Week 8: Flexible Regression

PLSC 40502 - Statistical Models

Review

Previously

- **Item response theory**
  - **Factor model** for categorical/nominal outcome variables
  - Model a matrix of **individual** responses across multiple common **questions**
  - Responses are a function of a common **indivdual** latent parameter
  - Identification via Bayes (prior on the individual latent parameters defines the scale/location).

This week

- **Flexible functional forms**
  - Semi-/Non-parametric approaches to modeling CEFs of Y_i given a continuous X_i
  - Regression and smoothing splines to allow for flexible relationships between
  - Penalty term to avoid "jumpy" regressions
  - Generalized Additive Models (GAMs) that combine "parametric" and "semi-/non-parametric" components
- **Regularization**
  - Why regularize?
  - L_0, L_1, and L_2 norms
  - Value of the lasso (the L_1 norm) - "sparse" regressions
  - Interpreting regularization in Bayesian terms.

Flexible regression

Flexible regression

- A common task in statistics is estimating the conditional expectation function $E[Y|X]$.
  - But typical methods for estimating the CEF assume that we know its functional form.
  - For example, we assume linearity -- can be trivially satisfied when X is discrete, but potentially problematic when X is continuous.

$$E[Y|X] = f(X) = X\beta$$

- We want to maintain the utility of a model that is **linear in the parameters** but introduce transformations of X to capture potentially non-linear relationships between Y and X.
- Define the **linear basis expansion** for a set of M basis functions $h_m(X)$

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

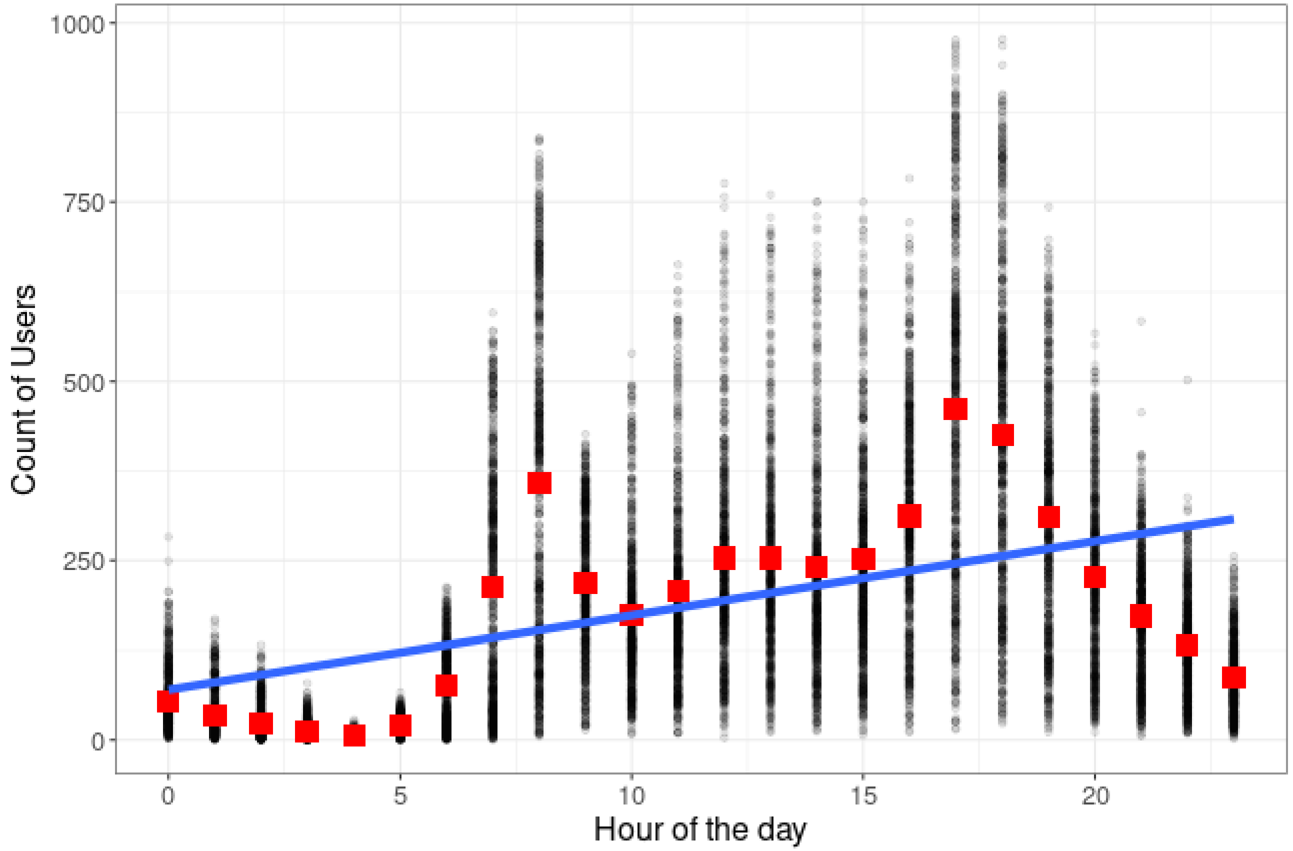
Example: Modeling Bike Rentals

- **Bikeshare usage** is highly variable from day-to-day and hour-to-hour. Capital Bikeshare in Washington D.C. recorded the hourly count of active users over a two-year period from 2011 to 2012.
  - For more on the dataset, see: Fanaee-T, Hadi, and Gama, Joao, "Event labeling combining ensemble detectors and background knowledge", *Progress in Artificial Intelligence* (2013): pp. 1-15

```
bike <- read_csv("data/bikes_hour.csv")
bike_by_hour <- bike %>% group_by(hr) %>% summarize(cnt = mean(cnt))
```

Example: Modeling Bike Rentals

- From the scatterplot of active usage vs. hour of the day, a simple linear fit (slope + intercept) seems quite poor at capturing the CEF



