Supervised Learning

Linear regression

June 21st, 2021

Simple linear regression

We assume a **linear relationship** for $Y = f(X)$:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad \text{for } i = 1, 2, \ldots, n$$

- Y_i is the ith value for the **response** variable
- X_i is the ith value for the **predictor** variable
- β_0 is an *unknown*, constant **intercept**: average value for Y if $X = 0$
- β_1 is an *unknown*, constant **slope**: increase in average value for Y for each one-unit increase in X
- ϵ_i is the **random** noise: assume **independent, identically distributed** (*iid*) from Normal distribution

$$\epsilon_i \overset{iid}{\sim} N(0, \sigma^2) \quad \text{with constant variance } \sigma^2$$

Simple linear regression estimation

We are estimating the **conditional expection (mean)** for Y:

$$\mathbb{E}[Y_i|X_i] = \beta_0 + \beta_1 X_i$$

- average value for Y given the value for X

How do we estimate the **best fitting** line?

Ordinary least squares (OLS) - by minimizing the **residual sum of squares (RSS)**

$$RSS\left(\beta_0, \beta_1\right) = \sum_{i=1}^{n} \left[Y_i - (\beta_0 + \beta_1 X_i)\right]^2 = \sum_{i=1}^{n} \left(Y_i - \beta_0 - \beta_1 X_i\right)^2$$

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2} \quad \text{and} \quad \widehat{\beta}_0 = \bar{Y} - \widehat{\beta}_1 \bar{X}$$

where $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$ and $\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$

Connection to covariance and correlation

Covariance describes the **joint variability of two variables**

$$\mathrm{Cov}(X, Y) = \sigma_{X,Y} = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

We compute the **sample covariance** (use $n-1$ since we are using the means and want **unbiased estimates**)

$$\hat{\sigma}_{X,Y} = \frac{1}{n-1} \sum_{i=1}^{n} \left(X_i - \bar{X}\right) \left(Y_i - \bar{Y}\right)$$

Correlation is a *normalized* form of covariance, ranges from -1 to 1

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

Sample correlation uses the sample covariance and standard deviations, e.g. $s_X^2 = \frac{1}{n-1} \sum_i (X_i - \bar{X})^2$

$$r_{X,Y} = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right) \left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 \sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2}}$$

Connection to covariance and correlation

So we have the following:

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2} \quad \text{compared to} \quad r_{X,Y} = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 \sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2}}$$

$\Rightarrow$ Can rewrite $\hat{\beta}_1$ as:

$$\widehat{\beta}_1 = r_{X,Y} \cdot \frac{s_Y}{s_X}$$

$\Rightarrow$ Can rewrite $r_{X,Y}$ as:

$$r_{X,Y} = \widehat{\beta}_1 \cdot \frac{s_X}{s_Y}$$

Can think of $\widehat{\beta}_1$ weighting the ratio of variance between X and Y…

Example data: NFL teams summary

Created dataset using `nflfastR` summarizing NFL team performances from 1999 to 2020

