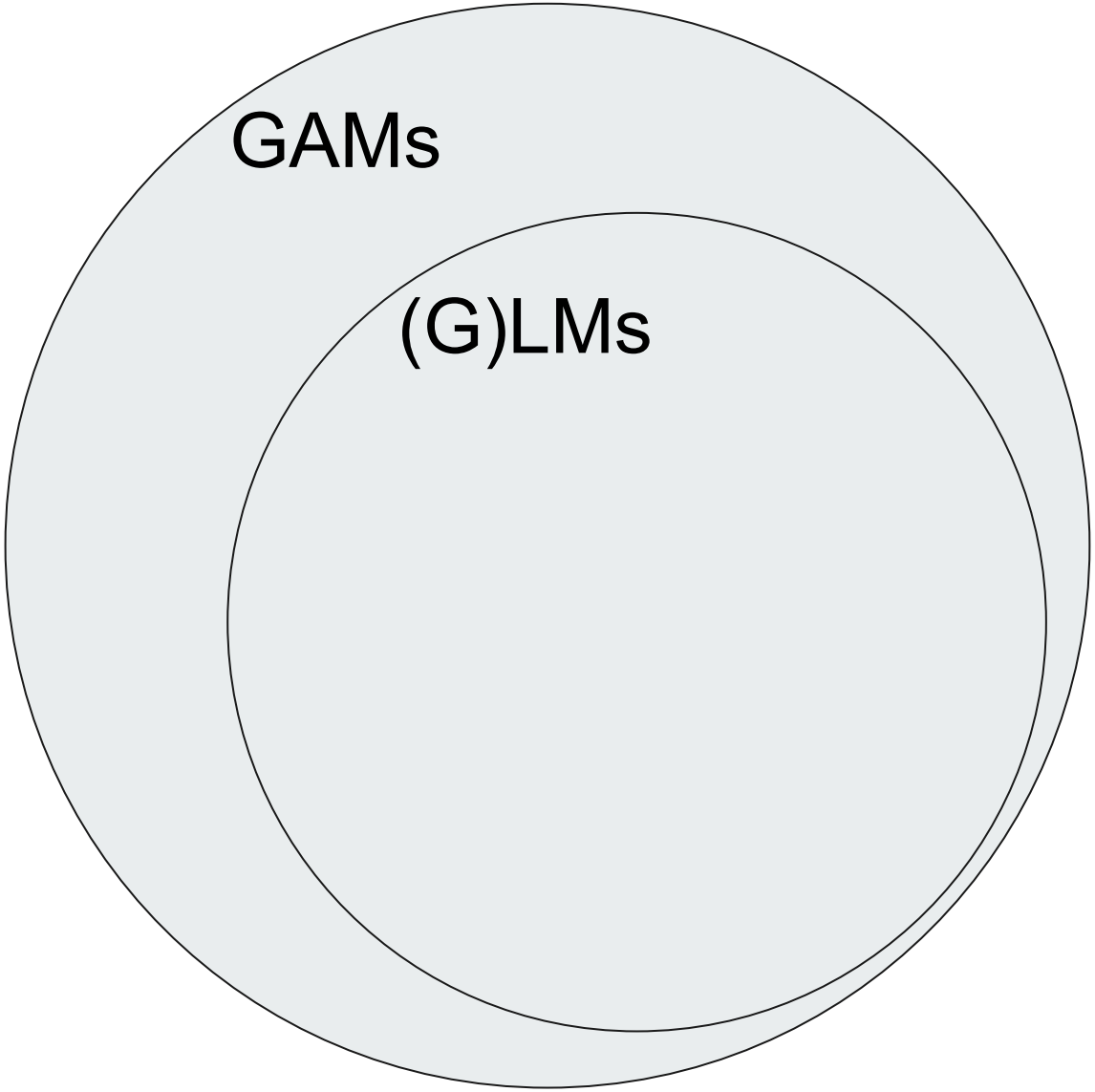
Plan

- What are GAMs
- A brief tour or splines
- Smoothing

What are GAMs

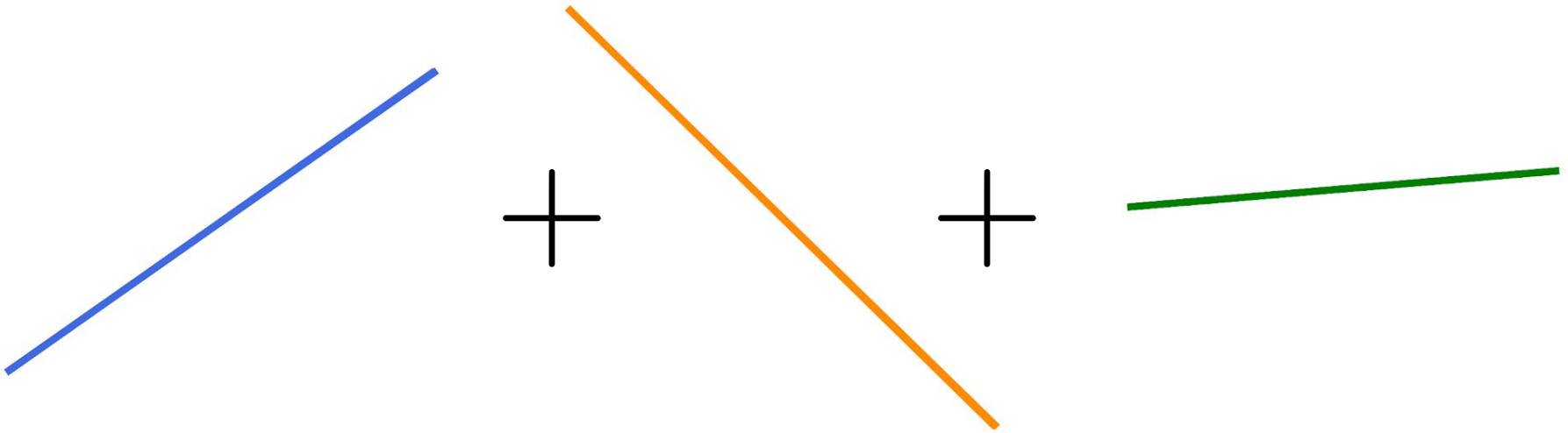


What are GAMs



What are GAMs

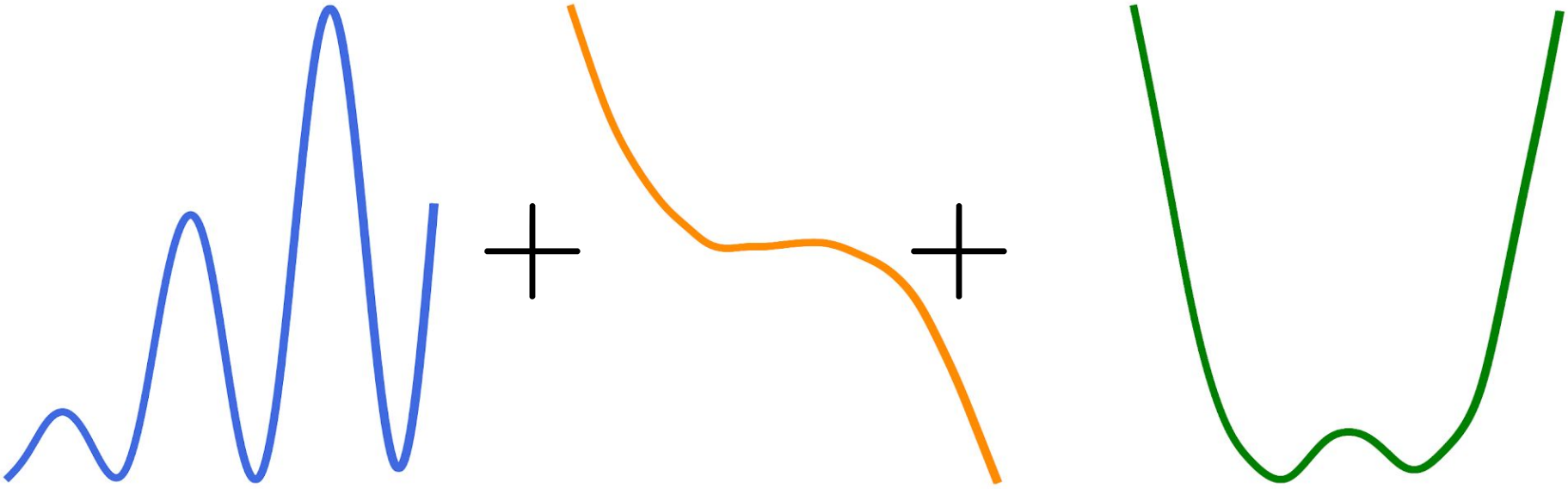
Linear Model



$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + c$$

What are GAMs

GAM



$$y = f_1(x_1) + f_2(x_2) + f_3(x_3) + c$$

What are GAMs

In general...

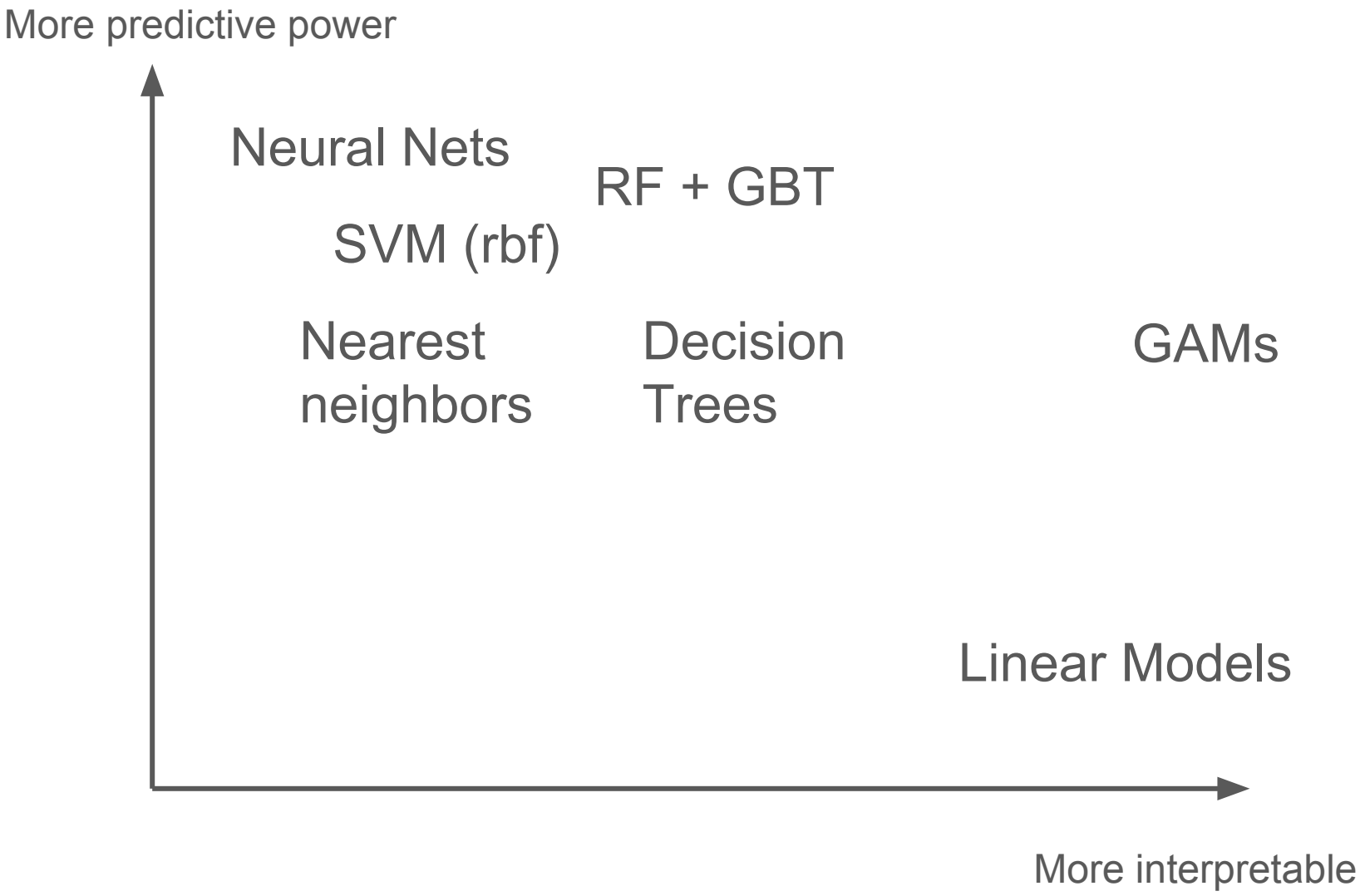
$$y = f_1(x_1) + f_2(x_2, x_3) + f_3(x_4) + \cdots + c$$

What are GAMs

but for us...

$$y = f_1(x_1) + f_2(x_2) + f_3(x_3) + c$$

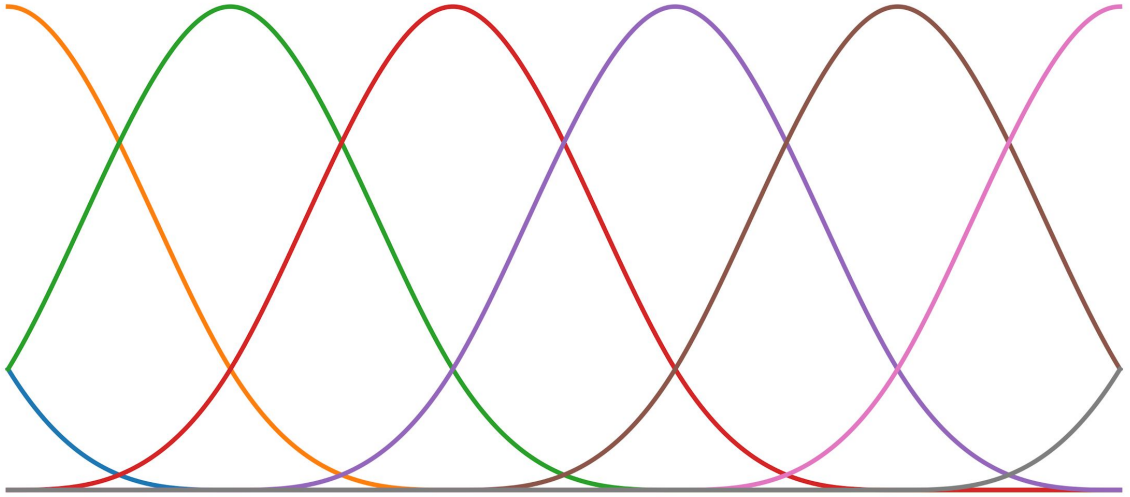
What are GAMs



What are GAMs

GAMs = GLMs

\+



\+ **Smoothing Penalty**

Plan

- What are GAMs
- A brief tour or splines
- Smoothing
- GAMs in Python

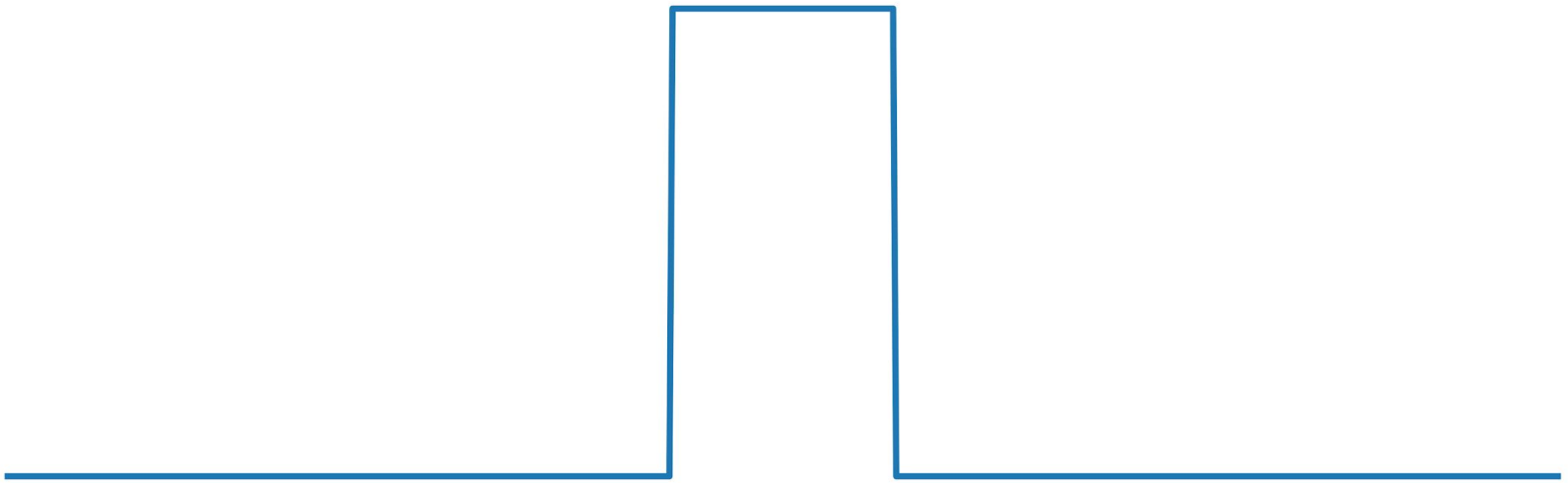
Splines

- Local support
- degree n spline has:
  - n + 1 segments of degree n
  - Continuous $(n-1)^{th}$ derivative