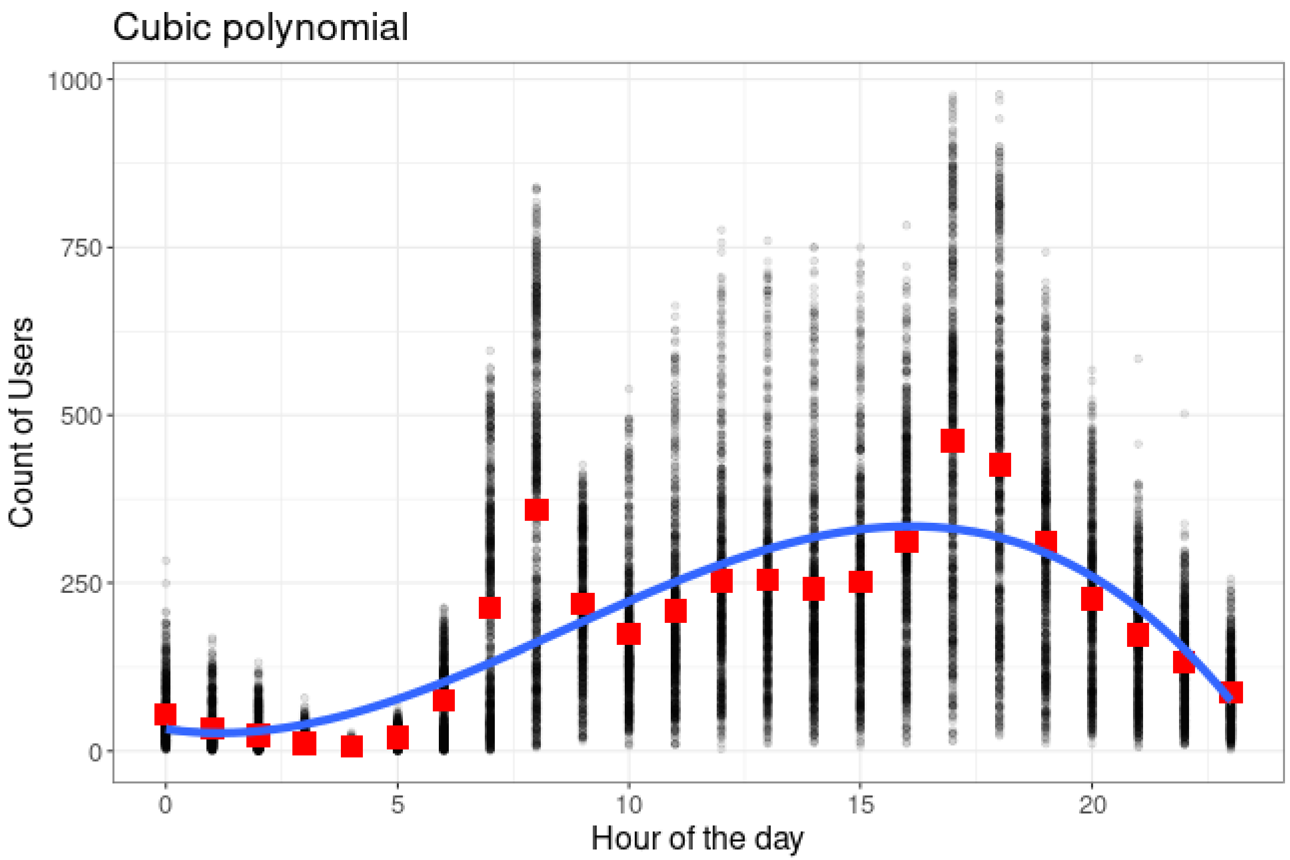
Polynomial basis

- A common set of basis functions to choose are the **global polynomial** basis
  - You've probably already done this when you've included squared terms in your regressions!
- For example, for a univariate X, the basis for a global cubic polynomial is:

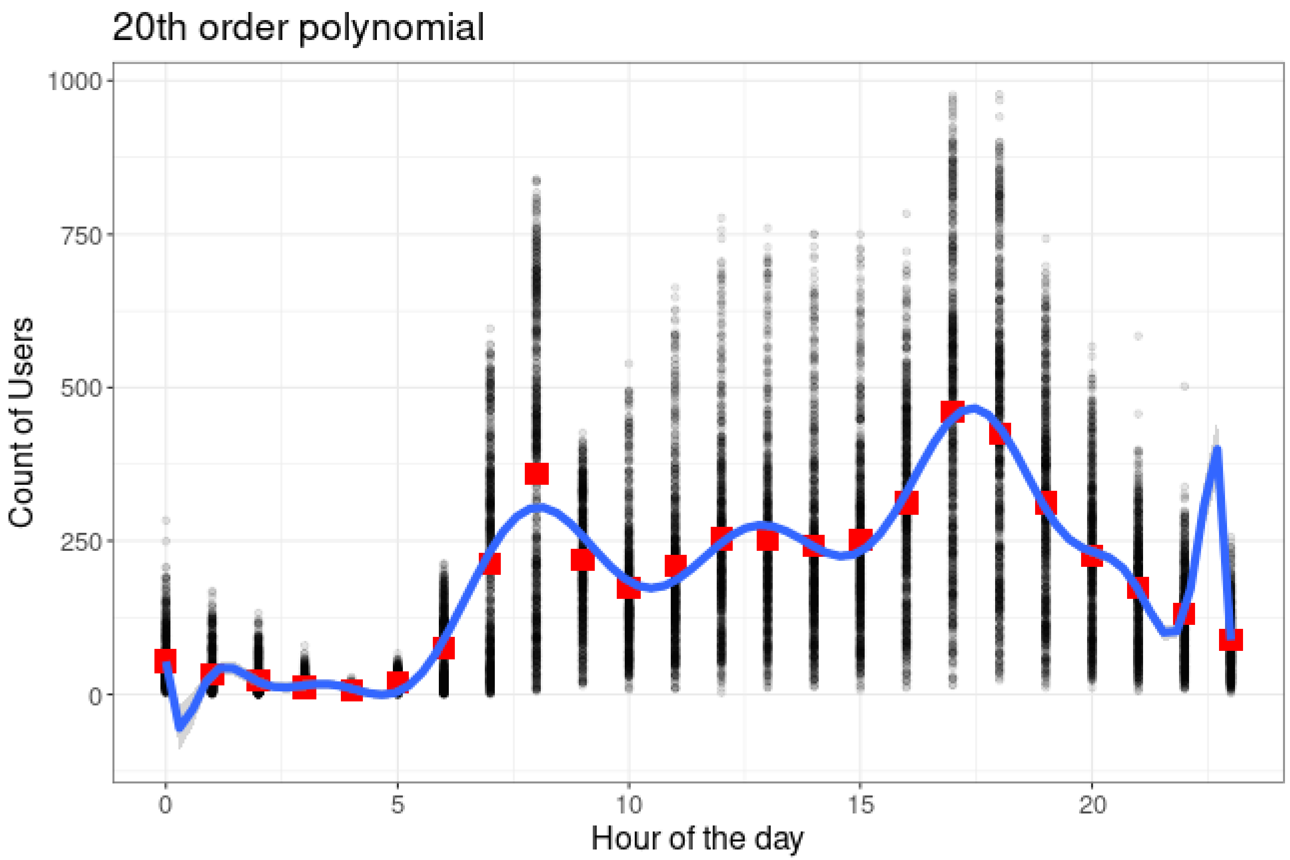
$$
\begin{aligned}
h_1(X) &= 1 \\
h_2(X) &= X \\
h_3(X) &= X^2 \\
h_4(X) &= X^3
\end{aligned}
$$

- A Kth order polynomial requires $K + 1$ parameters
- However, there are some drawbacks to using a global polynomial - namely that each observation influences the **entire** curve.