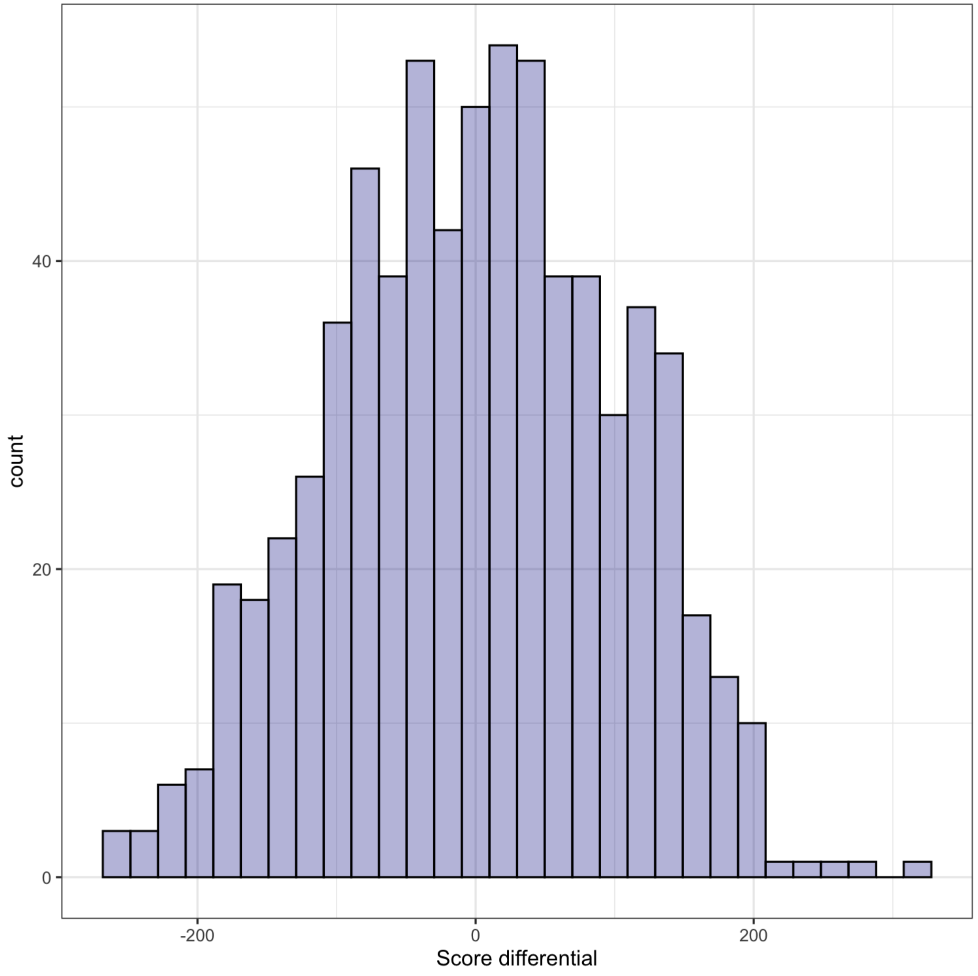
```r
library(tidyverse)
nfl_teams_data <- read_csv("http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/regression_pro
nfl_teams_data
```

```
## # A tibble: 701 × 55
##    season team  offens…¹ offen…² offen…³ offen…⁴ offen…⁵ offen…⁶ offen…⁷ offen…⁸
##     <dbl> <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1   1999 ARI      0.477    2796    1209    4.67    3.15       0     NaN       0
##  2   1999 ATL      0.504    3317    1176    6.08    3.20       0     NaN      11
##  3   1999 BAL      0.452    2805    1663    5.07    4.13       0     NaN       0
##  4   1999 BUF      0.540    3275    2038    6.17    4.13       0     NaN     161
##  5   1999 CAR      0.552    4144    1484    6.68    4.29       0     NaN      89
##  6   1999 CHI      0.561    4090    1359    5.75    3.55       0     NaN     508
##  7   1999 CIN      0.498    3178    1971    5.37    4.63       0     NaN       0
##  8   1999 CLE      0.489    2574    1140    4.71    3.67       0     NaN      35
##  9   1999 DAL      0.560    3083    2054    5.95    4.29       0     NaN       0
## 10   1999 DEN      0.546    3378    1852    5.85    4.05       0     NaN       9
## # … with 691 more rows, 45 more variables: offense_ave_yac <dbl>,
## #   offense_n_plays_pass <dbl>, offense_n_plays_run <dbl>,
## #   offense_n_interceptions <dbl>, offense_n_fumbles_lost_pass <dbl>,
## #   offense_n_fumbles_lost_run <dbl>, offense_total_epa_pass <dbl>,
```

Modeling NFL score differential

Interested in modeling a team's **score differential**

```r
nfl_teams_data <- nfl_teams_data %>%
  mutate(score_diff =
           points_scored - points_allowed)
nfl_teams_data %>%
  ggplot(aes(x = score_diff)) +
  geom_histogram(color = "black",
                 fill = "darkblue",
                 alpha = 0.3) +
  theme_bw() +
  labs(x = "Score differential")
```