Splines

degree 0

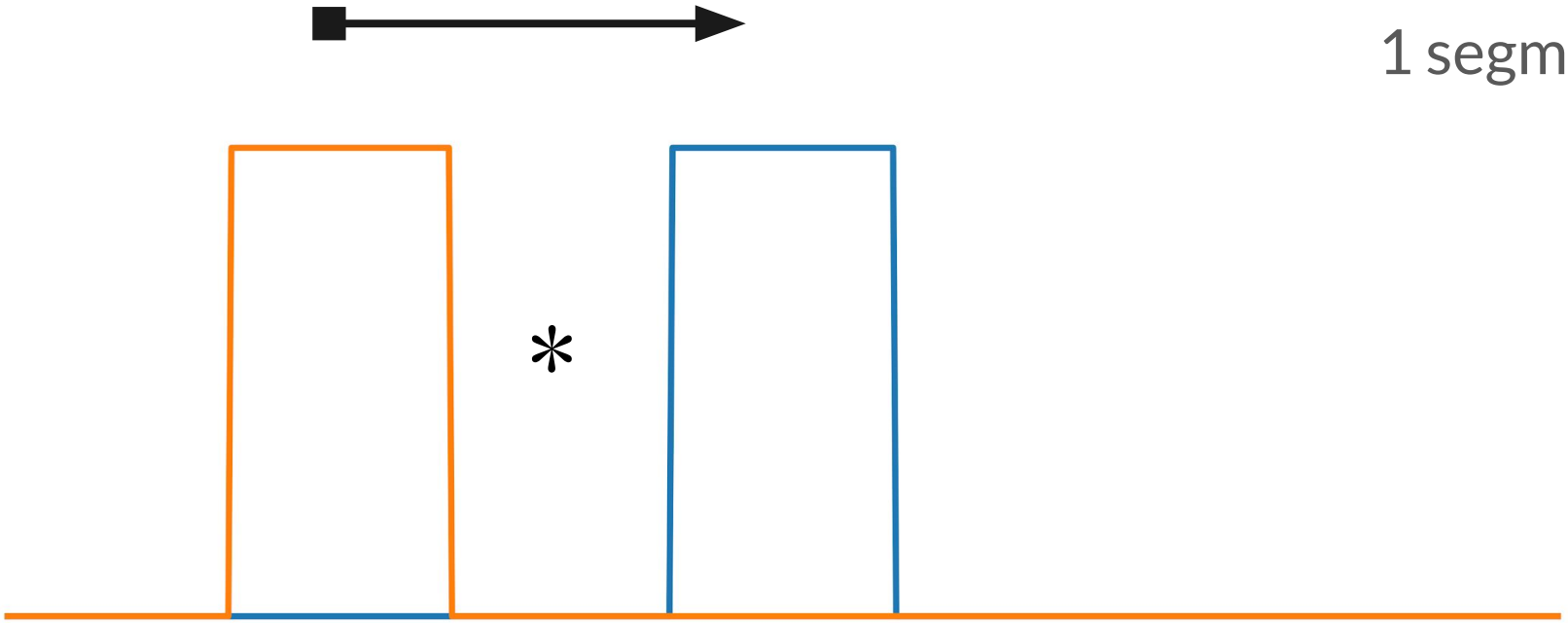
1 segment



Splines

degree 0

1 segment

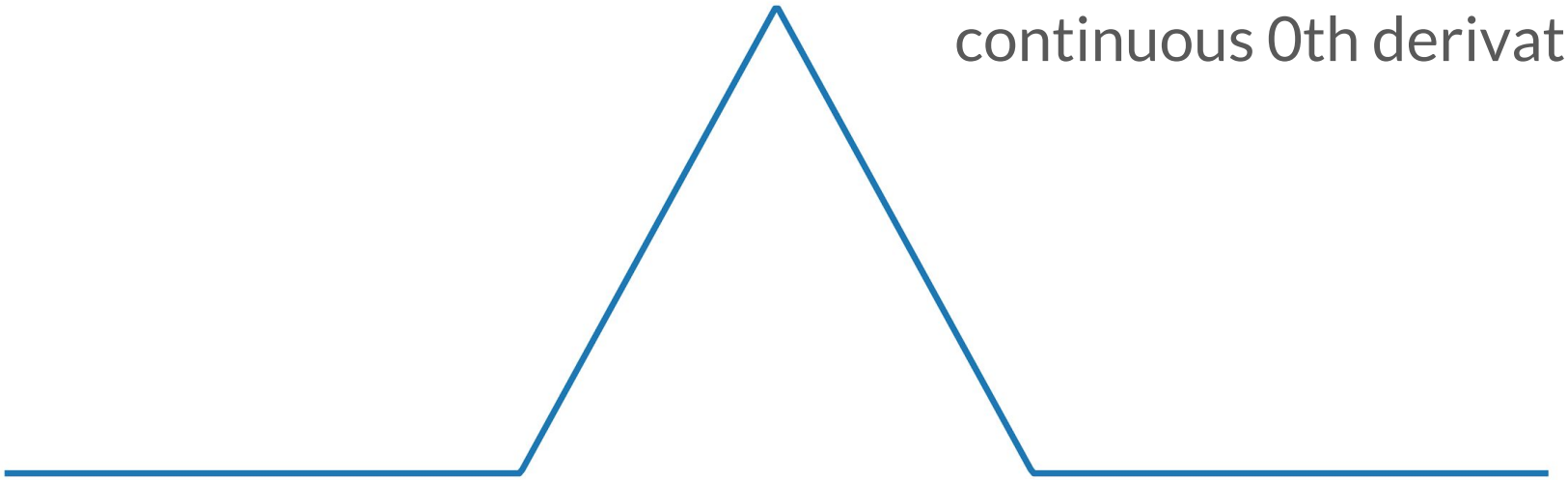


Splines

degree 1

2 segments

continuous 0th derivative

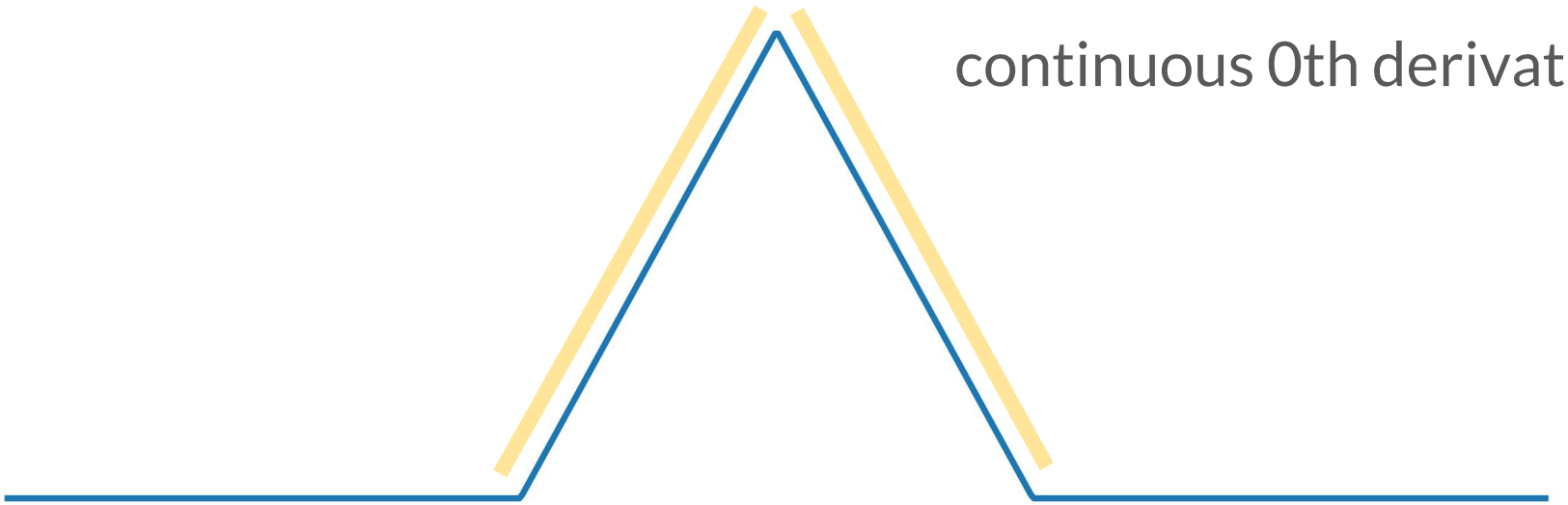


Splines

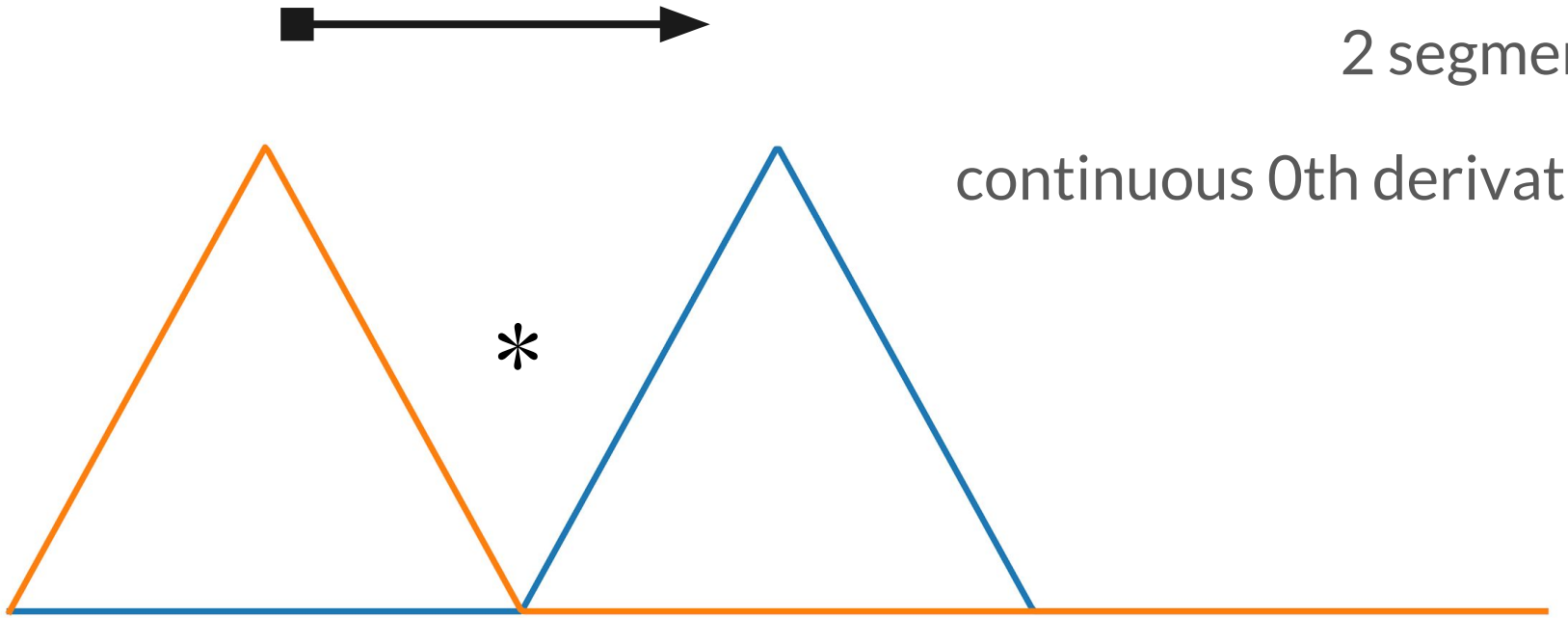
degree 1

2 segments

continuous 0th derivative



Splines



degree 1

2 segments

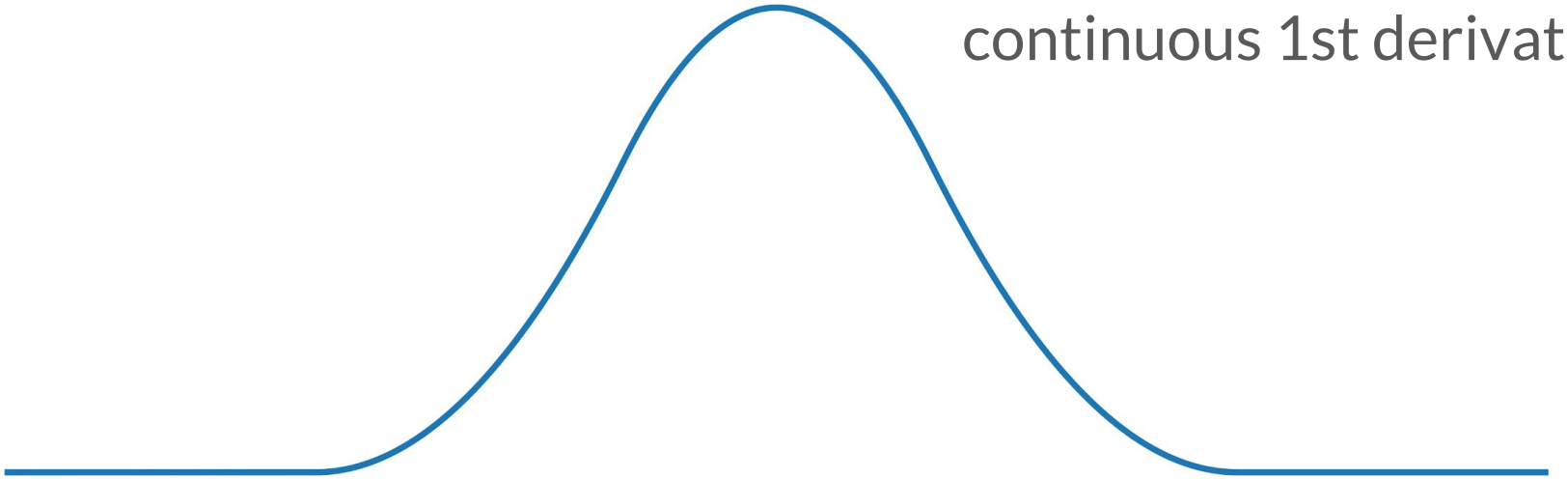
continuous 0th derivative

Splines

degree 2

3 segments

