Non-Linear Regression

Business Analytics
Stefan Feuerriegel

Outline

Outline

1. Trend Lines in ggplot2
2. Polynomial Regression
3. Step-Wise Functions
4. Generalized Additive Models
5. Wrap-Up

Linear Trend Line

- Load dataset to visualize the relationship between age and wage

```
library(ISLR)
data(Wage)

# fewer points look nicer on slides
Wage.small <- Wage[1:250, ]
```

- Load package `ggplot2` for visualization

```
library(ggplot2)
```

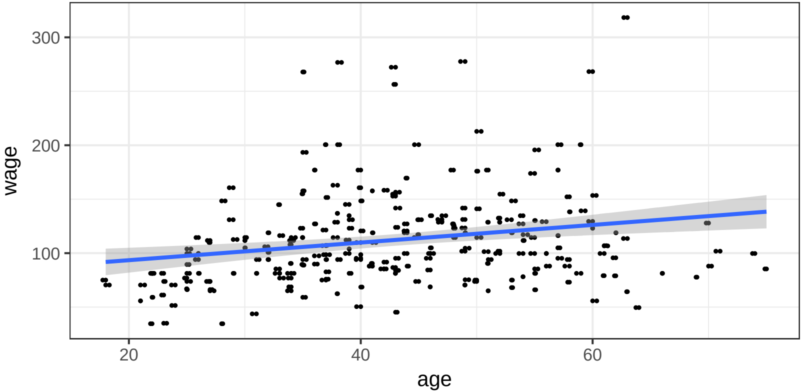
- Plot linear trend line, (→ OLS via `method="lm"`)

```
ggplot(Wage.small, aes(x=age, y=wage)) +
  geom_point(size=0.5) +
  geom_jitter(size=0.5) +
  geom_smooth(method="lm") +
  theme_bw()
```

- `geom_jitter(...)` jitters points to reduce overlaps
- `geom_smooth(...)` is a default way to add smoothed lines

Linear Trend Line

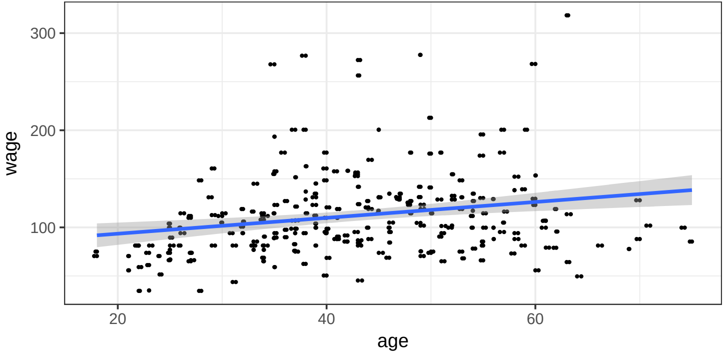
- Blue line is linear trend line with standard errors (gray area)



GAM Smoothing in ggplot2

- `ggplot2` has a built-in support for GAM via `method="gam"`

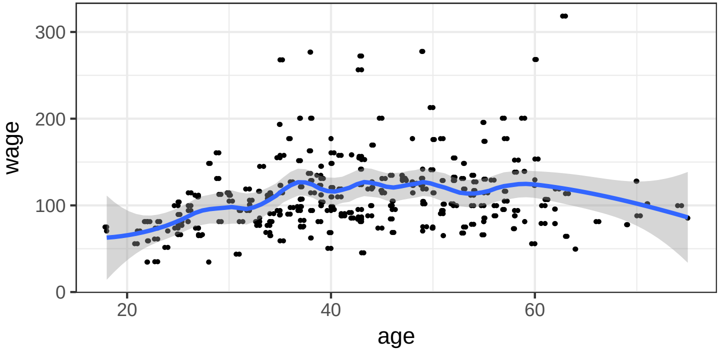
```
ggplot(Wage.small, aes(x=age, y=wage)) +
  geom_point(size=0.5) +
  geom_jitter(size=0.5) +
  geom_smooth(method="gam") +
  theme_bw()
```



LOESS in ggplot2

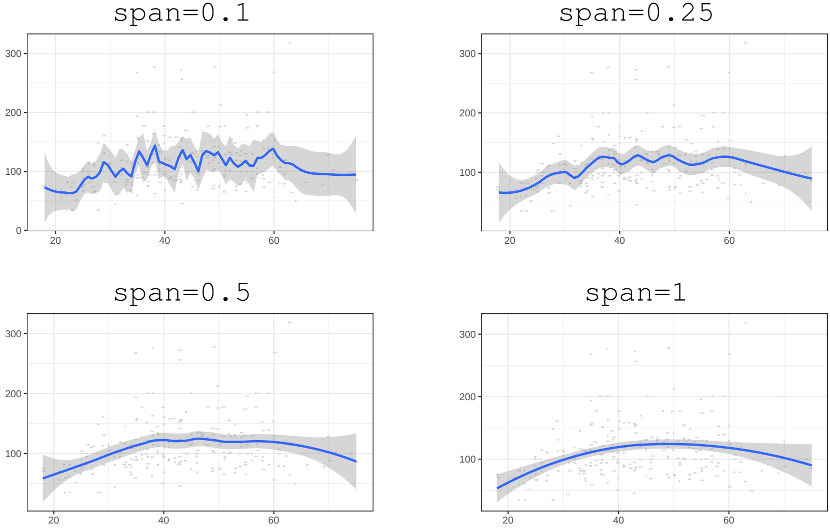
- `ggplot2` has a built-in support for LOESS via `method="loess"`

```
ggplot(Wage.small, aes(age, wage)) +
  geom_point(size=0.5) +
  geom_jitter(size=0.5) +
  geom_smooth(method="loess", span=0.3) +
  theme_bw()
```