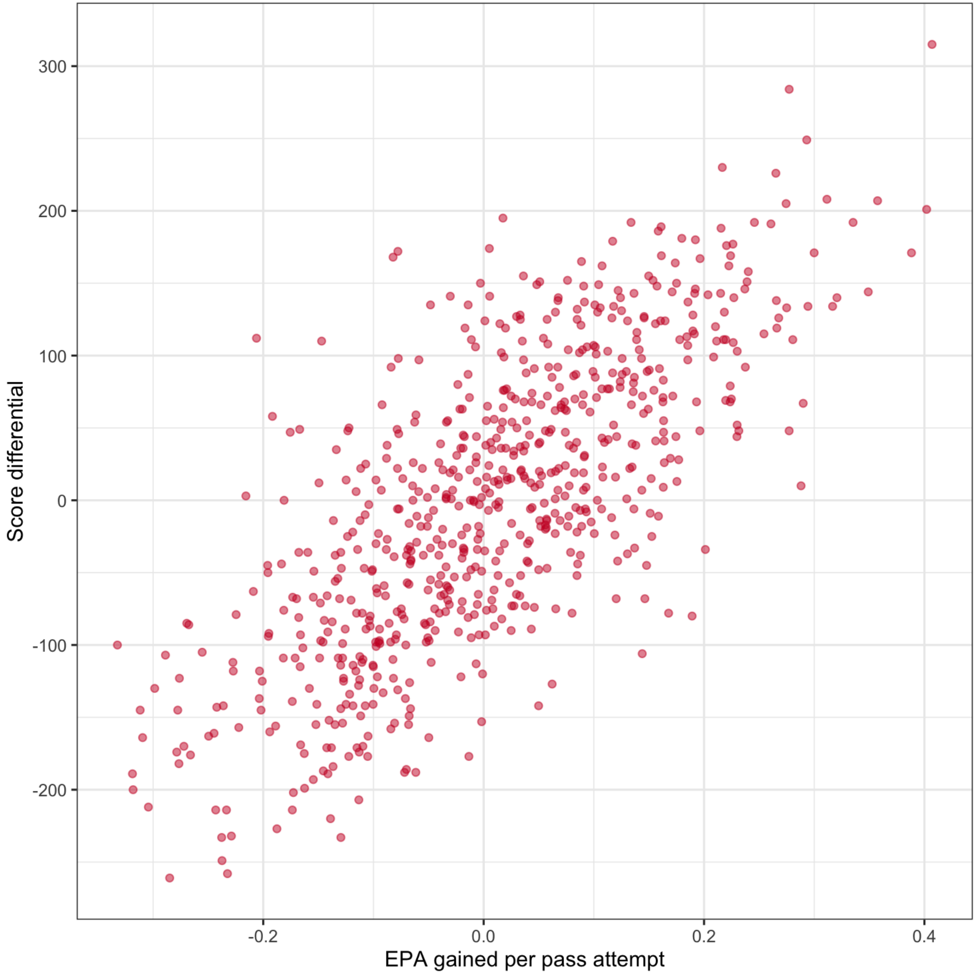
Relationship between score differential and passing performance

- `offense_ave_epa_pass`: team's average expected points added (EPA) per pass attempt

```
nfl_plot <- nfl_teams_data %>%
  ggplot(aes(x = offense_ave_epa_pass,
             y = score_diff)) +
  geom_point(alpha = 0.5) +
  theme_bw() +
  labs(x = "EPA gained per pass attempt",
       y = "Score differential")
nfl_plot
```

We fit linear regression models using `lm()`, formula is input as: `response ~ predictor`

```
init_lm <- lm(score_diff ~ offense_ave_epa_pa
              data = nfl_teams_data)
```