Polynomial basis



Polynomial basis

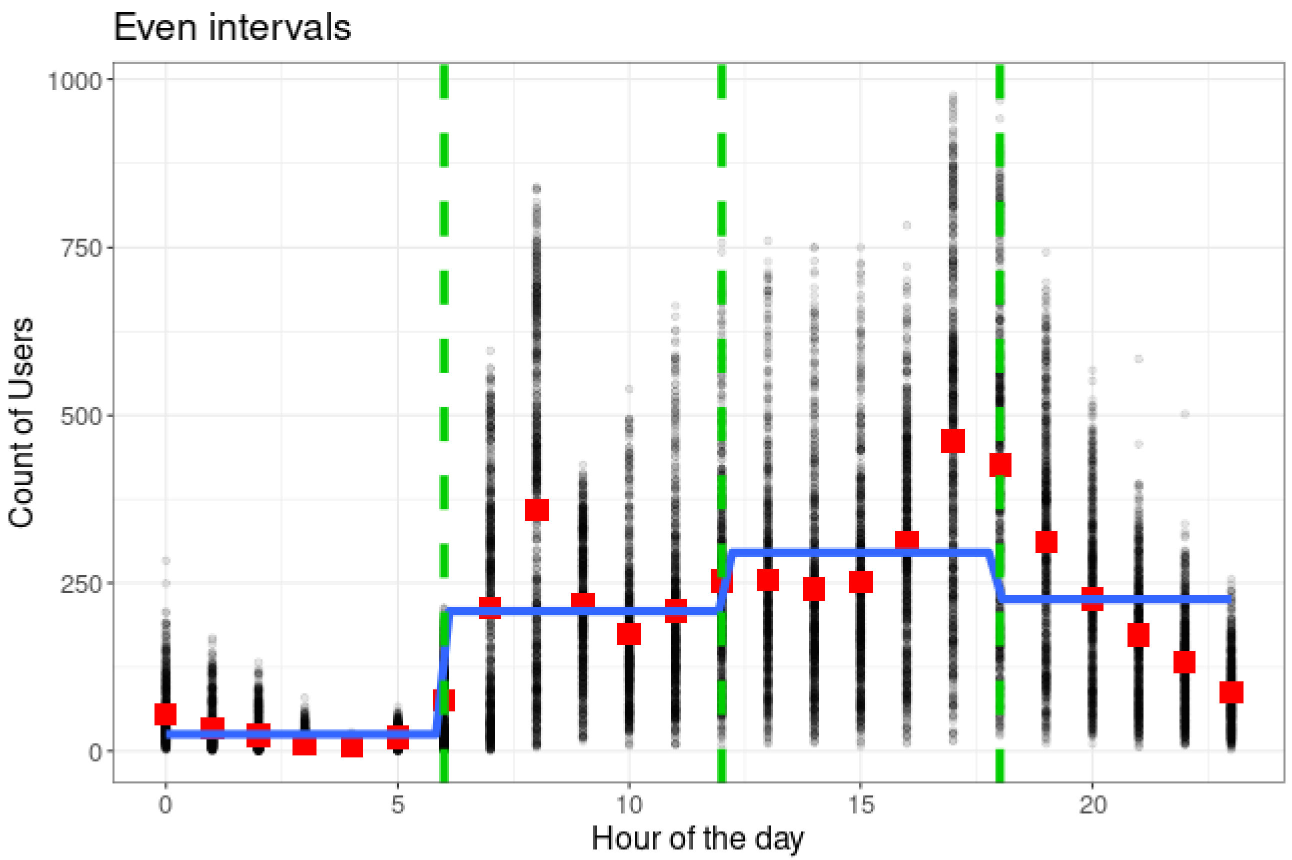


Step functions

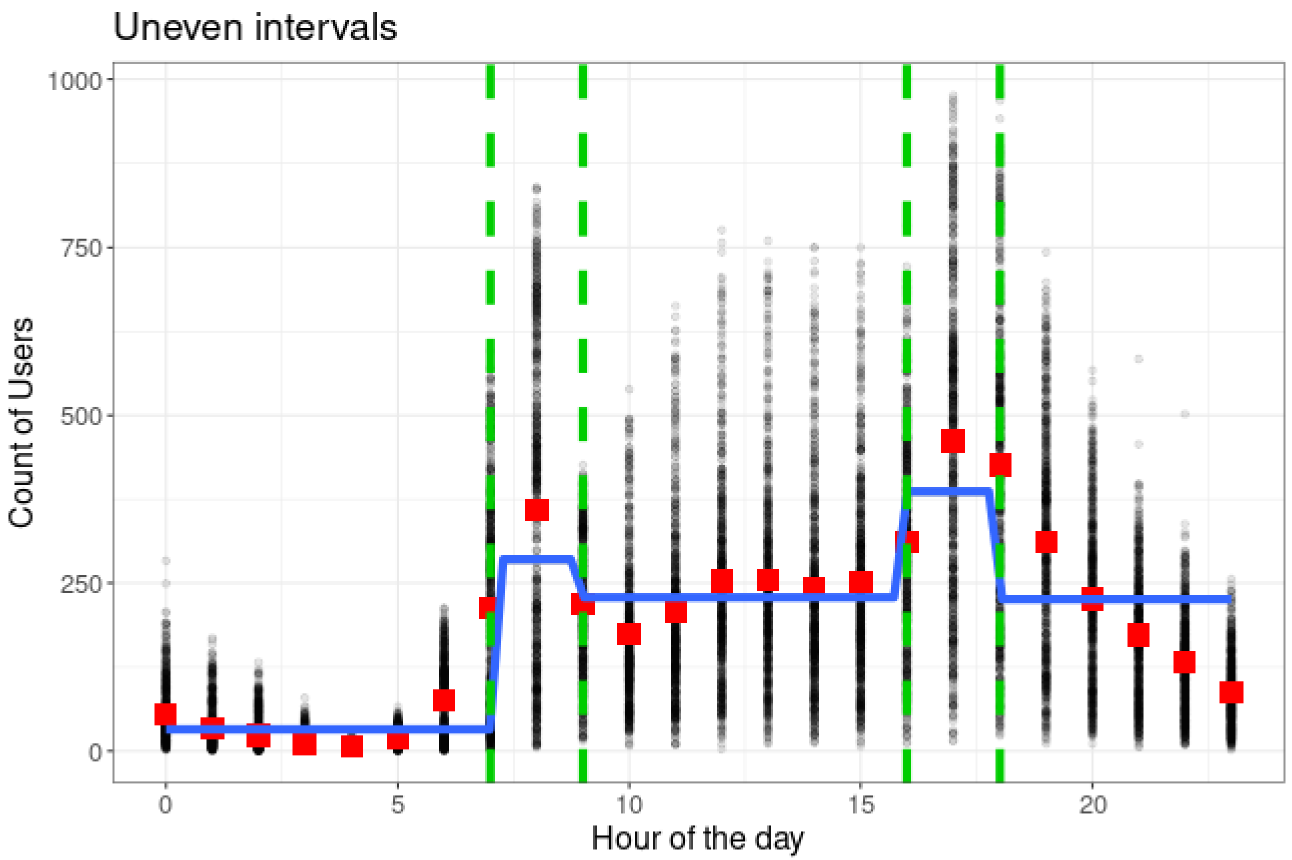
- Instead of forcing a single **global** polynomial, we might instead want to fit a set of **local** averages to different regions of X
- We could define a set of basis functions that are indicators which partition X into $M+1$ disjoint regions based on cutpoints $\xi_1, \xi_2, \ldots, \xi_M$

$$\begin{aligned}
h_1(X) &= I(X < \xi_1) \\
h_2(X) &= I(\xi_1 \leq X < \xi_2) \\
h_3(X) &= I(\xi_2 \leq X < \xi_3) \\
&\vdots \\
h_{M+1}(X) &= I(\xi_M \leq X)
\end{aligned}$$