continuous 1st derivative

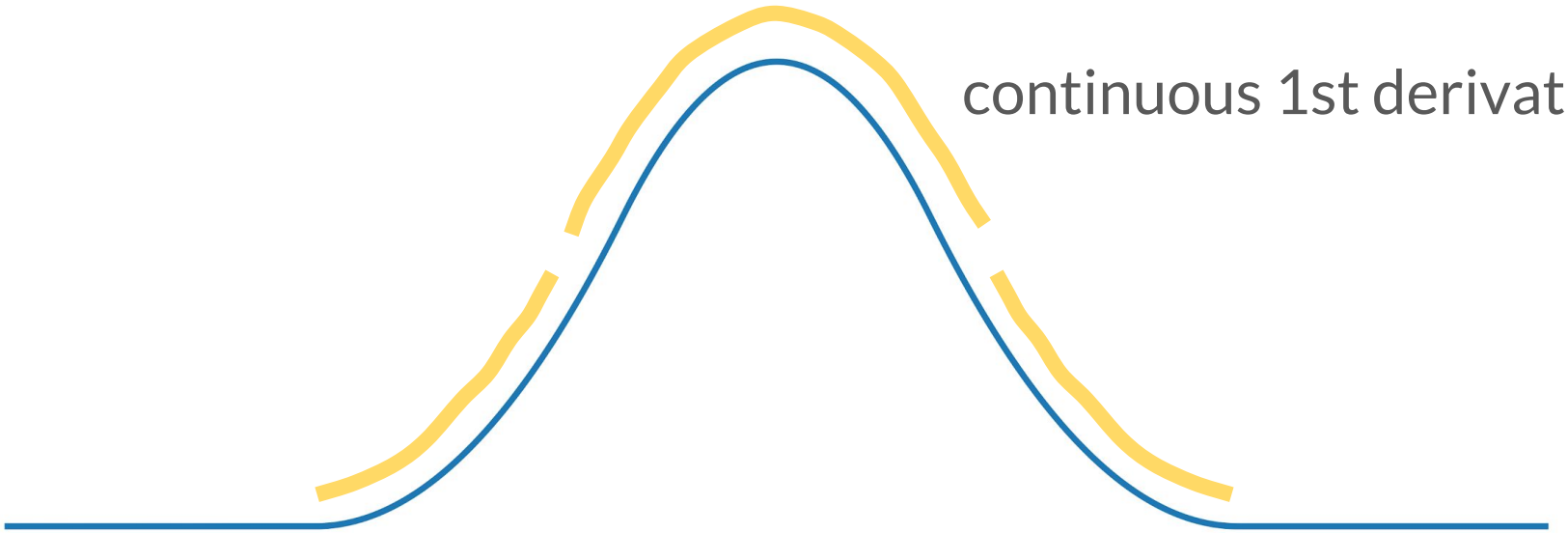


Splines

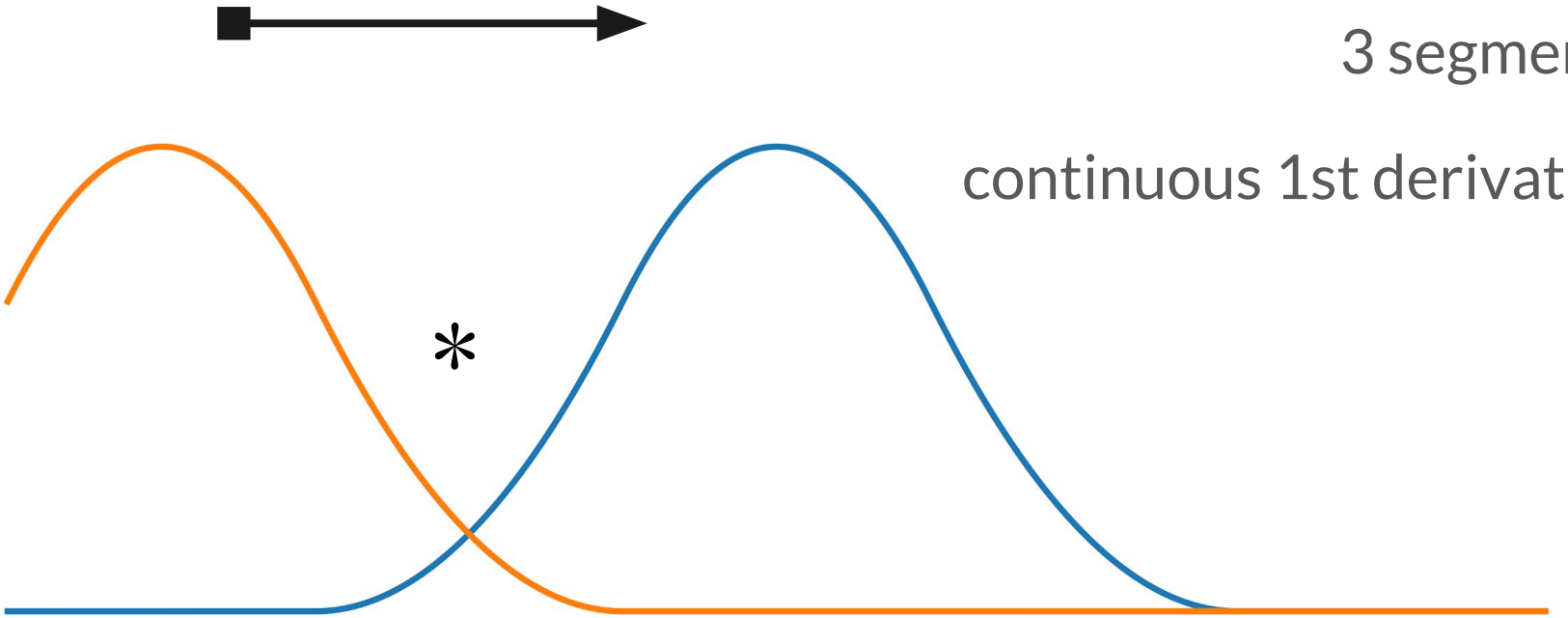
degree 2

3 segments

continuous 1st derivative



Splines



degree 2

3 segments

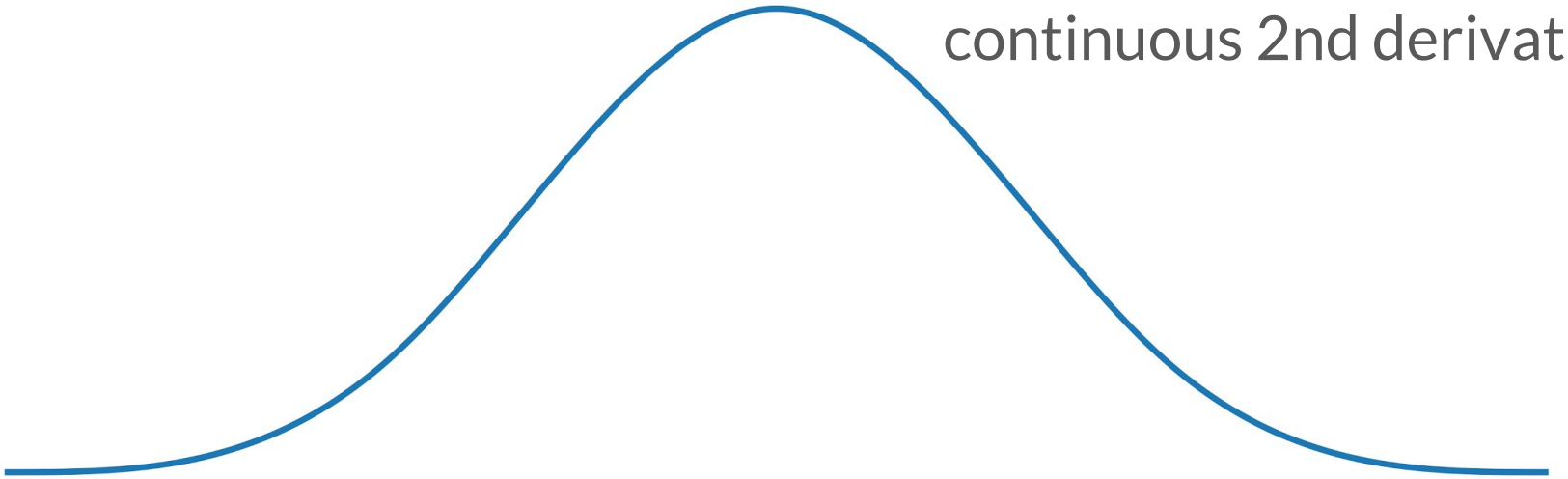
continuous 1st derivative

Splines

degree 3

4 segments

continuous 2nd derivative



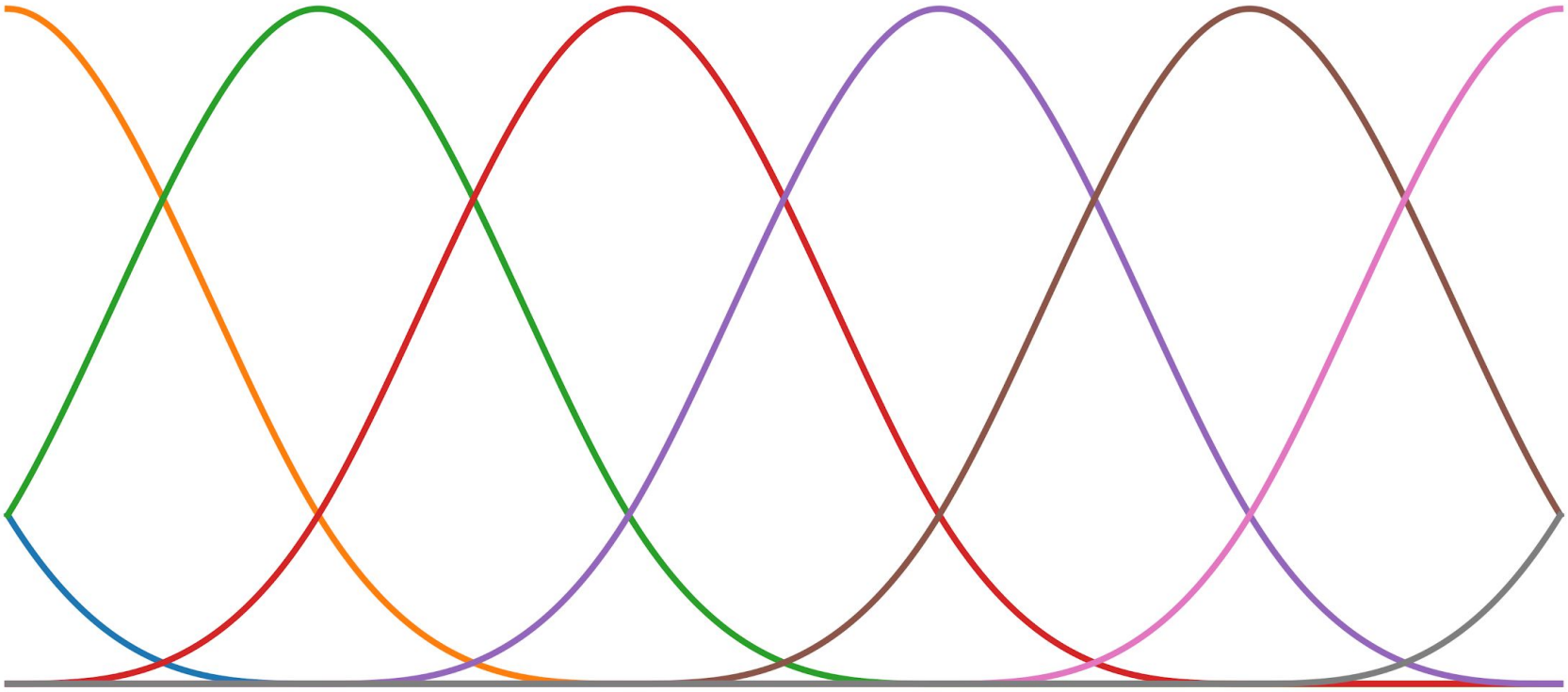
Splines

Spline -> Gaussian

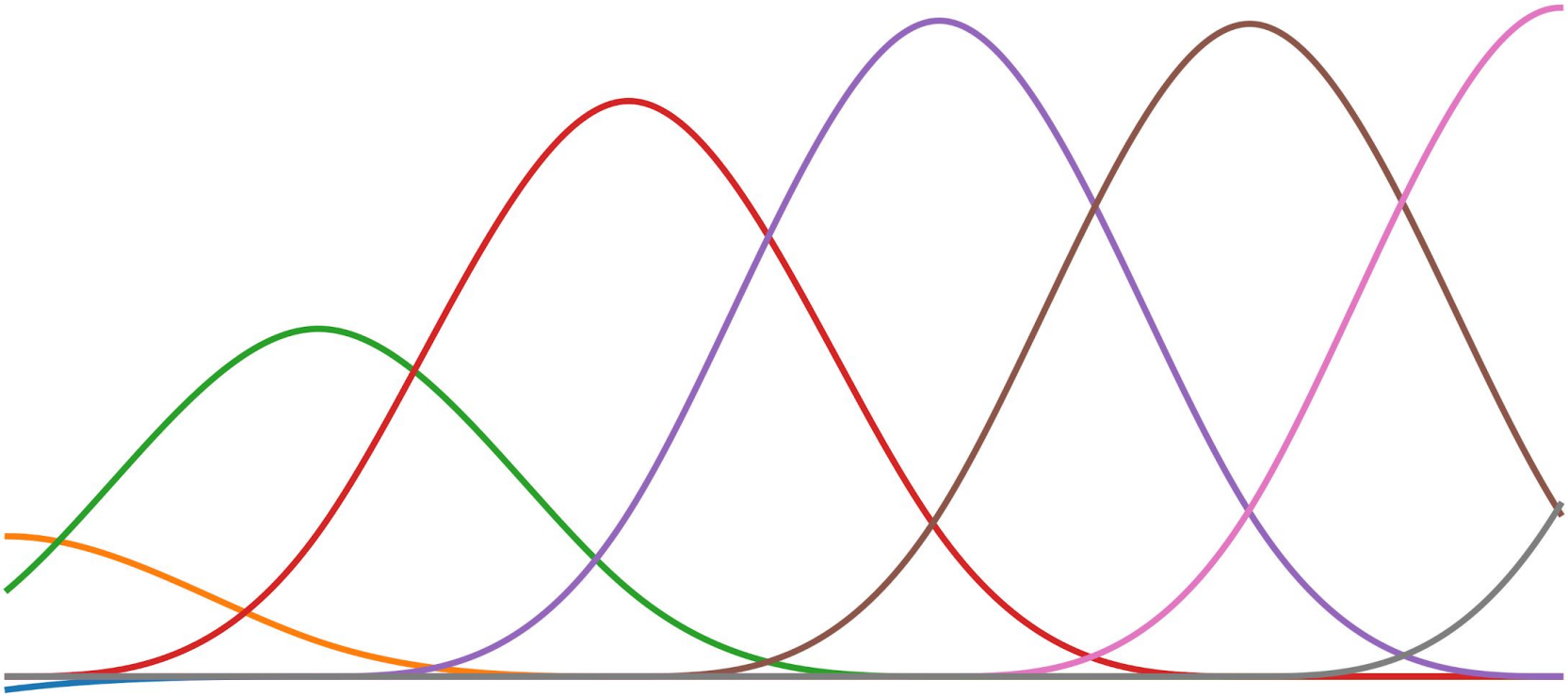
as

Spline Degree -> Infinity