View the model `summary()`

```
summary(init_lm)
```

```
##
## Call:
## lm(formula = score_diff ~ offense_ave_epa_pass, data = nfl_teams_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -182.611  -46.600   -1.326   44.123  233.640
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -4.890      2.561  -1.909   0.0566 .
## offense_ave_epa_pass  566.352     19.226  29.458   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 67.67 on 699 degrees of freedom
## Multiple R-squared:  0.5539,    Adjusted R-squared:  0.5532
## F-statistic: 867.8 on 1 and 699 DF,  p-value: < 2.2e-16
```

Inference with OLS

Reports the intercept and coefficient estimates: $\hat{\beta}_0 \approx -4.89 \quad , \quad \hat{\beta}_1 \approx 566.352$

Estimates of uncertainty for βs via **standard errors**: $\widehat{SE}(\hat{\beta}_0) \approx 2.561 \quad , \quad \widehat{SE}(\hat{\beta}_1) \approx 19.226$

t-statistics are coefficients `Estimates` / `Std. Error`, i.e., number of standard deviations from 0

- *p-values* (i.e., `Pr(>|t|)`): estimated probability observing value as extreme as `|t value|` **given the null hypothesis** $\beta = 0$
- p-value $<$ conventional threshold of $\alpha = 0.05$, **sufficient evidence to reject the null hypothesis that the coefficient is zero**,
- Typically `|t values|` > 2 indicate **significant** relationship at $\alpha = 0.05$
- i.e., there is a **significant** association between `offense_ave_epa_pass` and `score_diff`

Be careful!

Caveats to keep in mind regarding p-values:

If the true value of a coefficient $\beta = 0$, then the p-value is sampled from a Uniform(0,1) distribution

- i.e. it is just as likely to have value 0.45 as 0.16 or 0.84 or 0.9999 or 0.00001...

$\Rightarrow$ Hence why we typically only reject for low α values like 0.05

- Controlling the Type 1 error rate at $\alpha = 0.05$, i.e., the probability of a **false positive** mistake
- 5% chance that you'll conclude there's a significant association between x and y **even when there is none**

Remember what a standard error is? $SE = \frac{\sigma}{\sqrt{n}}$

- $\Rightarrow$ As n gets large **standard error goes to zero**, and *all* predictors are eventually deemed significant
- While the p-values might be informative, we will explore other approaches to determine which subset of predictors to include (e.g., holdout performance)

Back to the model summary: `Multiple R-squared`

Back to the connection between the coefficient and correlation:

$$r_{X,Y} = \widehat{\beta}_1 \cdot \frac{s_X}{s_Y} \quad \Rightarrow \quad r^2_{X,Y} = \widehat{\beta}_1^2 \cdot \frac{s^2_X}{s^2_Y}$$

Compute the correlation with `cor()`:

```
with(nfl_teams_data, cor(offense_ave_epa_pass, score_diff))
```

```
## [1] 0.7442135
```

The squared `cor` matches the reported `Multiple R-squared`

```
with(nfl_teams_data, cor(offense_ave_epa_pass, score_diff))^2
```

```
## [1] 0.5538537
```

Back to the model summary: `Multiple R-squared`

Back to the connection between the coefficient and correlation:

$$r_{X,Y} = \widehat{\beta}_1 \cdot \frac{s_X}{s_Y} \quad \Rightarrow \quad r^2_{X,Y} = \widehat{\beta}_1^2 \cdot \frac{s^2_X}{s^2_Y}$$

r^2 (or also R^2) estimates the **proportion of the variance** of Y explained by X

- More generally: variance of model predictions / variance of Y

```
var(predict(init_lm)) / var(nfl_teams_data$score_diff)
```

```
## [1] 0.5538537
```

Generating predictions

We can use the `predict()` function to either get the fitted values of the regression:

```
train_preds <- predict(init_lm)
head(train_preds)
```

```
##          1          2          3          4          5          6 
## -120.21628  -33.58829 -104.31202   21.15045   57.18906  -23.34489 
```