Step functions

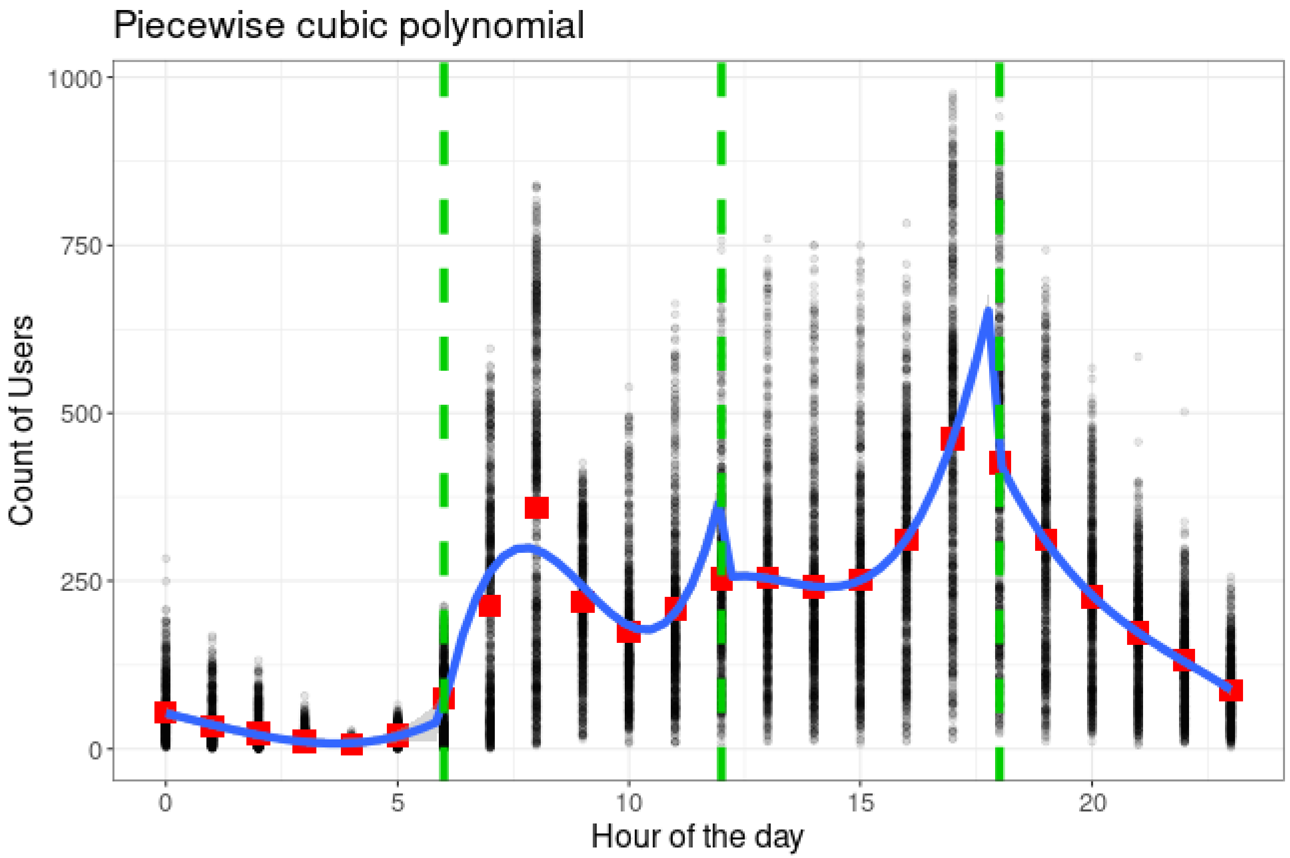


Step functions



Piecewise polynomials

- Rather than just taking the mean within each disjoint region, we could imagine fitting a polynomial to **just** that subset.



Splines

- The piecewise polynomials might fit better, but we still have these irritating discontinuities in the CEF.
  - We might want to impose some **additional** conditions on the function regarding continuity around the cutpoints.
- A $K-1$th order **spline** with M knots $\xi_1, \xi_2, \ldots, \xi_M$ is a piece-wise polynomial that...
  - ...is a polynomial of degree $K-1$ on the intervals $(-\infty, \xi_1,], [\xi_1, \xi_2], [\xi_2, \xi_3], \ldots, [\xi_m, \infty)$
  - ...has a jth derivative that is continuous at each of the knots $\xi_1, \xi_2, \ldots, \xi_m$ for $j = 0, 1, 2, \ldots, K-2$
- Intuitively, if we **also** forced the kth derivative to be continuous, we'd recover the **global** polynomial.
- Most common spline is the **cubic** spline $k = 4$ (third order polynomial).
  - Splines allow for local flexibility while still retaining continuity across X.

Splines

- There are actually multiple ways to define the basis functions for a spline. The most intuitive for understanding how they work is the **truncated power basis**