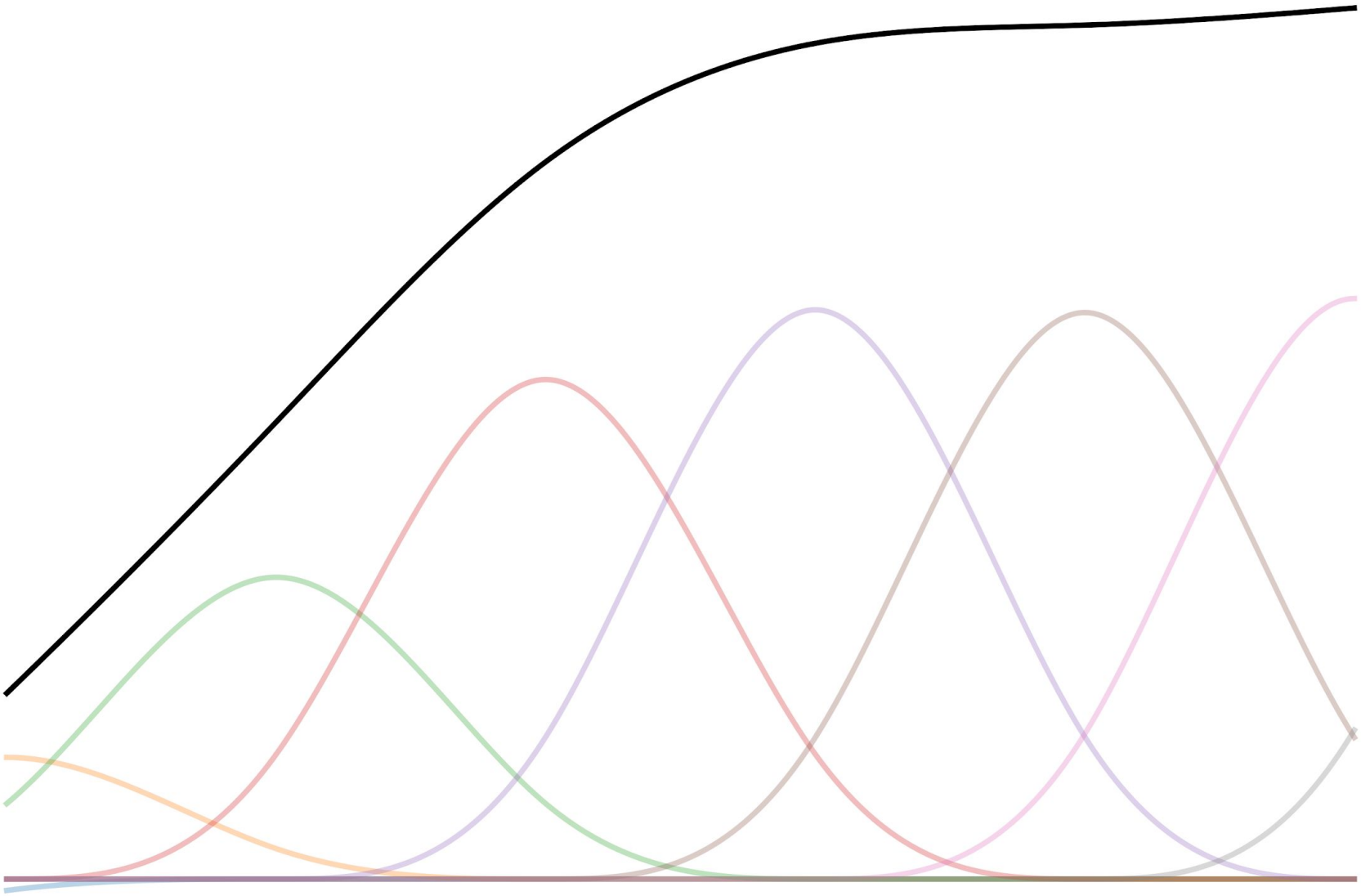
Splines



Splines



Splines



Some math...

A simple linear model:

$$y_i = X_i\beta + \epsilon_i$$

$$\epsilon_i = \mathcal{N}(0, \sigma^2)$$

Some math...

A simple GAM:

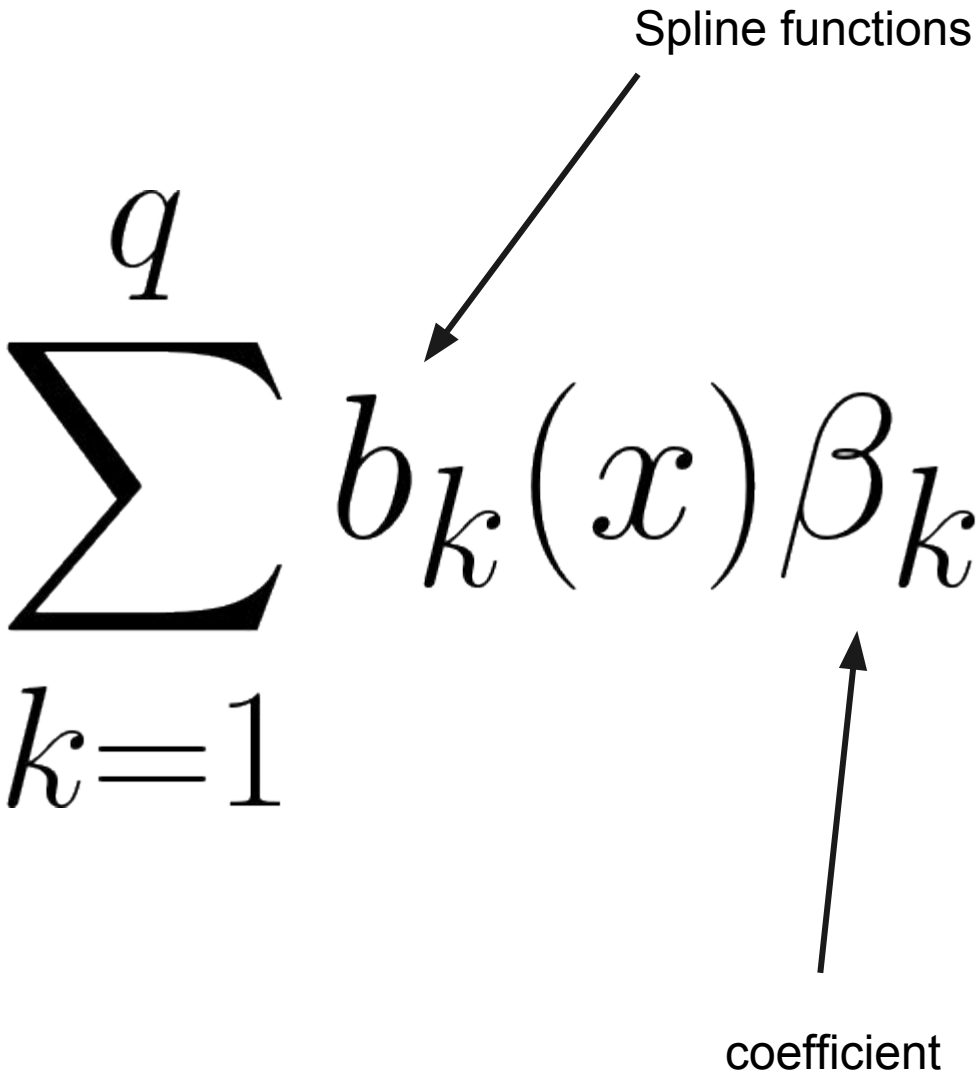
$$y_i = f(x_i) + \epsilon_i$$

$$\epsilon_i = \mathcal{N}(0, \sigma^2)$$

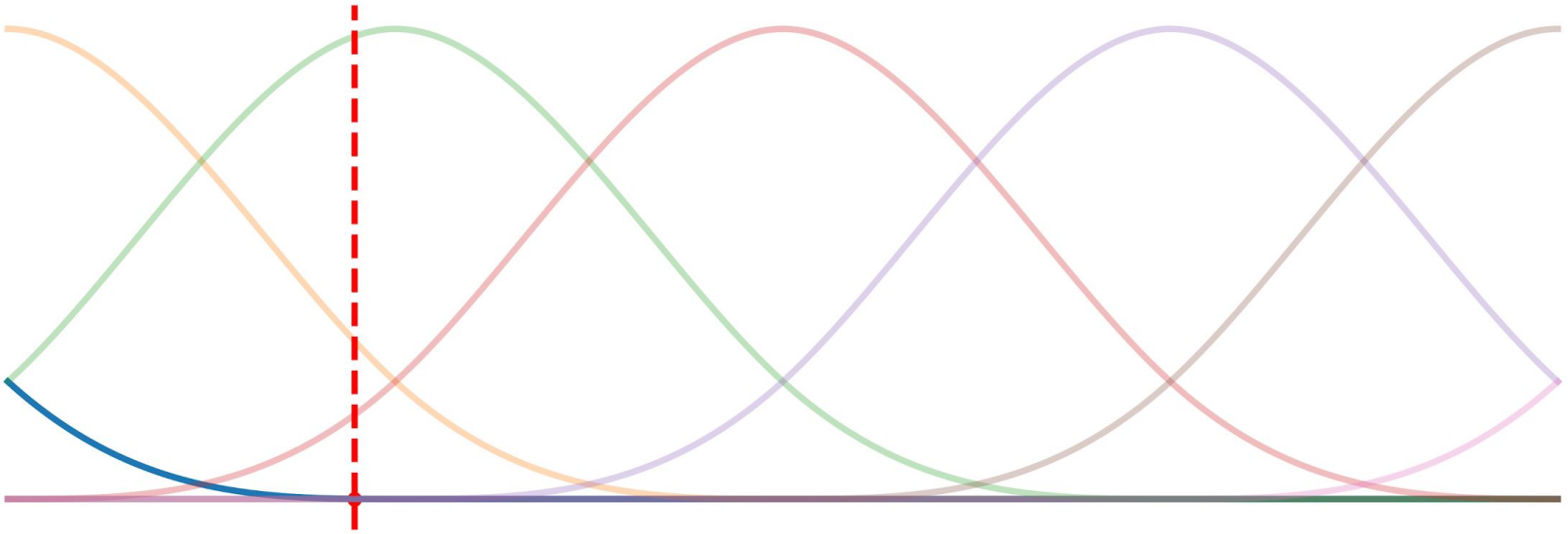
Some math...

$$f(x) = \sum_{k=1}^{q} b_k(x)\beta_k$$

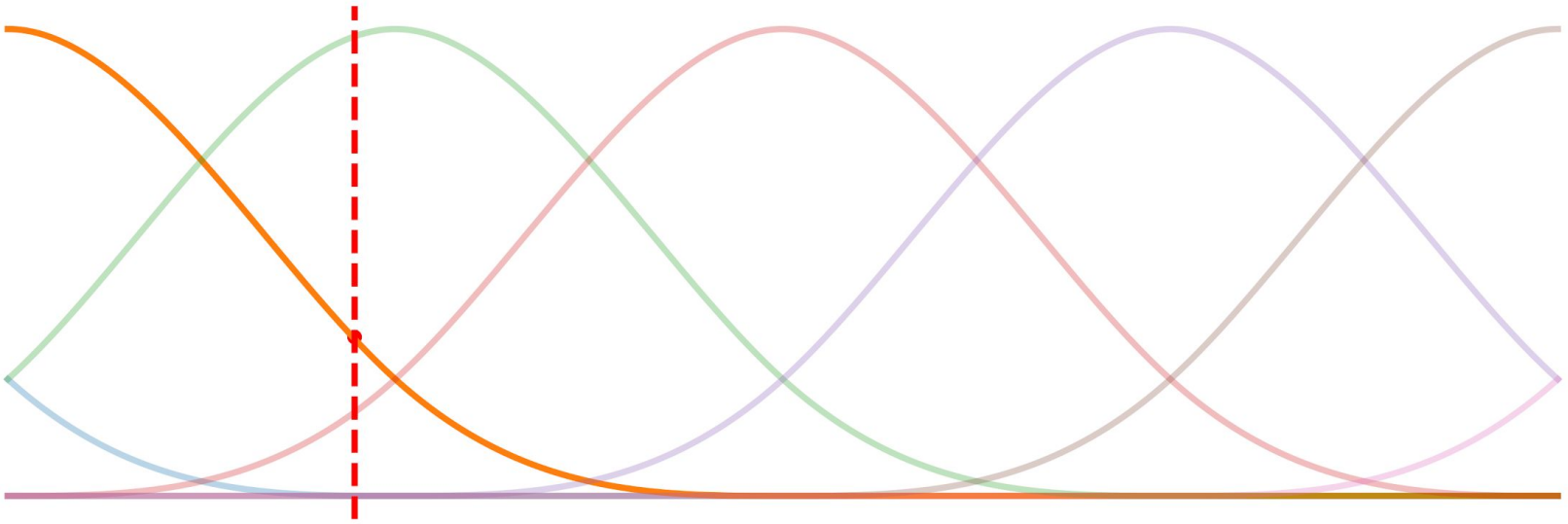
Some math...