Which is equivalent to using:

```
head(init_lm$fitted.values)
```

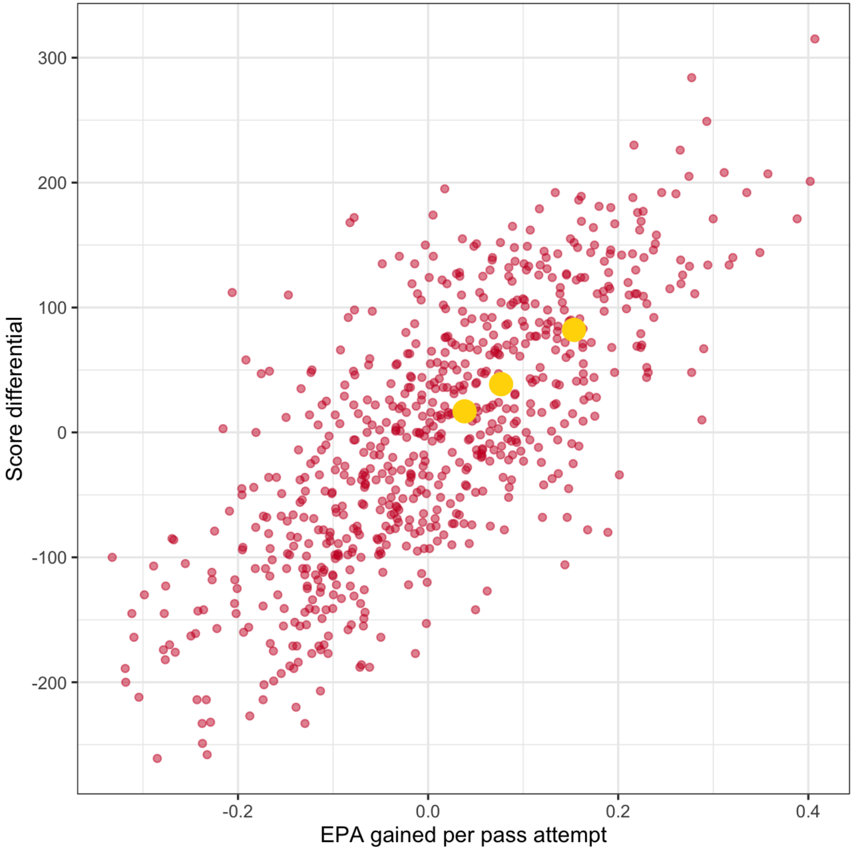
```
##          1          2          3          4          5          6 
## -120.21628  -33.58829 -104.31202   21.15045   57.18906  -23.34489 
```

Predictions for new data

Or we can provide it `newdata` which **must contain the explanatory variables**:

```
steelers_data <- nfl_teams_data %>%
  filter(team == "PIT", season == 2020)

new_steelers_data <- steelers_data %>%
  dplyr::select(team, offense_ave_epa_pass) %
  slice(rep(1, 3)) %>%
  mutate(adj_factor = c(0.5, 1, 2),
         offense_ave_epa_pass = offense_ave_e
new_steelers_data$pred_score_diff <-
  predict(init_lm, newdata = new_steelers_dat
nfl_plot +
  geom_point(data = new_steelers_data,
             aes(x = offense_ave_epa_pass,
                 y = pred_score_diff),
             color = "gold", size = 5)
```