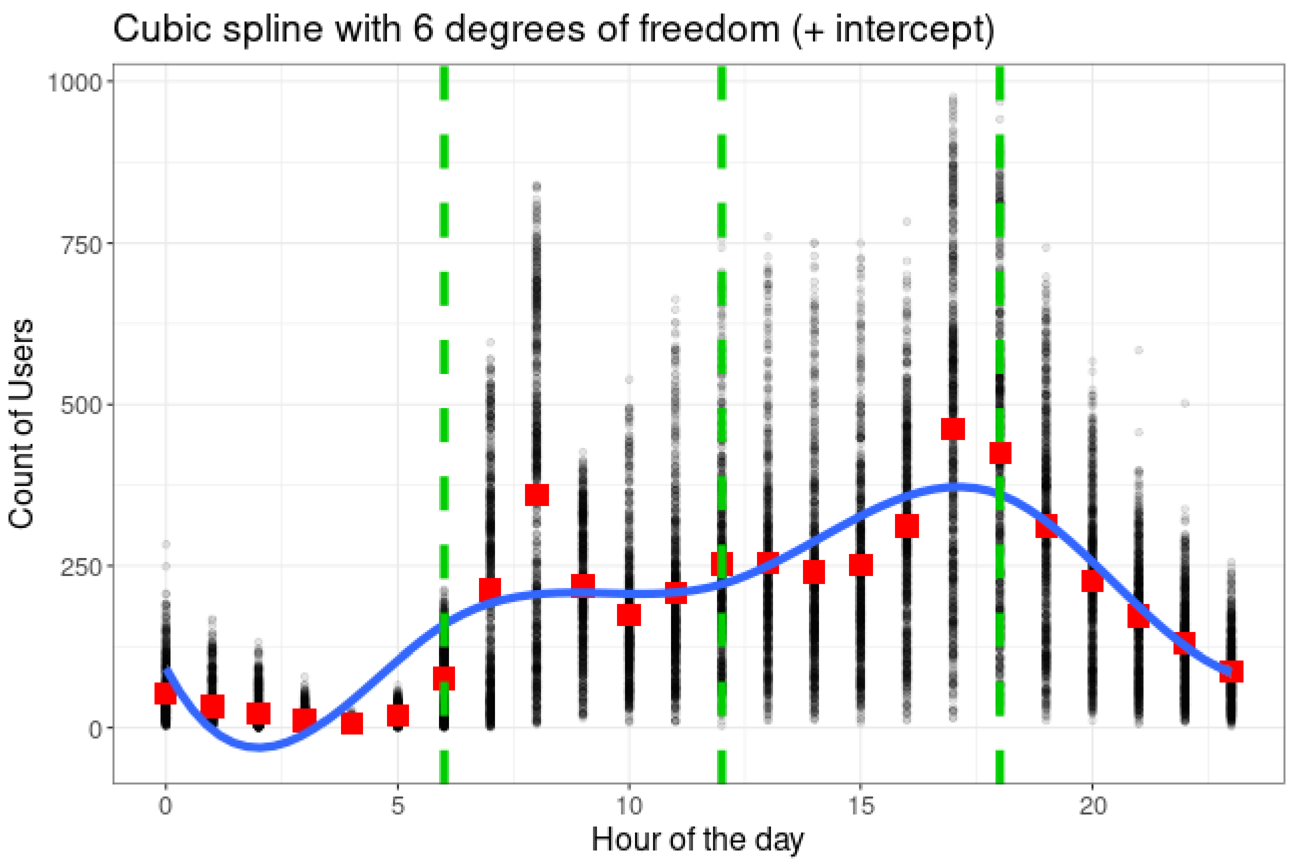
$$
\begin{aligned}
h_k(X) &= x^{k-1} & k &= 1, \ldots, K \\
h_{m+K}(X) &= (x - \xi_m)_+^{K-1} & m &= 1, \ldots, M
\end{aligned}
$$

where $(\cdot)_+$ denotes a function which returns $\max(\cdot, 0)$

- Splines have $M + K$ "degrees of freedom"
  - "Natural" splines add the constraint that the function is **linear** beyond the constraints of the data
- When the degree and number of knots are fixed, commonly called **regression splines**
  - Contrast with **smoothing splines** where number of indirectly controlled via penalization
- **Trade-offs**
  - Higher K and higher M = better in-sample fit but risks overfitting
  - Lower K and lower M = poorer in-sample fit (baseline is a global polynomial), but potentially more robust out-of-sample.

Splines

```
bike %>% ggplot(aes(x=hr, y=cnt)) + geom_point(alpha=.1) + geom_point(data=bike_by_hour, col="r
  xlab("Hour of the day") + ylab("Count of Users") + ggtitle("Cubic spline with 6 degrees of fr
  theme_bw() + theme(text = element_text(size = 16))
```



B-splines

- Recall that there are multiple ways of defining the basis functions that construct a spline.
  - Truncated power basis is interpretable but can have poor computational properties
- Alternative: **B-spline** basis
  - Define the spline basis **recursively**
  - Advantage: Non-zero over a limited domain
- For a sequence of $K + M$ knots $\tau_1, \tau_2, \ldots, \tau_{M+K}$

$$B_{i,1}(x) = \begin{cases} 1 & \text{if} \qquad\qquad \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x)$$

- **Intuition**: Basis functions for higher-order splines are weighted averages of the "neighboring" lower-order basis functions

B-splines

- R will generate a b-spline basis for you using the `bs()` function in `splines`
  - You can treat these like transformations of the regressors

```
bike_bs <- bs(bike$hr, df=6) # By default it's a cubic (degree = 3) and the intercept is omitte
head(bike_bs)
```

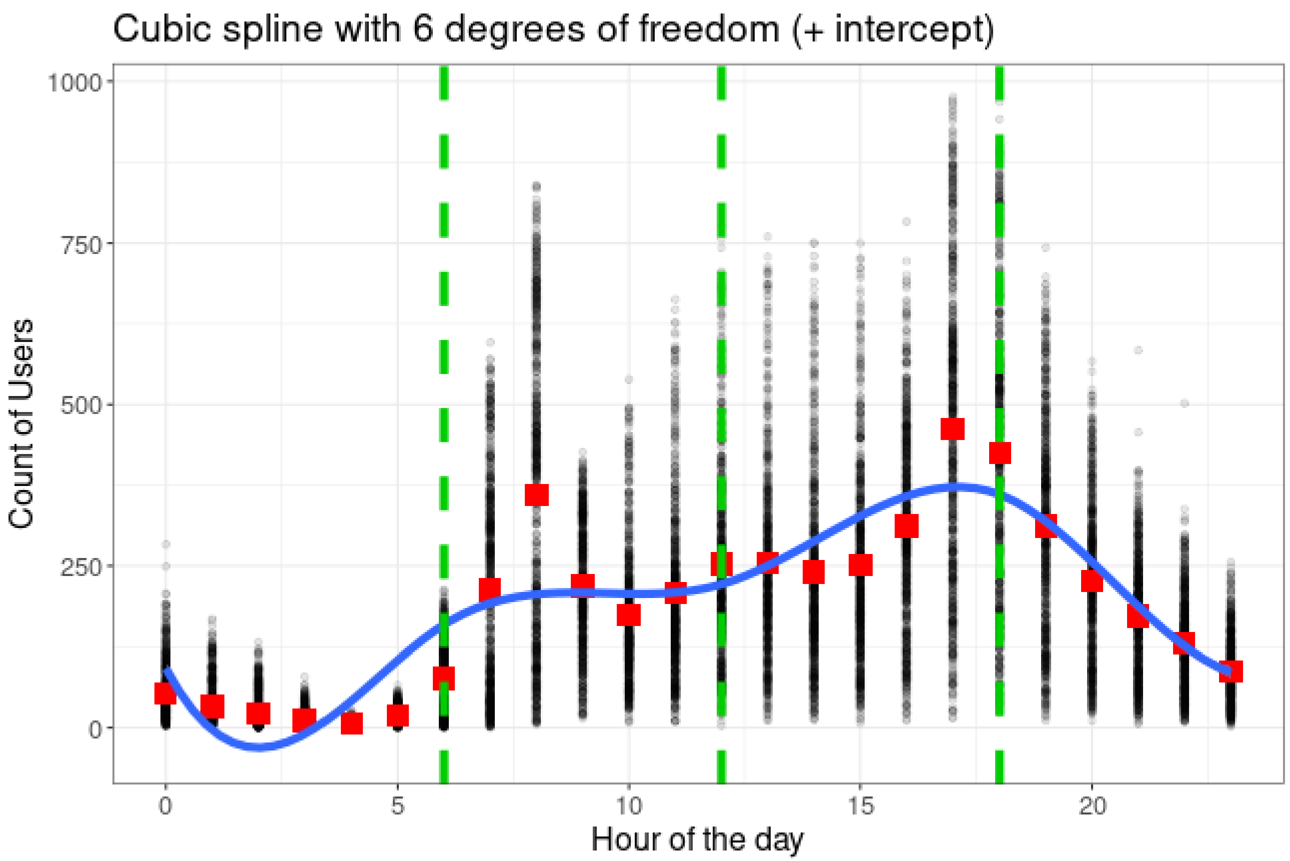
```
##             1      2        3 4 5 6
## [1,] 0.000 0.0000 0.000000 0 0 0
## [2,] 0.383 0.0374 0.000772 0 0 0
## [3,] 0.565 0.1327 0.006173 0 0 0
## [4,] 0.594 0.2604 0.020833 0 0 0
## [5,] 0.519 0.3951 0.049383 0 0 0
## [6,] 0.388 0.5112 0.096451 0 0 0
```

```
attributes(bike_bs)$knots
```

```
## 25% 50% 75%
##   6  12  18
```