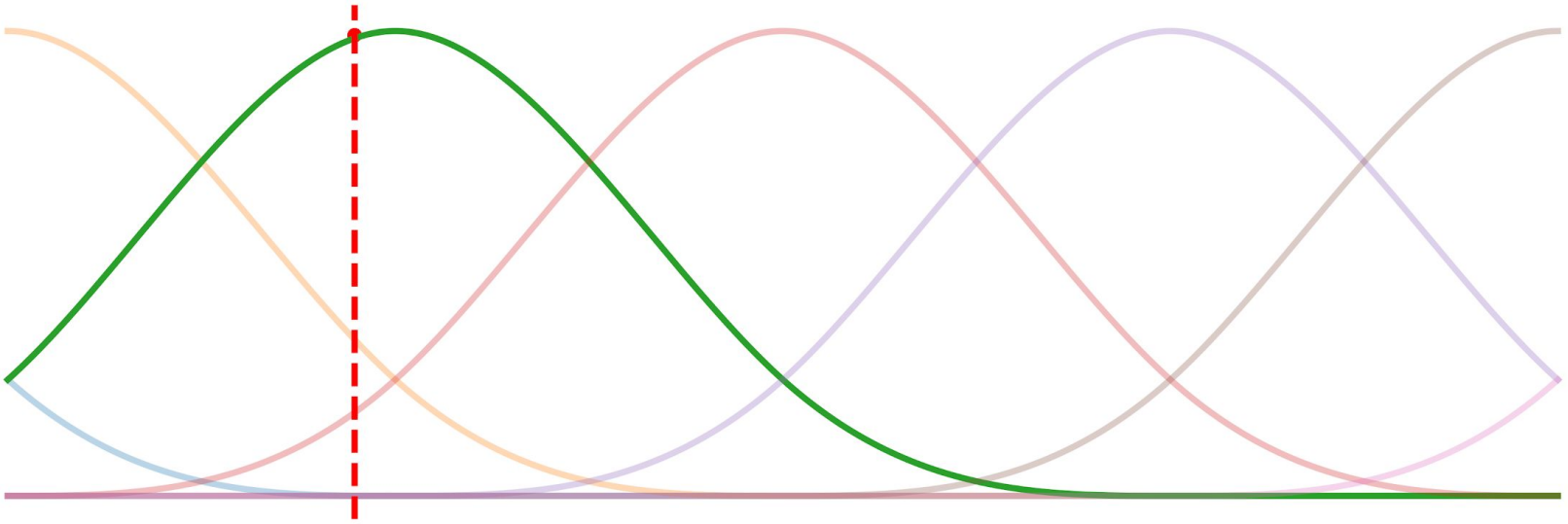
Some math...



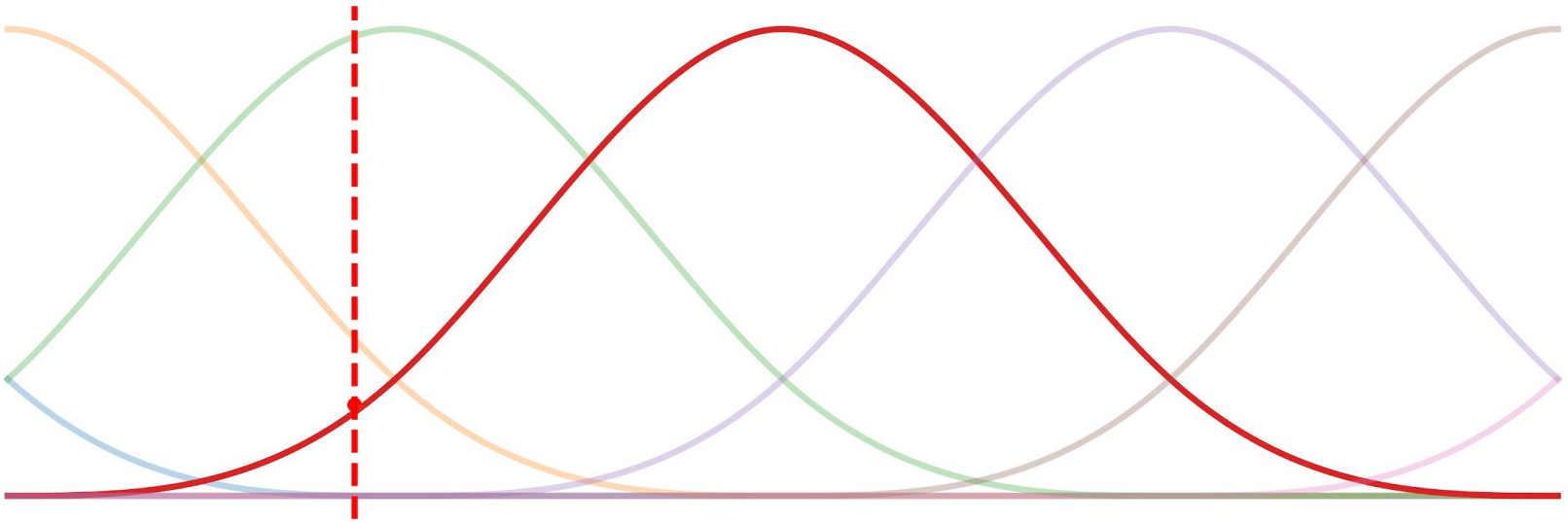
Some math...



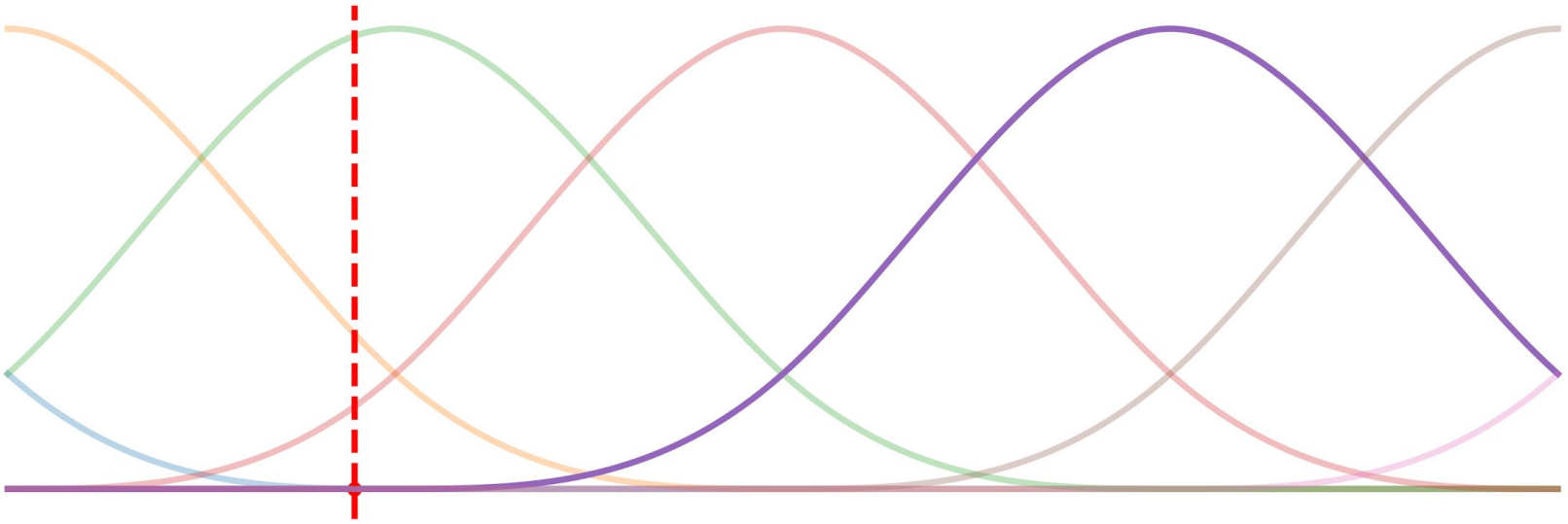
Some math...



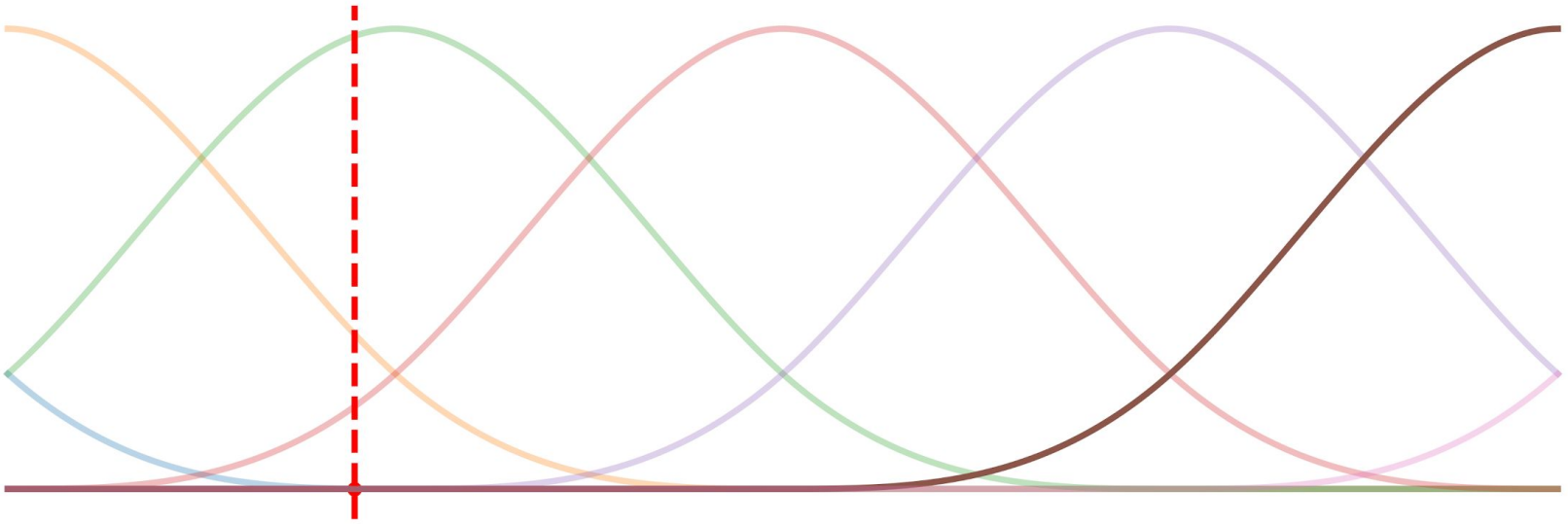
Some math...



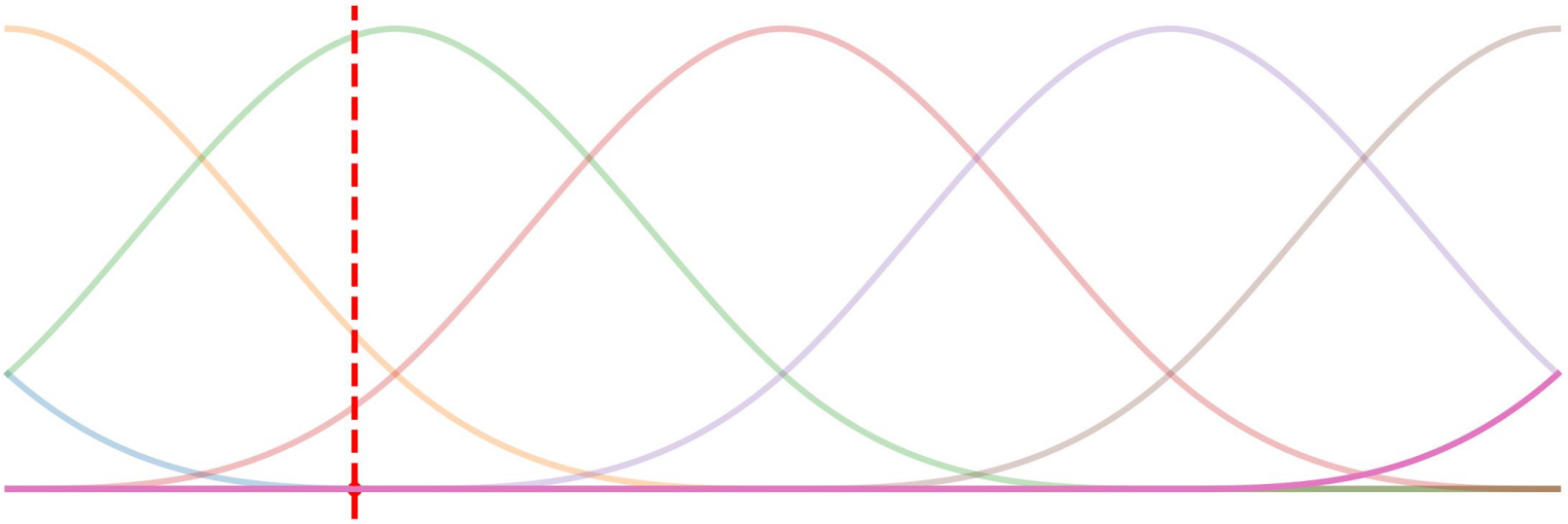
Some math...



Some math...



Some math...



Some math...

$$\sum_{k=1}^{q} b_k(x)\beta_k$$

Some math...

$$\sum_{k=1}^{q} \hat{x}_k \beta_k$$

Some math...

$$\hat{X}\beta$$

Some math...

$$y = \hat{X}\beta + \epsilon$$

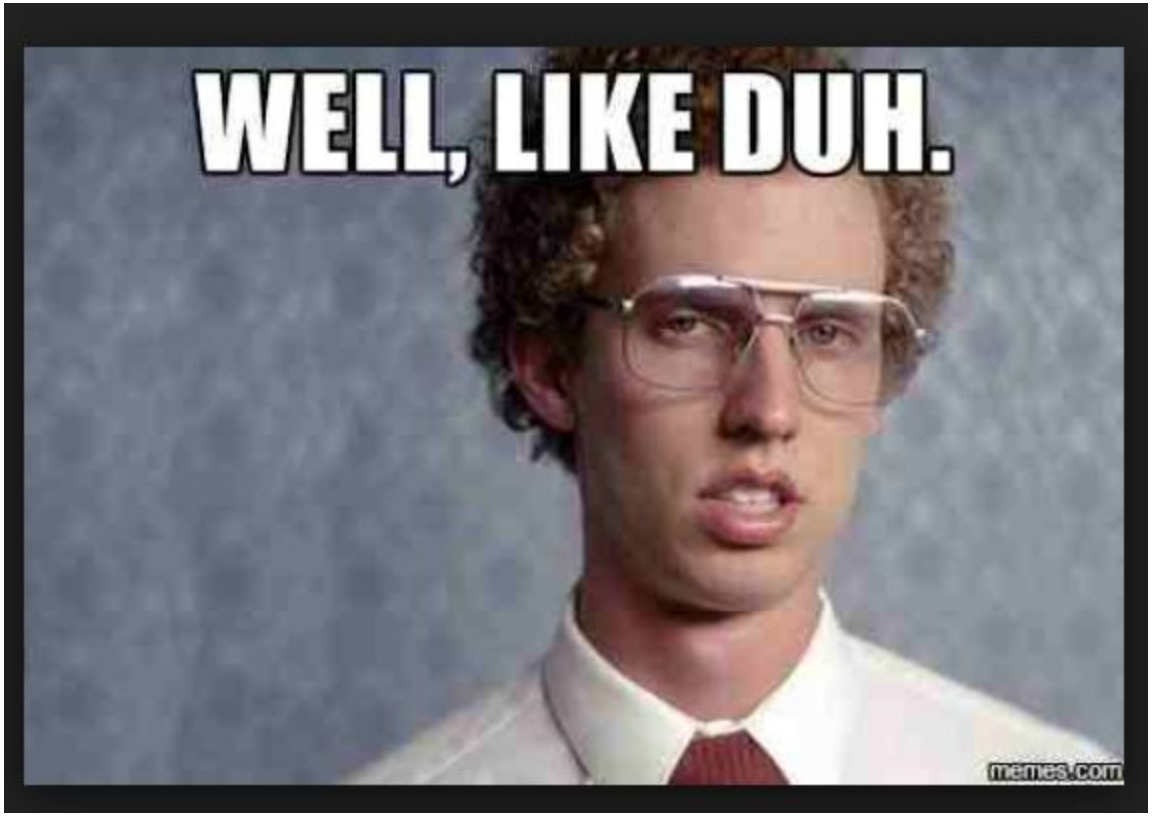
Some math...

$$y_i = \hat{X}_i \beta + \epsilon_i$$

Some math...