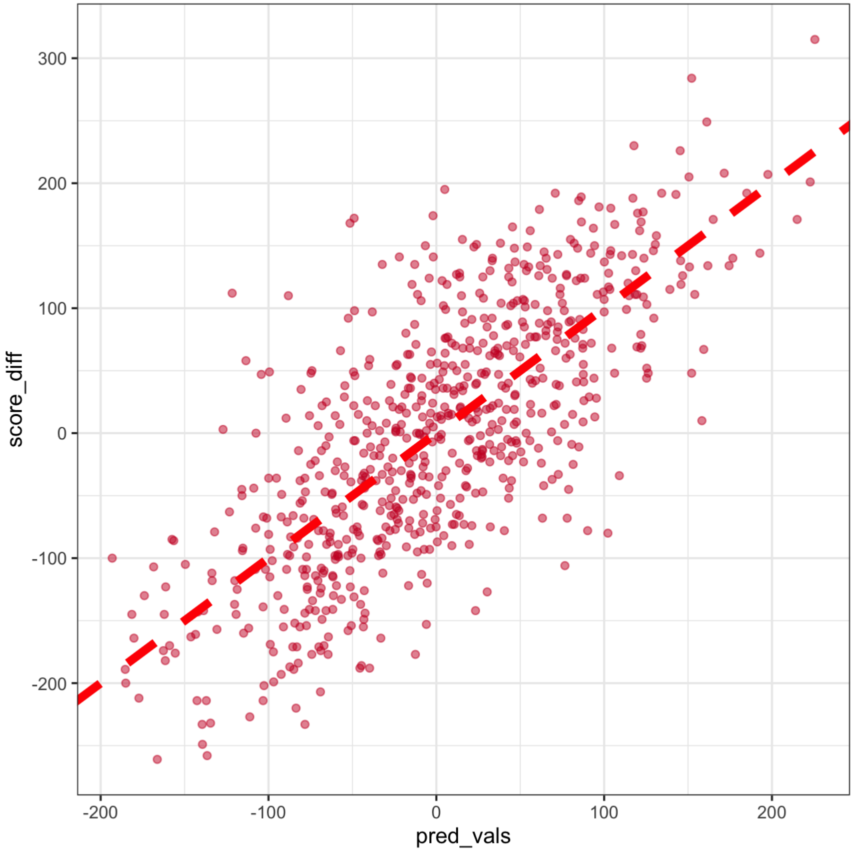


Plot observed values against predictions

Useful diagnostic (for **any type of model**, not just linear regression!)

```
nfl_teams_data %>%
  mutate(pred_vals = predict(init_lm)) %>%
  ggplot(aes(x = pred_vals,
             y = score_diff)) +
  geom_point(alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0,
              linetype = "dashed",
              color = "red",
              size = 2) +
  theme_bw()
```