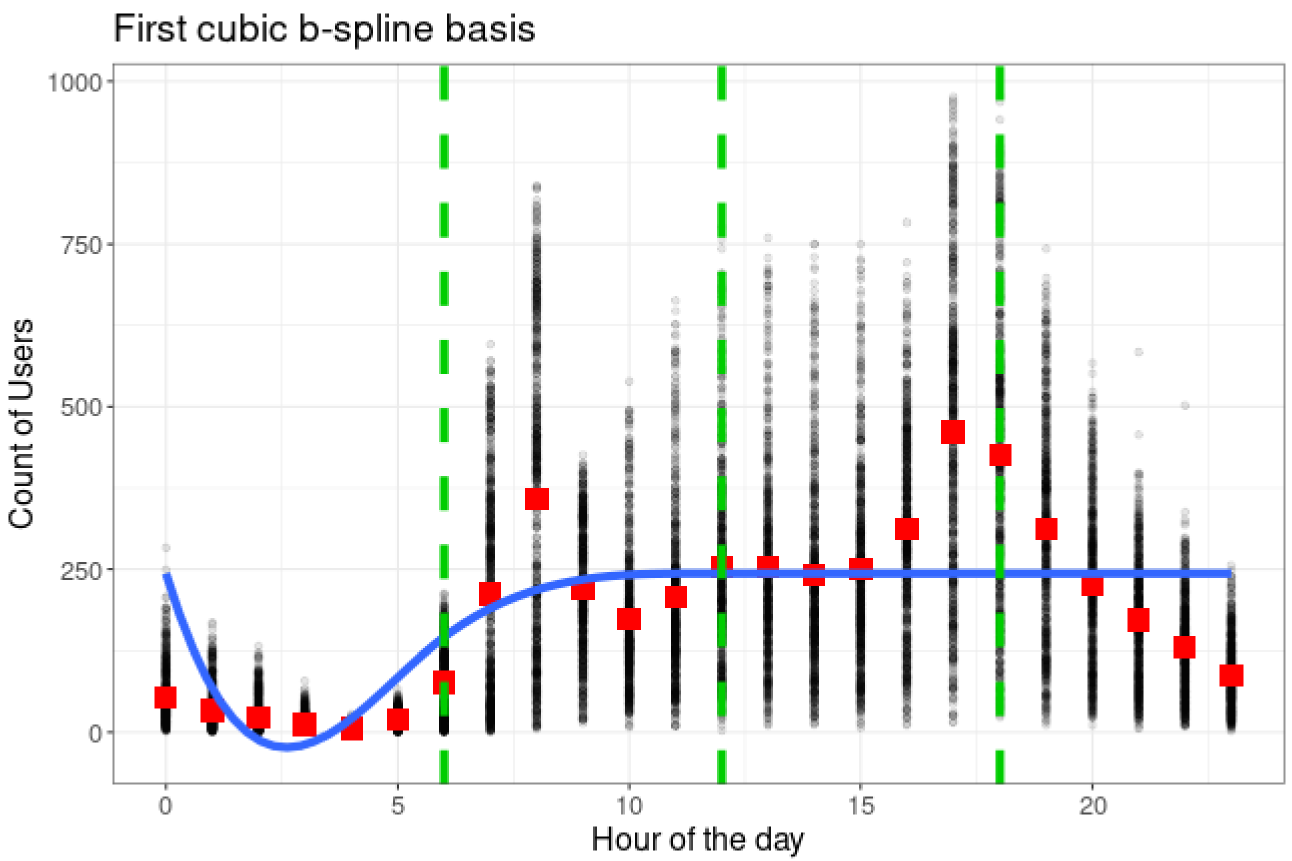
Splines

```
bike %>% ggplot(aes(x=hr, y=cnt)) + geom_point(alpha=.1) + geom_point(data=bike_by_hour, col="r
  xlab("Hour of the day") + ylab("Count of Users") + ggtitle("Cubic spline with 6 degrees of fr
  theme_bw() + theme(text = element_text(size = 16))
```