GAM = big GLM

Some math...

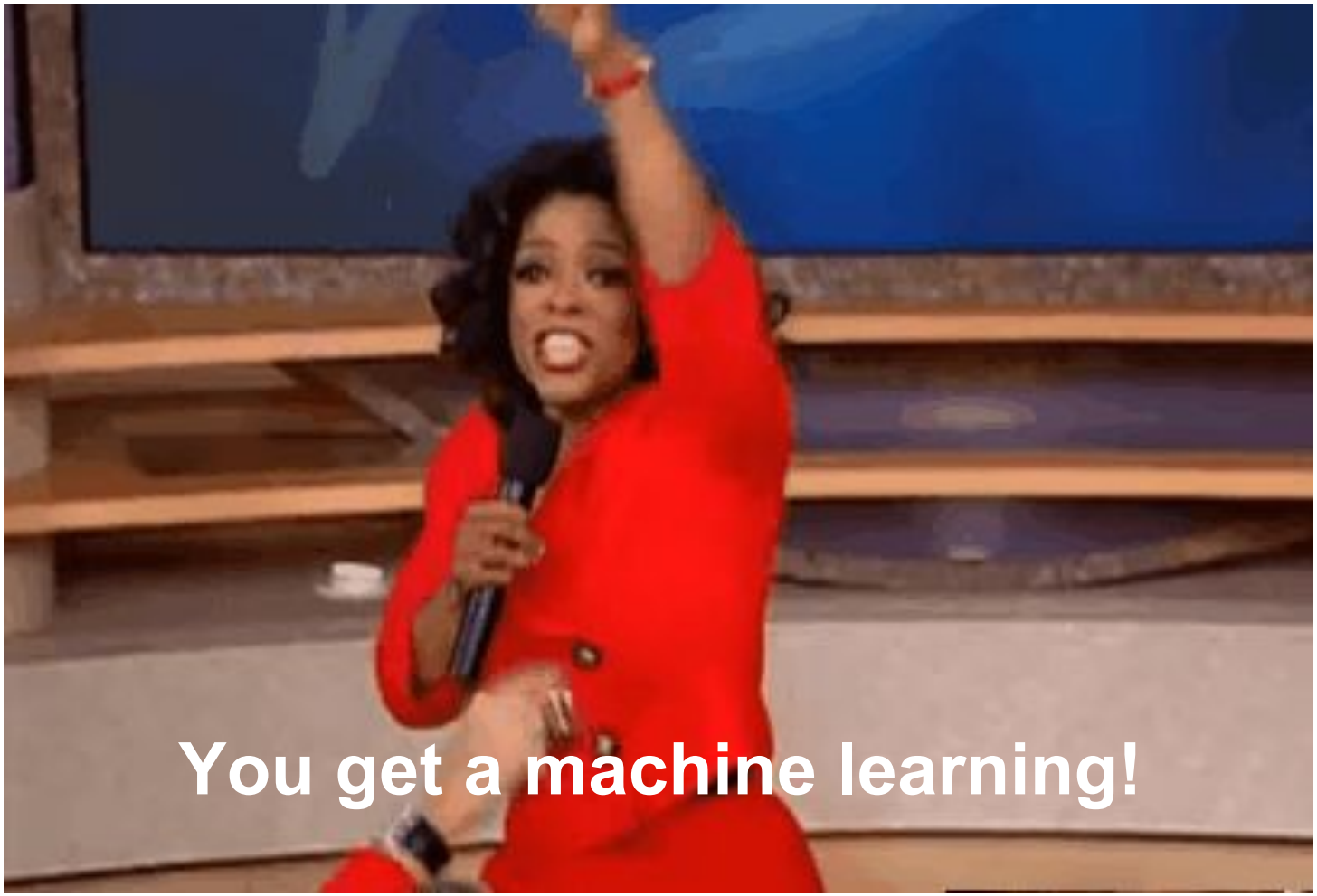


Free stuff!

Lots of **GLM** theory for free

- Confidence intervals
- Prediction intervals
- non-Normal observations
- Feature selection with p-values
- Fast cross-validation via GCV

Free stuff!

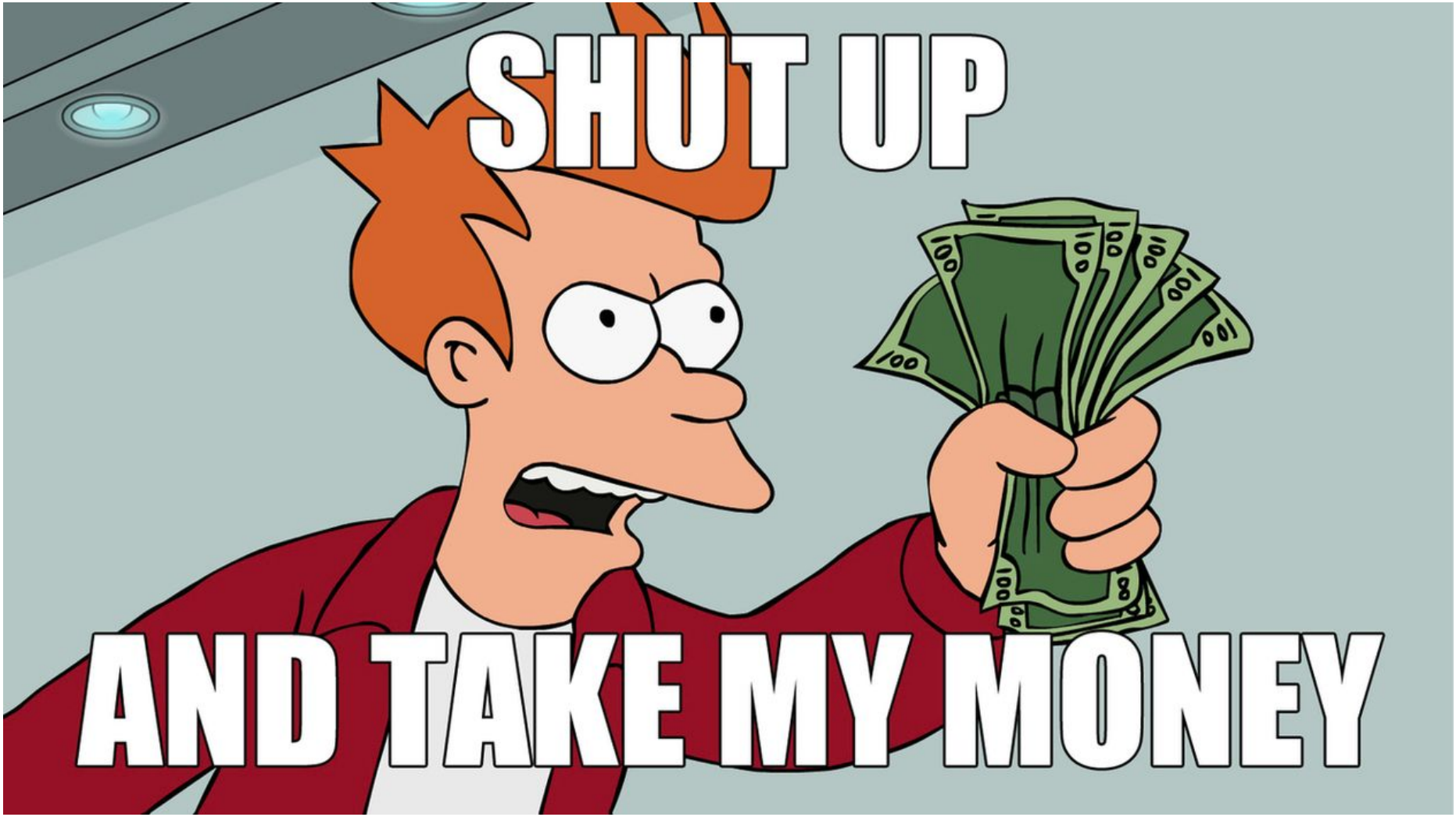


Free stuff!

And specific **GAM** stuff

- Intuitive regularization via smoothing
- Automatically model non-linearities
- Constraints (convexity, monotonicity, periodicity, ...)
- Controlled extrapolation

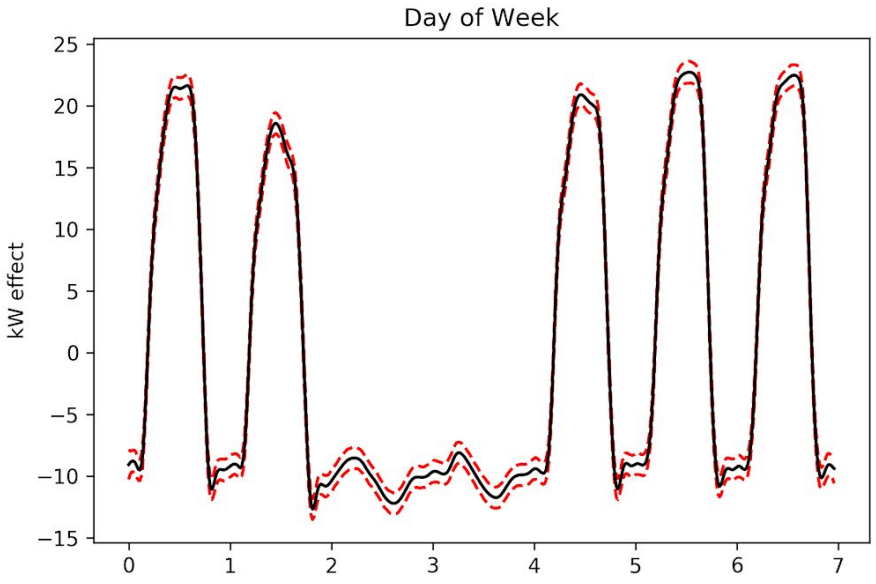
Free stuff!

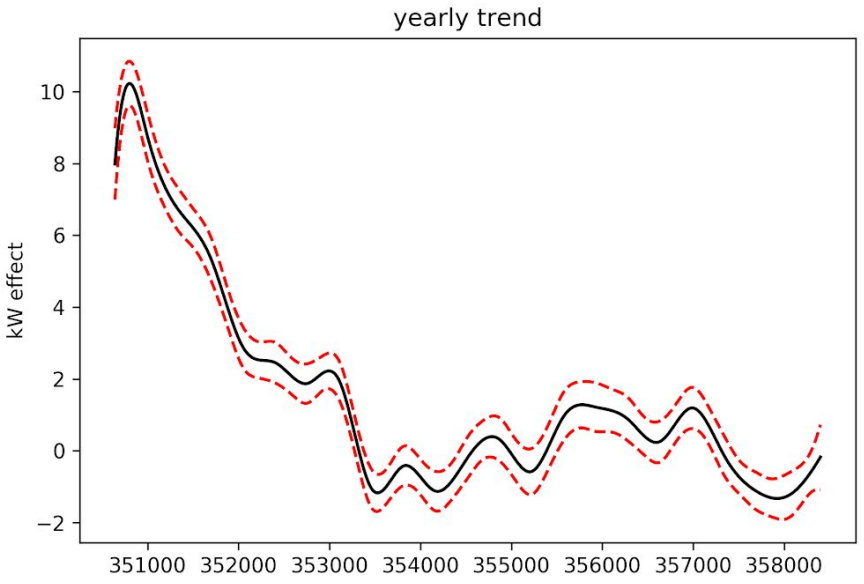


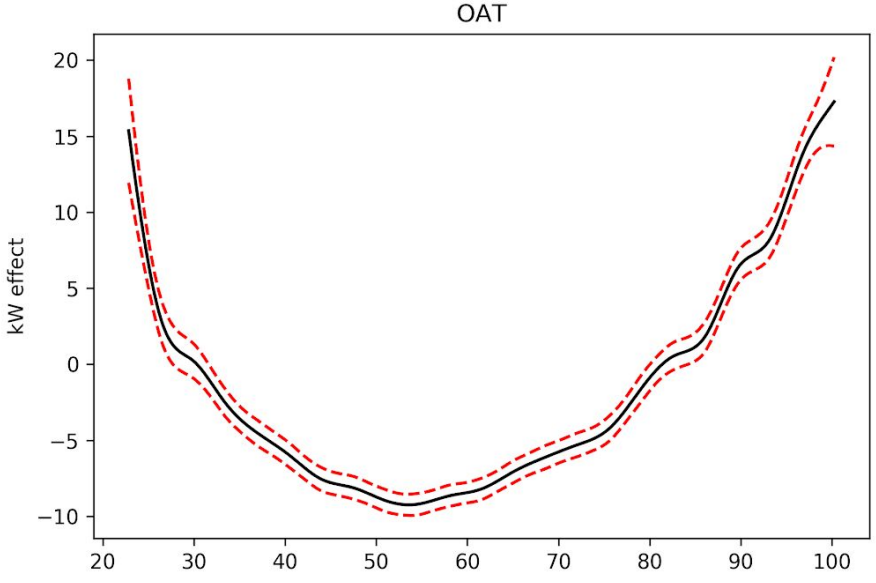
Plan

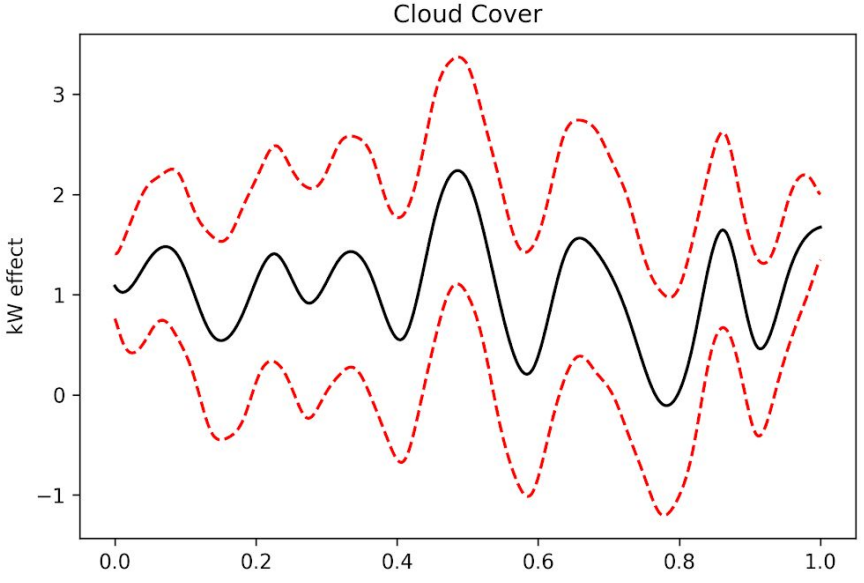
- What are GAMs
- A brief tour or splines
- Smoothing

Smoothing


Day of Week
kW effect
25
20
15
10
5
0
−5
−10
−15
0
1
2
3
4
5
6
7


yearly trend
kW effect
10
8
6
4
2
0
−2
351000
352000
353000
354000
355000
356000
357000
358000