- "Perfect" model will follow **diagonal**

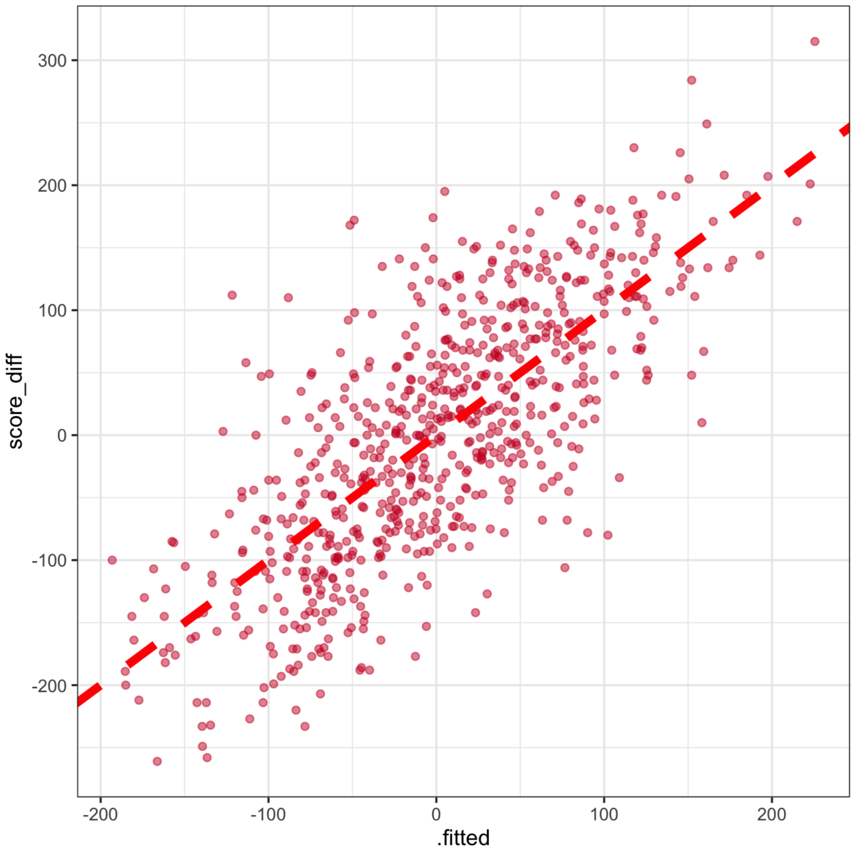


Plot observed values against predictions

Can augment the data with model output using the `broom` package

```
nfl_teams_data <-
  broom::augment(init_lm, nfl_teams_data)
nfl_teams_data %>%
  ggplot(aes(x = .fitted,
             y = score_diff)) +
  geom_point(alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0,
              linetype = "dashed",
              color = "red",
              size = 2) +
  theme_bw()
```

- Adds various columns from model fit we can use in plotting for model diagnostics

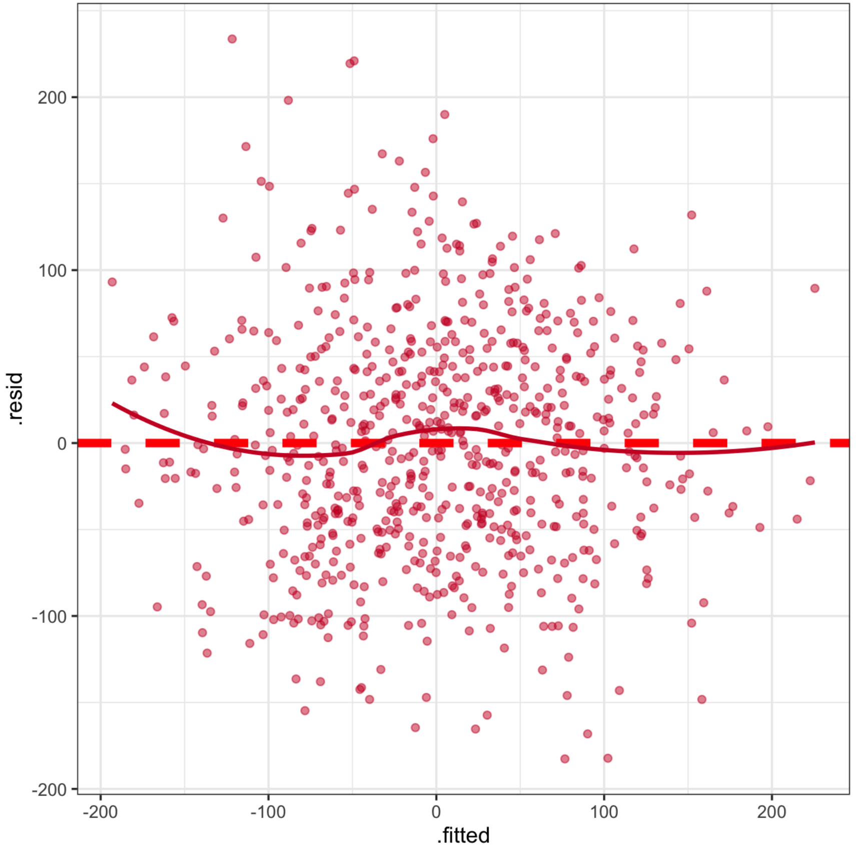


Plot residuals against predicted values

- Residuals = observed - predicted
- Conditional on the predicted values, the **residuals should have a mean of zero**

```
nfl_teams_data %>%
  ggplot(aes(x = .fitted,
             y = .resid)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "red",
             size = 2) +
  # To plot the residual mean
  geom_smooth(se = FALSE) +
  theme_bw()
```

- Residuals **should NOT display any pattern**



Multiple regression

We can include as many variables as we want (assuming $n > p$!)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

OLS estimates in matrix notation ($\boldsymbol{X}$ is a $n \times p$ matrix):

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{Y}$$

Can just add more variables to the formula in `R`

```
multiple_lm <- lm(score_diff ~ offense_ave_epa_pass + defense_ave_epa_pass,
                  data = nfl_teams_data)
```

- Use the `Adjusted R-squared` when including multiple variables $= 1 - \frac{(1-R^2)(n-1)}{(n-p-1)}$
  - Adjusts for the number of variables in the model p
  - Adding more variables **will always increase** `Multiple R-squared`

What about the Normal distribution assumption???