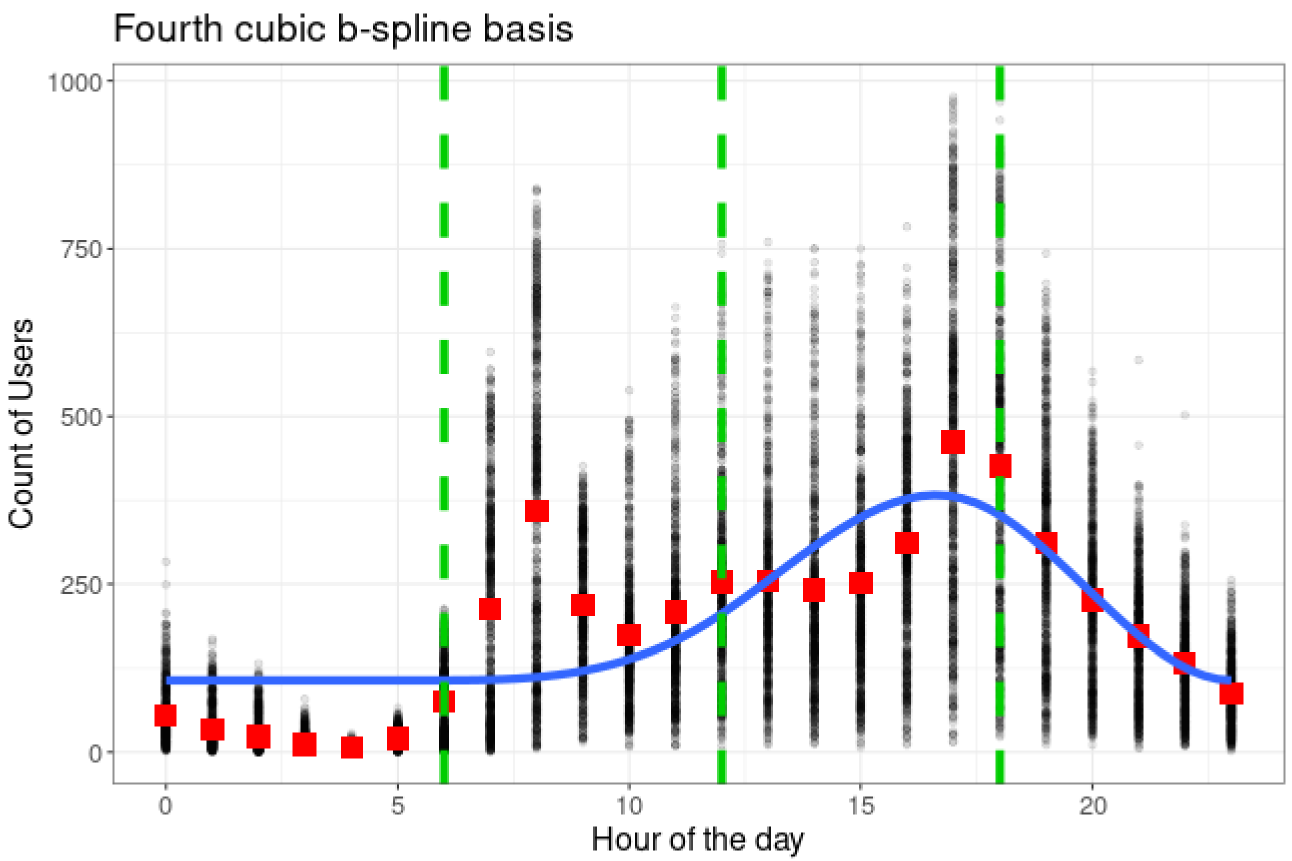
Splines

```
bike %>% ggplot(aes(x=hr, y=cnt)) + geom_point(alpha=.1) + geom_point(data=bike_by_hour, col="r
  xlab("Hour of the day") + ylab("Count of Users") + ggtitle("First cubic b-spline basis") + ge
  theme_bw() + theme(text = element_text(size = 16))
```



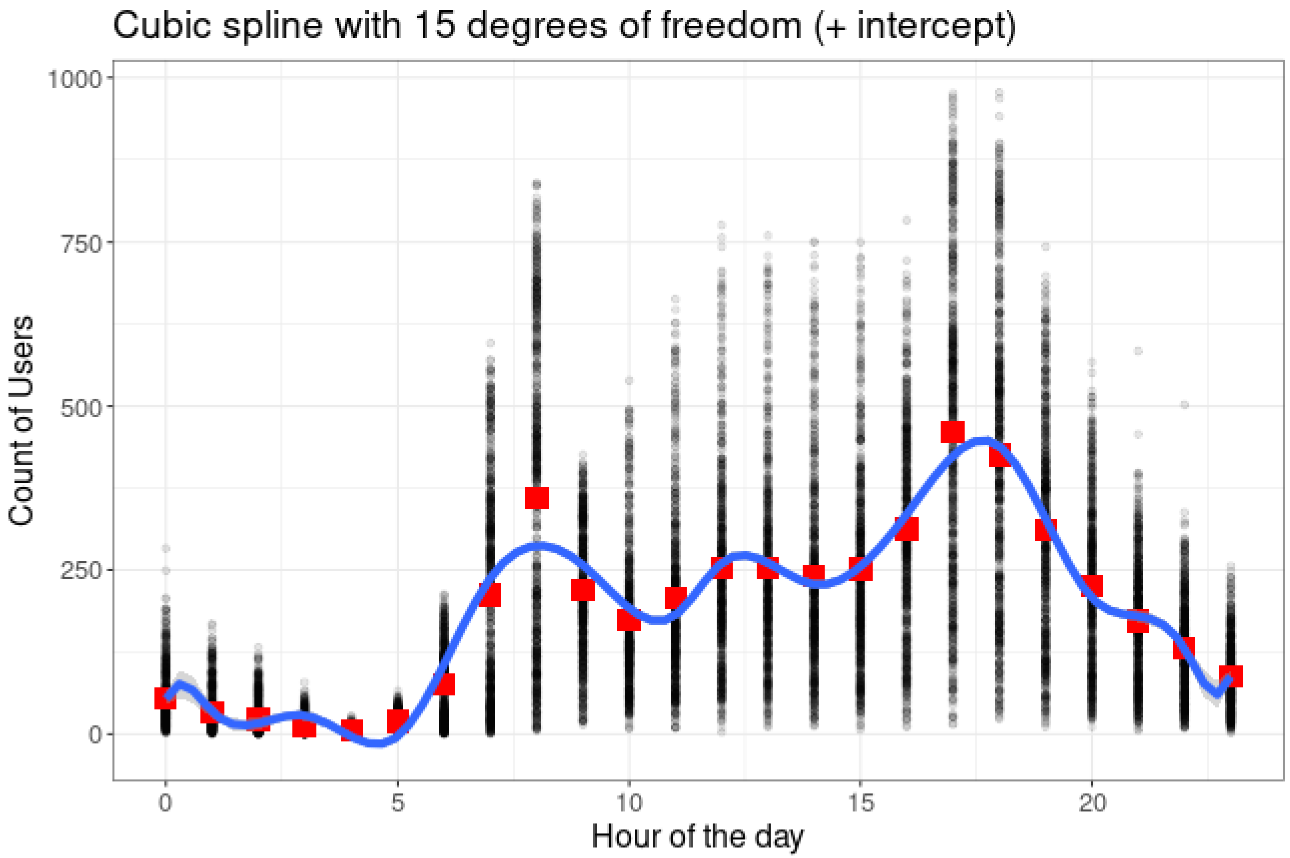
Splines

```
bike %>% ggplot(aes(x=hr, y=cnt)) + geom_point(alpha=.1) + geom_point(data=bike_by_hour, col="r
  xlab("Hour of the day") + ylab("Count of Users") + ggtitle("Fourth cubic b-spline basis") + g
  theme_bw() + theme(text = element_text(size = 16))
```



Splines

- Splines with many degrees of freedom have potentially weird behavior in areas with little data - "squiggly" interpolations



Smoothing splines

- What if we set the **maximum** possible number of knots
  - N total knots -- one for each observation
- Without penalization, we would have $N + 4$ parameters for a cubic spline for N observations.
  - This is not feasible using conventional least-squares - the solution is underdetermined!
- What if we controlled the fit via some **penalty** parameter
  - All-else-equal, we'd prefer a fit where the regression function is not very "jumpy"
  - Formalize this in terms of the second-derivative $f''(x)$
- Our "smoothing spline" takes the form of a **penalized** optimization problem. We want to find the function $f(x)$ that minimizes:

$$\sum_{i=1}^{n}(Y_i - f(X_i))^2 + \lambda \int f''(t)dt$$

where λ is a non-negative "tuning" or "penalty" parameter.