LOESS in ggplot2

Parameter `span` controls the intensity of smoothing



Outline

Polynomial Regression in R

- Generate sample data

```
set.seed(0)
x <- runif(50, min=0, max=100)
y <- sin(x/50*pi) + runif(50, min=-0.5, max=0.5)
```

- Generate polynomial terms of up to degree *d* via `poly(x, degree=d, raw=TRUE)`, then perform least squares

```
m <- lm(y ~ poly(x, degree=3, raw=TRUE))
```

  Note: `raw=TRUE` chooses default polynomials; else it uses orthogonal ones which are numerically more convenient

- Manual alternative

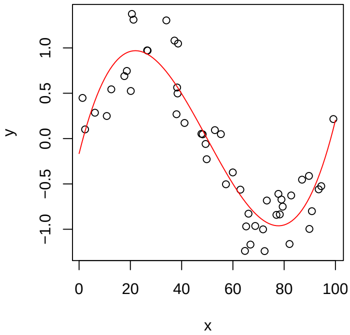
```
m <- lm(y ~ x + I(x^2) + I(x^3))
```

  Note: `I(...)` is necessary to interpret arithmetic operations in a formula as such

Polynomial Regression in R

- Visualize result by manually generating the fitted line

```
predict_x <- seq(from=0, to=100, by=1)
# Named dataframe to avoid generating polynomial terms
predict_y <- predict(m, newdata=data.frame(x=predict_x))

plot(x, y)
lines(predict_x, predict_y, col="red")
```



Polynomial Regression in R

- ANOVA tests can identify the best-fit model

```
m.d2 <- lm(y ~ poly(x, degree=2, raw=TRUE))
m.d3 <- lm(y ~ poly(x, degree=3, raw=TRUE))
m.d4 <- lm(y ~ poly(x, degree=4, raw=TRUE))
anova(m.d2, m.d3, m.d4)

## Analysis of Variance Table
##
## Model 1: y ~ poly(x, degree = 2, raw = TRUE)
## Model 2: y ~ poly(x, degree = 3, raw = TRUE)
## Model 3: y ~ poly(x, degree = 4, raw = TRUE)
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)
## 1     47 12.1890
## 2     46  3.8570  1    8.3321 97.4464 7.786e-13 ***
## 3     45  3.8477  1    0.0093  0.1085    0.7434
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The P-value comparing $d = 2$ and $d = 3$ is almost zero
- Quadratic model is not sufficient $\rightarrow$ cubic is preferred

Outline

Stew-Wise Functions in R

- Generate sample data

```r
set.seed(0)
x <- c(runif(20, min=0, max=40), runif(20, min=40, max=100))
y <- c(runif(20, min=0, max=10), runif(20, min=30, max=40))
y <- y + runif(40, min=-5, max=5)
```

- Estimate linear model with dummies

```r
m <- lm(y ~ I(x < 40))
coef(m)

##   (Intercept) I(x < 40)TRUE
##      35.27628     -30.15988
```

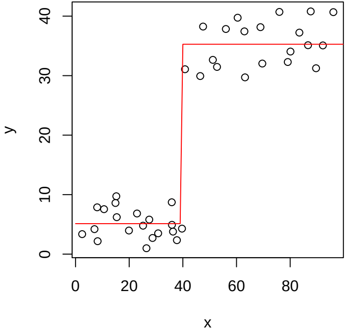
- Alternative is to split data via `cut(x, breaks=...)`

```r
x2 <- cut(x, breaks=c(0, 40, 100))
coef(lm(y ~ x2))

## (Intercept)  x2(40,100]
##    5.116405   30.159878
```

Step-Wise Regression in R

- Visualize result by manually generating the fitted line

```
predict_x <- seq(from=0, to=100, by=1)
# Named dataframe to avoid generating polynomial terms
predict_y <- predict(m, newdata=data.frame(x=predict_x))

plot(x, y)
lines(predict_x, predict_y, col="red")
```



Outline

GAM in R

- Load the `gam` package

```
library(gam)
```

- Estimate model, e. g. with smoothing splines

```
m.gam <- gam(wage ~ s(year, 4) + s(age, 5) + education,
             data=Wage)
m.gam

## Call:
## gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wa
##
## Degrees of Freedom: 2999 total; 2986 Residual
## Residual Deviance: 3689770
```

- `s(variable, df)` introduces smoothing splines of degree `df`
- `ns(variable, df)` are natural splines
- `education` is a factor and thus not treated
- Detailed summary on results via

```
summary(m.gam)
```

GAM in R

- ANOVA test identifies best-fit model
  $\rightarrow$ e. g. excluding `year` or assuming a linear or non-linear effect

```
m.gam1 <- gam(wage ~ s(age, 5) + education, data=Wage)
m.gam2 <- gam(wage ~ year + s(age, 5) + education, data=Wage)
m.gam3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data=Wage)
anova(m.gam1, m.gam2, m.gam3)

## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 0.0001419 ***
## 3      2986    3689770  3   4071.1 0.3483897
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- GAM with linear `year` is better than without (P-value < 0.001)
- Non-linear effect of `year` is not necessary (P-value > 0.05)

Outline

Wrap-Up

- `ggplot(...)` is helpful to quickly gain first insights or for nice visualizations
- ggplot2 uses LOESS by default for up to 1000 data points, otherwise GAM
- Package `mgcv` is a newer alternative to `gam`