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

- ϵ_i is the **random** noise: assume **independent, identically distributed** (*iid*) from Normal distribution

$$\epsilon_i \overset{iid}{\sim} N(0, \sigma^2) \quad \text{with constant variance } \sigma^2$$

OLS doesn't care about this assumption, it's just estimating coefficients!

In order to perform inference, **we need to impose additional assumptions**

By assuming $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$, what we really mean is:

$$Y \overset{iid}{\sim} N(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p, \sigma^2)$$

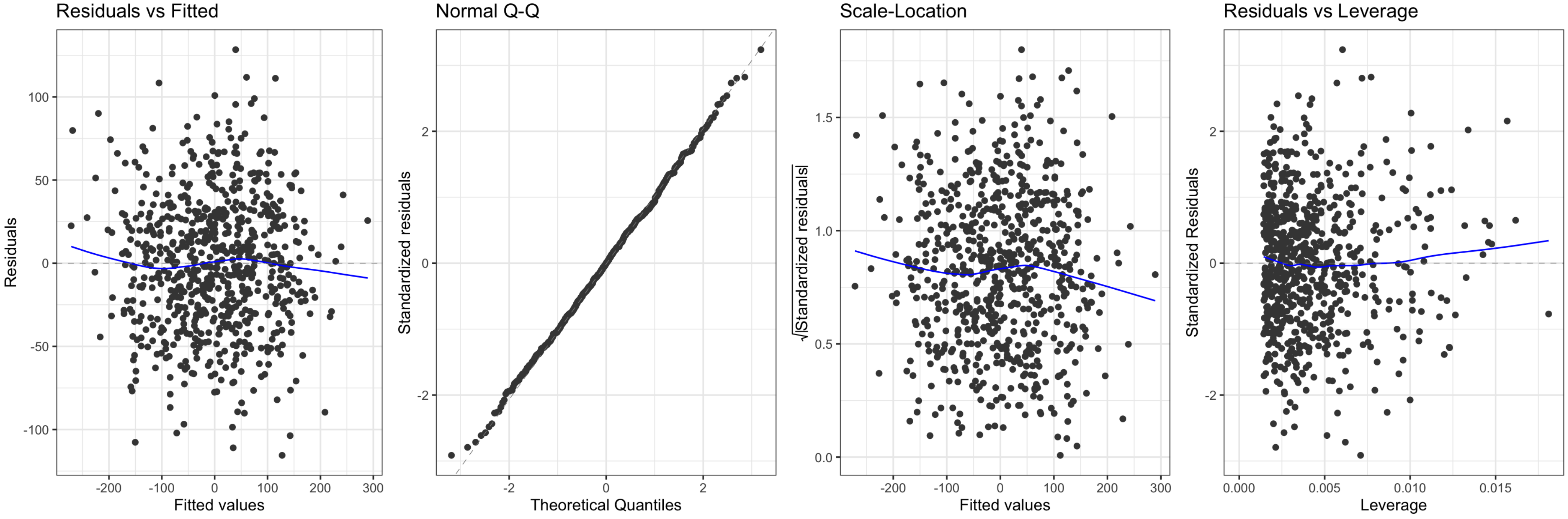
So we're estimating the mean μ of this conditional distribution, but what about σ^2?

Unbiased estimate $\hat{\sigma}^2 = \frac{RSS}{n-(p+1)}$, its square root is the `Residual standard error`

- **Degrees of freedom**: $n - (p+1)$, data supplies us with n "degrees of freedom" and we used up $p + 1$

Check the assumptions about normality with ggfortify

```r
library(ggfortify)
autoplot(multiple_lm, ncol = 4) + theme_bw()
```



- Standardized residuals = residuals / `sd(residuals)` (see also `.std.resid` from `augment`)