Smoothing splines

- It turns out that the function $f()$ that optimizes the "smoothing spline" objective function has some useful features
  1. It's a piecewise cubic polynomial
  2. It has knots at the unique values of the data $x_1, x_2, \ldots, x_N$
  3. It has continuous first and second derivatives at each of the knots.
  4. It's linear outside of the knots
- It's a **natural cubic spline**
  - But with **penalized** parameter estimates (shrunken towards zero)
- λ chosen via cross-validation
  - Can conduct leave-one-out cross-validation very easily (formula exists to use the fit for all observations, so no need to re-fit)

Generalized Additive Models

- **Generalized Additive Models** (GAMs) allow us to extend the conventional multiple linear regression model to accomodate the non-linear transformations of X_i.
- Instead of our original linear model:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \epsilon_i$$

- We fit:

$$Y_i = \beta_0 + f_1(X_{i1}) + f_2(X_{i2}) + \ldots + \epsilon_i$$

- With conventional regression splines for $f_1()$, $f_2()$, $f_3()$, etc..., this just becomes a giant linear regression with the spline bases substituted for the original regressors
- With *smoothing splines*, slightly more complicated - can't use OLS, but conventional software (`gam()` in R) implements the "backfitting" algorithm.
- Can extend to other functions $f_i()$ such as local regressions or just plain polynomials

Generalized Additive Models

- Fitting our smoothing spline using the `gam()` function in the `mgcv` library

```
hour_fit <- gam(cnt ~ s(hr, bs="cr"), data = bike)
summary(hour_fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## cnt ~ s(hr, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  189.463      0.992     191   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##       edf Ref.df    F p-value
## s(hr)   9      9 1785  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.48   Deviance explained =   48%
## GCV =  17111  Scale est. = 17101     n = 17379
```

Generalized Additive Models

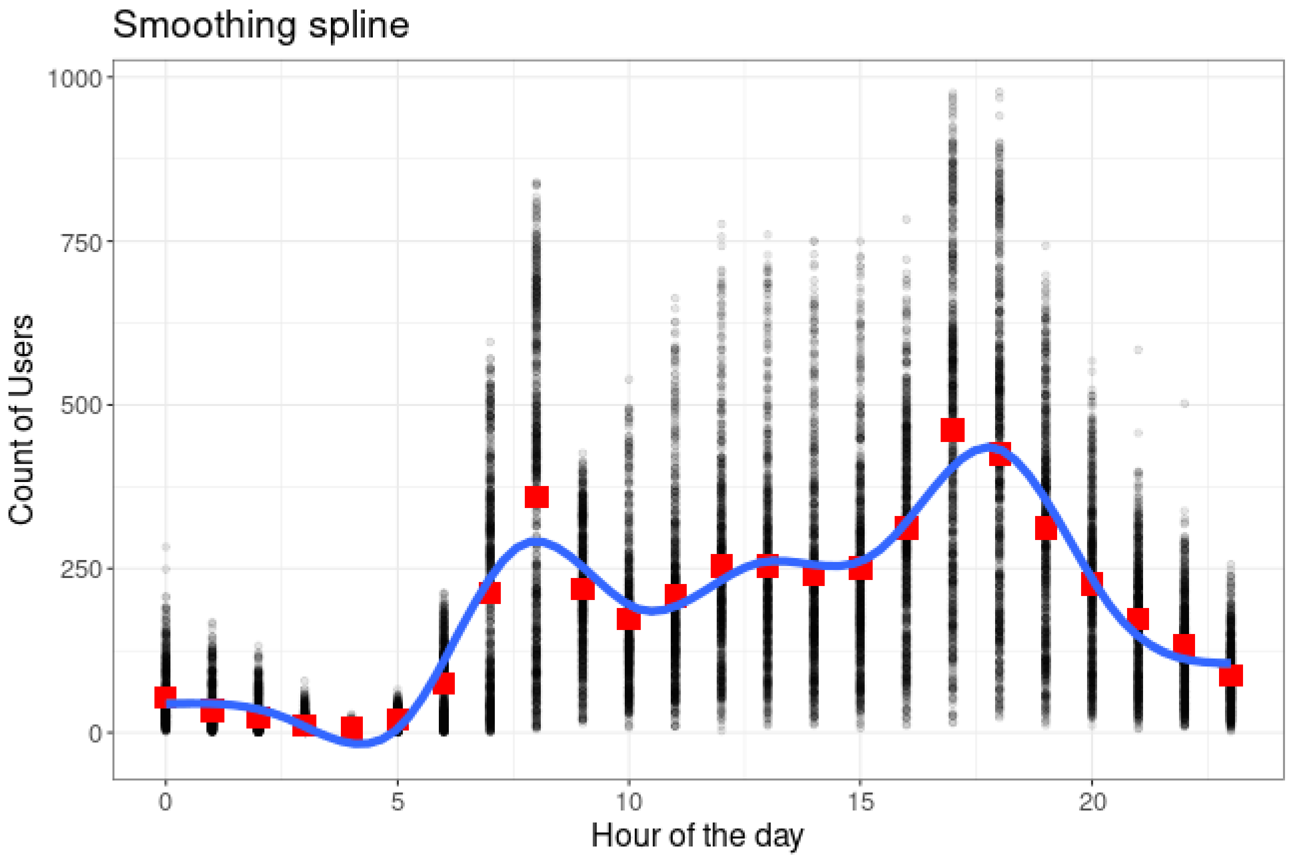
- We can combine "parametric" and "non-parametric" terms in the `gam()` function

```
hour_work_fit <- gam(cnt ~ workingday + s(hr, bs="cr"), data=bike)
summary(hour_work_fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## cnt ~ workingday + s(hr, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   181.73       1.76  103.27  < 2e-16 ***
## workingday     11.33       2.13    5.32  1.1e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##       edf Ref.df    F p-value
## s(hr)   9      9 1787  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.481   Deviance explained = 48.1%
```

Generalized Additive Models

- Smoothing splines are also the default in `geom_smooth()` for large datasets



Regularization