OAT
kW effect
20
15
10
5
0
−5
−10
20
30
40
50
60
70
80
90
100


Cloud Cover
kW effect
3
2
1
0
−1
0.0
0.2
0.4
0.6
0.8
1.0

Smoothing

Penalize excessive wiggliness

$$wiggliness = \int_a^b [f''(x)]^2 dx$$

Smoothing

strength of smoothing

minimize

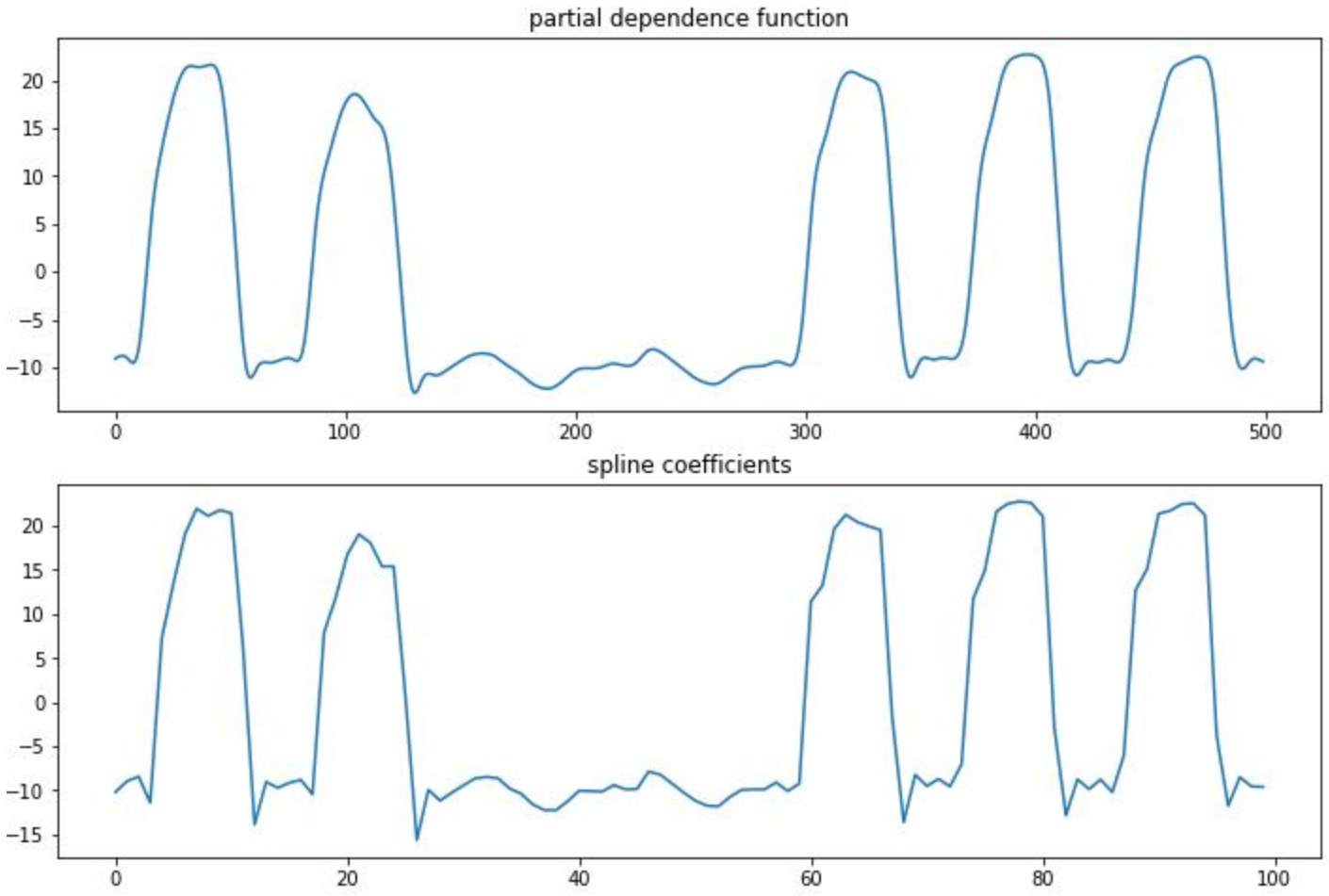
$$||y - X\beta||^2 + \lambda \int_a^b [f''(x)]^2 dx$$

Smoothing

2 modifications:

- Compute on spline coefficients β
- Use numerical derivatives + integrals

Smoothing


partial dependence function
20
15
10
5
0
−5
−10
0
100
200
300
400
500
spline coefficients
20
15
10
5
0
−5
−10
−15
0
20
40
60
80
100

Smoothing

numerical derivatives

minimize

$$||y - X\beta||^2 + \lambda\beta^\top S\beta$$

model coefficients

Smoothing

minimize

$$\beta = (X^\top X + \lambda S)^{-1} X^\top y$$

Smoothing

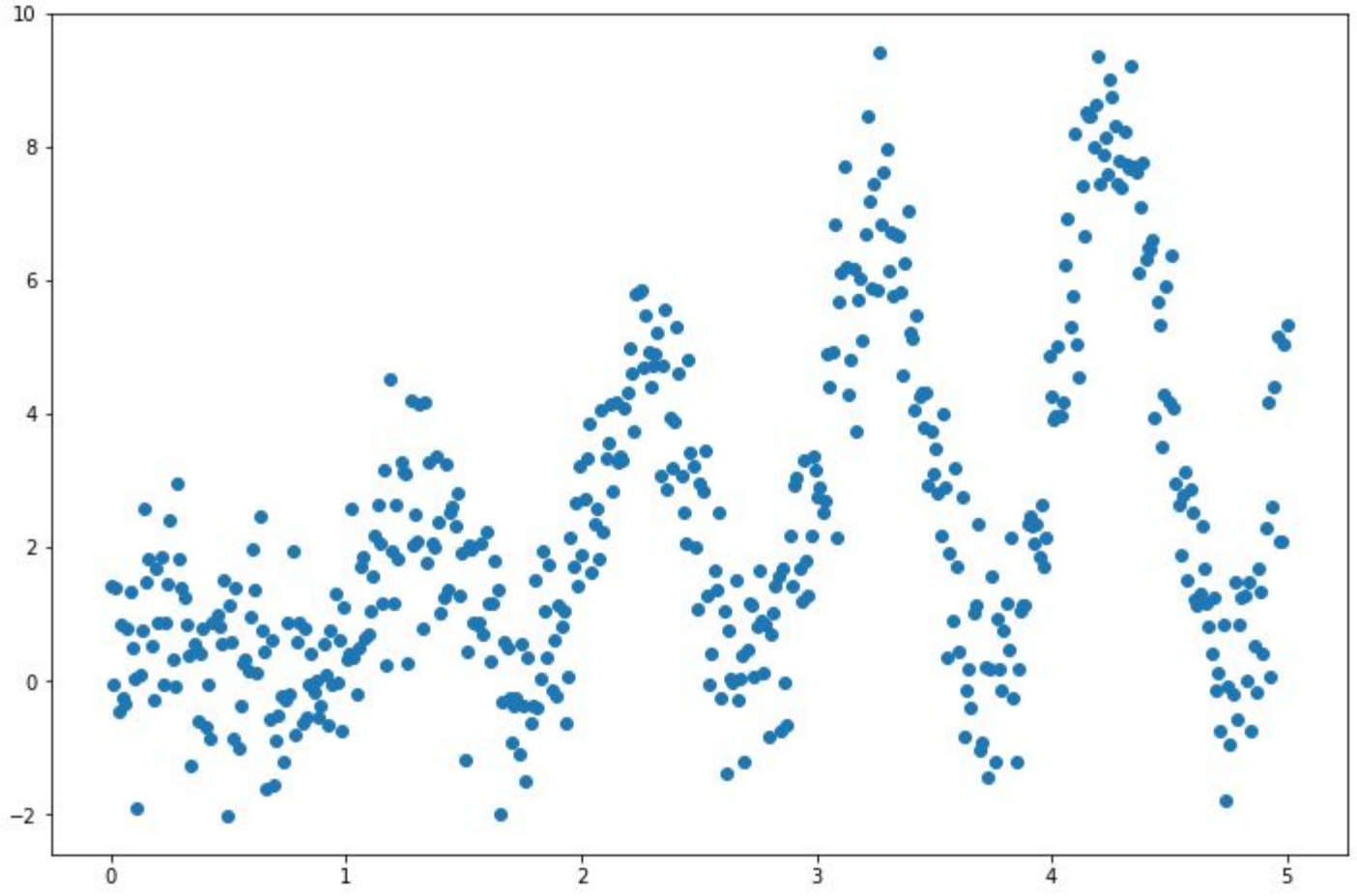
Similar to ridge regression

$$\beta = (X^\top X + \lambda I)^{-1} X^\top y$$

Smoothing

```
X = np.linspace(0, 5, 500)
y = np.sin(X * 2 * np.pi)*X + X + np.random.randn(len(X))

plt.scatter(X, y)
```

Smoothing



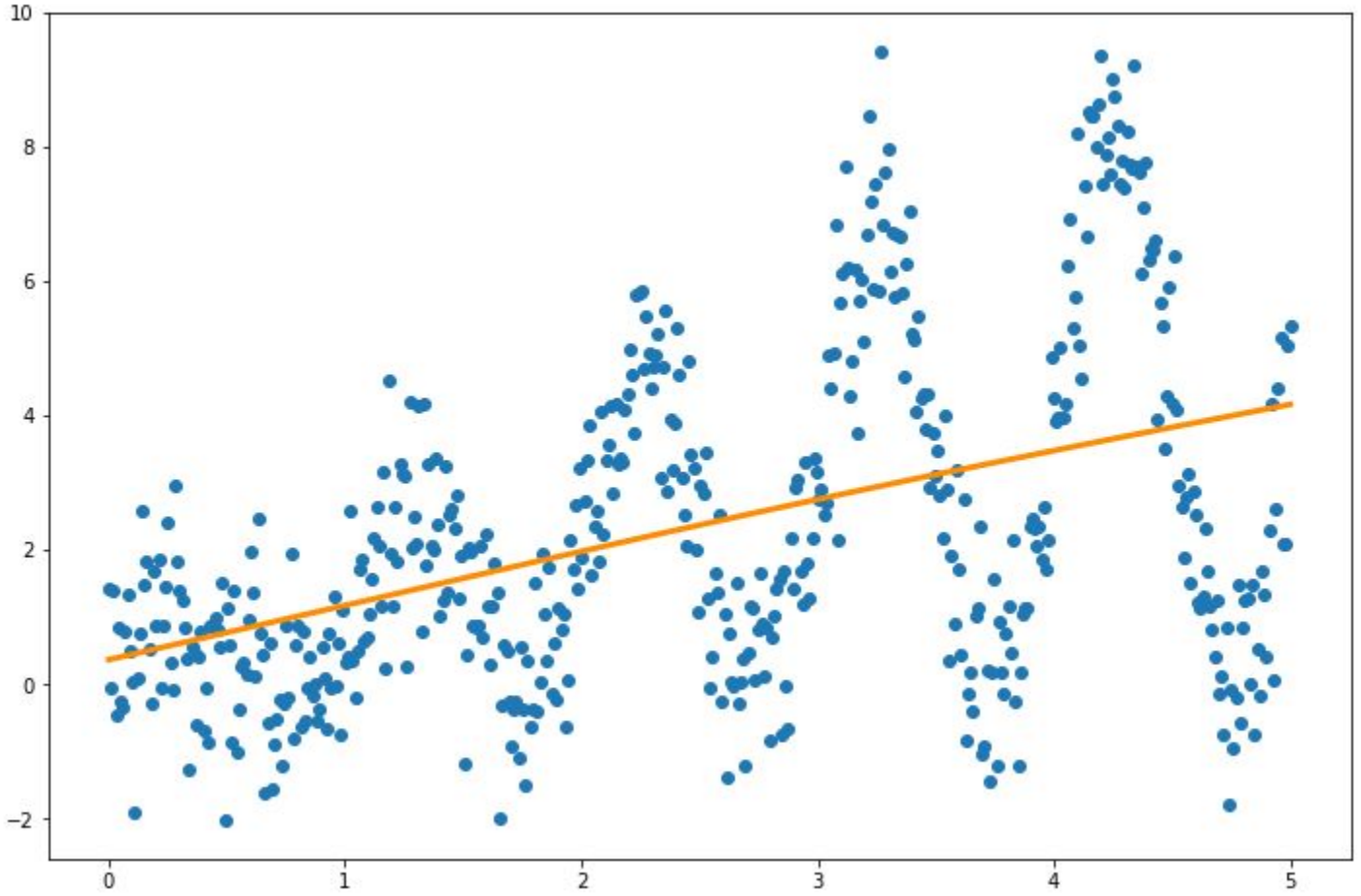
Smoothing

```
from pygam import LinearGAM

# high lambda forces a straight line
gam = LinearGAM(lam=1e6, n_splines=50).fit(X, y)
```

Smoothing

```
from pygam import LinearGAM

# high lambda forces a straight line
gam = LinearGAM(lam=1e6, n_splines=50).fit(X, y)
```

Smoothing




```
# generate a nice grid
XX = gam.generate_X_grid()
plt.plot(XX, gam.predict(XX))
```

Smoothing

```
>> gam.summary()
```

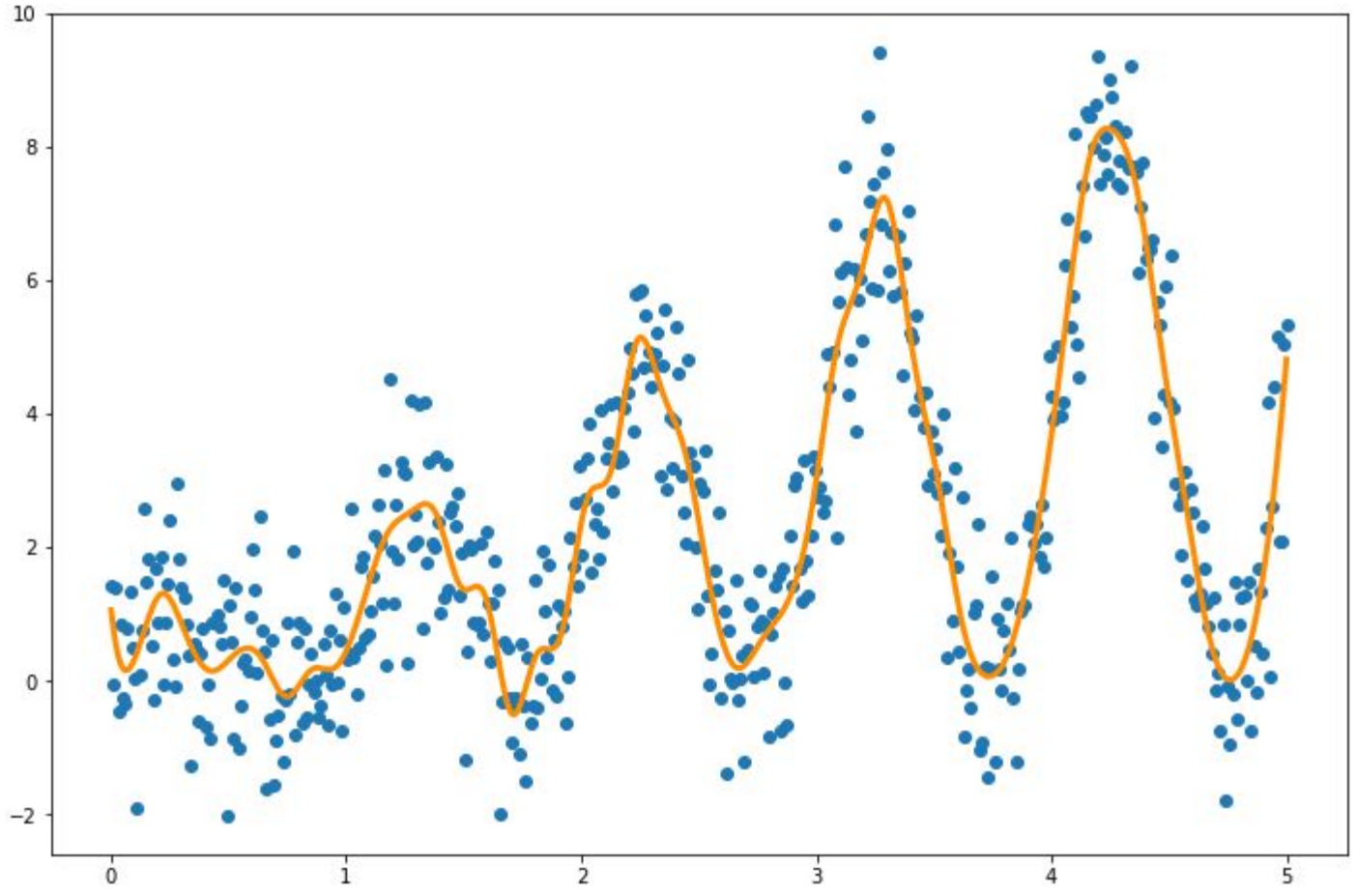
Smoothing

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     1.1141
Link Function:                    IdentityLink Log Likelihood:                                -1339.885
Number of Samples:                         500 AIC:                                           2685.9981
                                               AICc:                                          2686.0498
                                               GCV:                                              5.3358
                                               Scale:                                            5.2952
                                               Pseudo R-Squared:                                 0.2076
=================================================================================================================
Feature Function    Data Type      Num Splines   Spline Order  Linear Fit  Lambda     P > x      Sig. Code
================== ============= ============ ============ =========== ========== ========== ==========
feature 1          numerical      50            3             False       1000000.0  1.11e-16   ***
intercept                                                                           1.11e-16   ***
=================================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

Smoothing

```
# tiny lambda is super flexible
gam = LinearGAM(lam=1e-6, n_splines=50).fit(X, y)
```

Smoothing



Smoothing

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     48.9993
Link Function:                    IdentityLink Log Likelihood:                                  -674.2603
Number of Samples:                        500 AIC:                                             1450.5194
                                              AICc:                                            1462.3583
                                              GCV:                                                1.1718
                                              Scale:                                               0.963
                                              Pseudo R-Squared:                                   0.8616
==========================================================================================================
Feature Function   Data Type      Num Splines   Spline Order  Linear Fit  Lambda     P > x      Sig. Code
================== ============== ============ ============ =========== ========== ========== ==========
feature 1          numerical      50            3             False       0.0        1.11e-16   ***
intercept                                                                          1.11e-16   ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

Smoothing

So how do we choose the **right** amount of smoothing?

- Cross validation!

Smoothing

pyGAM computes GCV

- Fast approximation to Leave-One-Out CV

$$\frac{n||\boldsymbol{y}-\hat{\boldsymbol{\mu}}||}{(n-tr(A))^2}$$

Smoothing

pyGAM computes GCV

- Fast approximation to Leave-One-Out CV

$$\frac{n||\boldsymbol{y}-\hat{\boldsymbol{\mu}}||}{(n-edof)^2}$$

Smoothing

```
gam = LinearGAM(n_splines=50).fit(X, y)
```

Smoothing

```
gam = LinearGAM(n_splines=50).gridsearch(X, y)
```

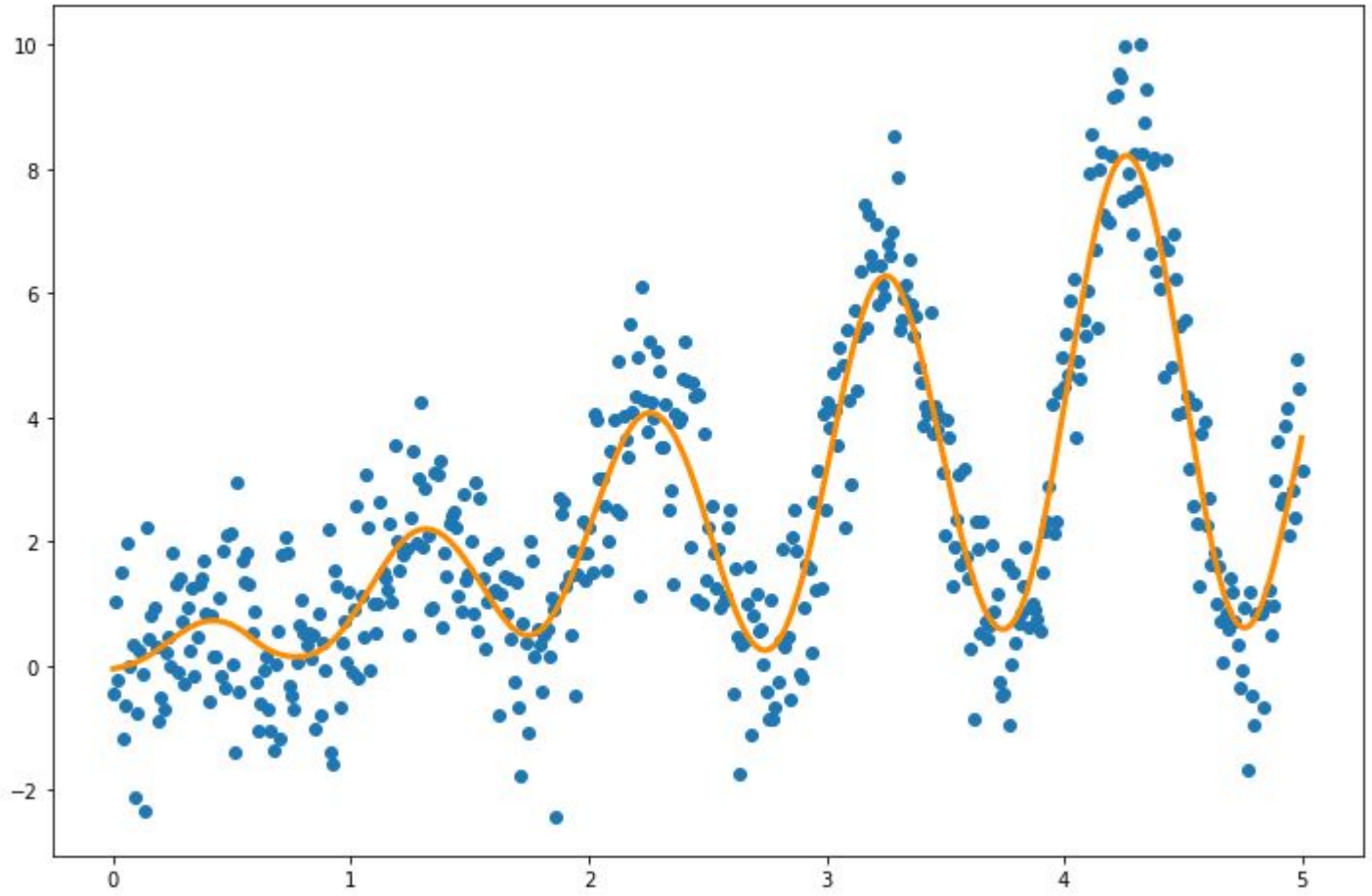
Smoothing

```
lams = np.logspace(-4, 0, 100)
gam = LinearGAM(n_splines=50).gridsearch(X, y, lam=lams)
```

Smoothing

```
LinearGAM
============================================= ==========================================================
Distribution:                      NormalDist Effective DoF:                                     25.1906
Link Function:                    IdentityLink Log Likelihood:                                  -692.0659
Number of Samples:                        500 AIC:                                             1438.5129
                                              AICc:                                            1441.7621
                                              GCV:                                                1.0855
                                              Scale:                                              0.9836
                                              Pseudo R-Squared:                                   0.8477
==========================================================================================================
Feature Function   Data Type      Num Splines   Spline Order  Linear Fit  Lambda     P > x      Sig. Code
================== ============== ============ ============ =========== ========== ========== ==========
feature 1          numerical      50            3             False       1.0        1.11e-16   ***
intercept                                                                          1.11e-16   ***
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

Smoothing



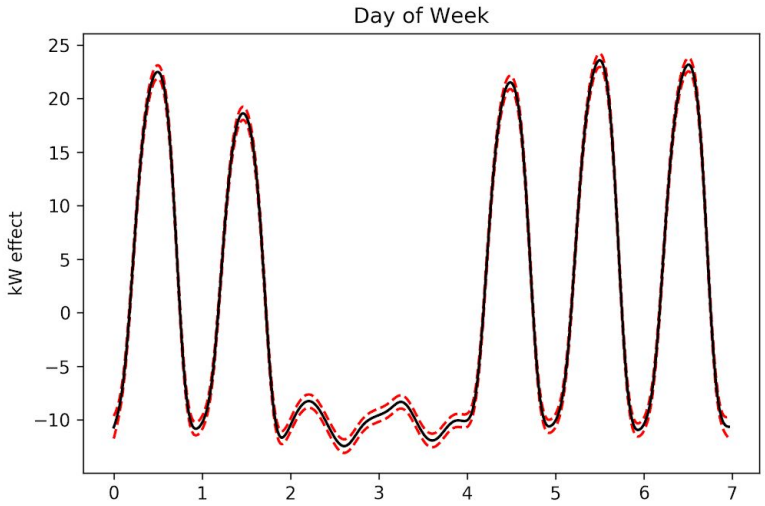
Back at PyDatatopolis

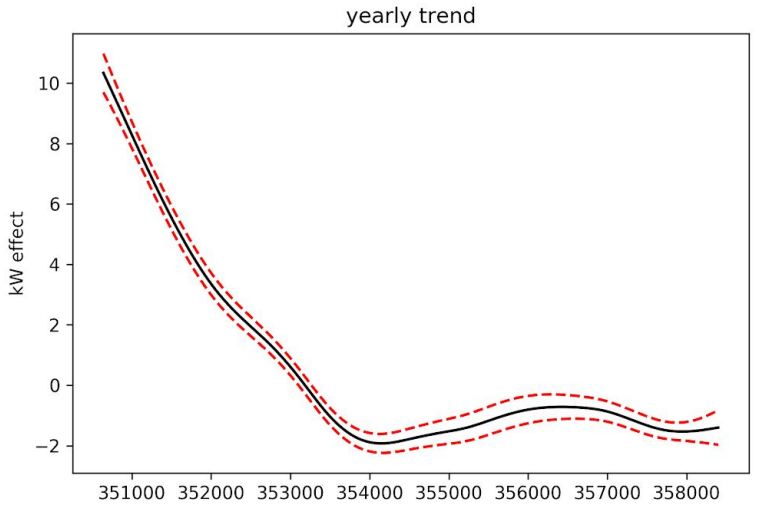
- Select right amount of smoothing
- Cloud cover?

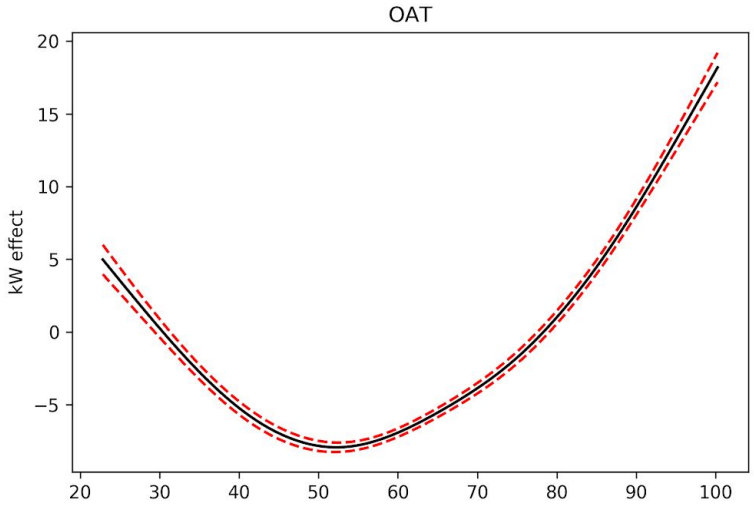
Back at PyDatatopolis

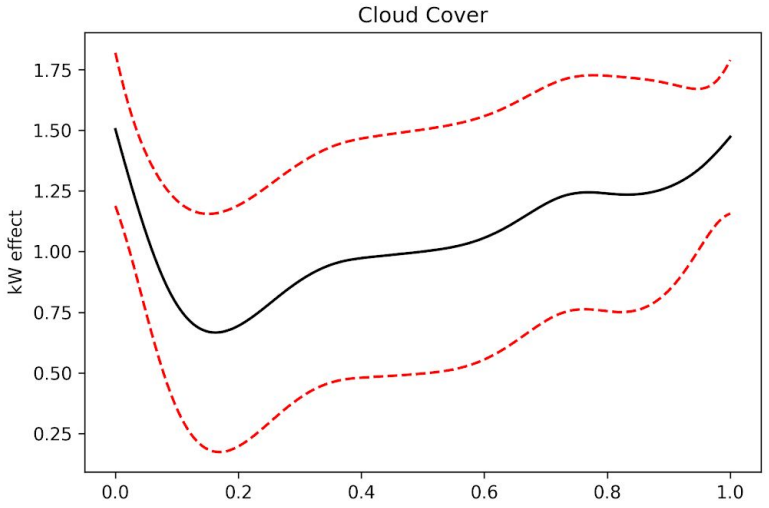
```
dist = np.random.uniform(1e-3, 1e3, (100,4))

gam = LinearGAM(n_splines=[100,25,25,25]).gridsearch(X, y, lam=dist)
```

Back at PyDatatopolis









Back at PyDatatopolis

```
LinearGAM
============================================= ==========================================================
Distribution:                        NormalDist Effective DoF:                                     56.6547
Link Function:                     IdentityLink Log Likelihood:                                -32515.6534
Number of Samples:                         7759 AIC:                                            65146.6162
                                                AICc:                                           65147.4945
                                                GCV:                                               26.2008
                                                Scale:                                             25.8566
                                                Pseudo R-Squared:                                   0.8865
=================================================================================================================
Feature Function   Data Type      Num Splines   Spline Order  Linear Fit  Lambda     P > x      Sig. Code
================== ============== ============= ============= =========== ========== ========== ==========
feature 1          numerical      100           3             False       36.4721    1.11e-16   ***
feature 2          numerical      25            3             False       229.9264   1.11e-16   ***
feature 3          numerical      25            3             False       938.4219   1.11e-16   ***
feature 4          numerical      25            3             False       430.7849   4.70e-01
intercept                                                                           1.11e-16   ***
=================================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Future work

Lots of cool features…

- Feature interactions
- Periodicity constraints
- Direct smoothing parameter optimization
- ExpectileGAM
- …

https://github.com/dswah/pyGAM/issues

Vielen Dank!

Questions?

- Can pyGAM do classification?
- What other distributions can we model?
- What size of data can pyGAM handle?
- Patsy compatible?

Some math...