Modelos de Regressão

Douglas Cardoso

9/29/2021

R Markdown

Há uma certa relação entre as variáveis e se eu conseguir achar um modelo para essa relação, eu consigo fazer previsões. O objetivo é encontrar o melhor modelo que se ajusta aos dados.

Modelo linear

Encontrar o melhor valor de β_0 e β_1 que minimizem o erro. A difererença entre o valor que estimei e a observação é o **resíduo**, isto é, o erro. Uma das propriedades fundamentais é que o modelo assumo que os pontos estão distribuídos normalmente.

$$y = \beta_0 + \beta_1 x + \epsilon$$

- β_0: intercepto, o ponto que corta o eixo y
- β_1: inclinação da reta

Como os betas podem não ser suficiente para explicar o modelo, inclui-se o erro ϵ_i, que representa as informações que meu modelo não consegue captar. Assume-se que o erro tem média 0 e variância σ^2, ou seja, uma distribuição normal. Portanto, ao redor da reta do modelo há uma curva normal, ou seja, o valor que eu observo é uma realização dessa distribuição normal centrada nessa reta, com a probabilidade maior da observação cair em cima da reta, visto que o máximo da distribuição, a média, vai ser onde está o valor que foi predito.

Assumindo como variáveis aleatórias:

- Y = variável resposta
- X = preditor

Calculando a esperança em ambos os lados da equação:

$$E[Y] = E[\beta_0 + \beta_1 X + \epsilon]$$

Como a esperança é uma função linear:

$$E[Y] = E[\beta_0] + E[\beta_1 X] + E[\epsilon]$$

Como temos uma constante, podemos retirá-la

$$E[Y] = \beta_0 + \beta_1 E[X] + E[\epsilon]$$

E, na verdade, o último termo $E[\epsilon] = 0$, visto que assumi que a média é igual a zero.

$$\beta_0 = E[Y] - \beta_1 E[X]$$

Calculando a covariância entre X e Y:

$$Cov(X, Y) = Cov(X, \beta_0 + \beta_1 X) = Cov(X, \beta_0) + Cov(X, \beta_1)$$

Lembrando que:

$$Cov(X, Y) = E[XY] - E[X]E[Y]$$

Assim:

$$= \beta_0 Cov(X, 1) + \beta_1 Cov(X, X) = Cov(X, Y) = \beta_1 Var(X)$$

E temos:

$$\beta_1 = \frac{Cov(X, Y)}{V(X)}$$
$$\beta_0 = E[Y] - \beta_1 E[X]$$

Com essas equações, posso usar os estimadores de máxima verossimilhança para a média e para a covariância.

Para média:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} X_i,$$

e para a covariância:

$$S_{XY} = \frac{1}{n} \sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})$$

Ficamos com

$$\hat{\beta}_1 = \frac{S_{XY}}{S_X X}$$

e

$$\hat{\beta}_1 = \hat{Y} - \hat{\beta}_1 \overline{X}$$

Vamos entender a aplicação prática para entender a estimação desses coeficientes.

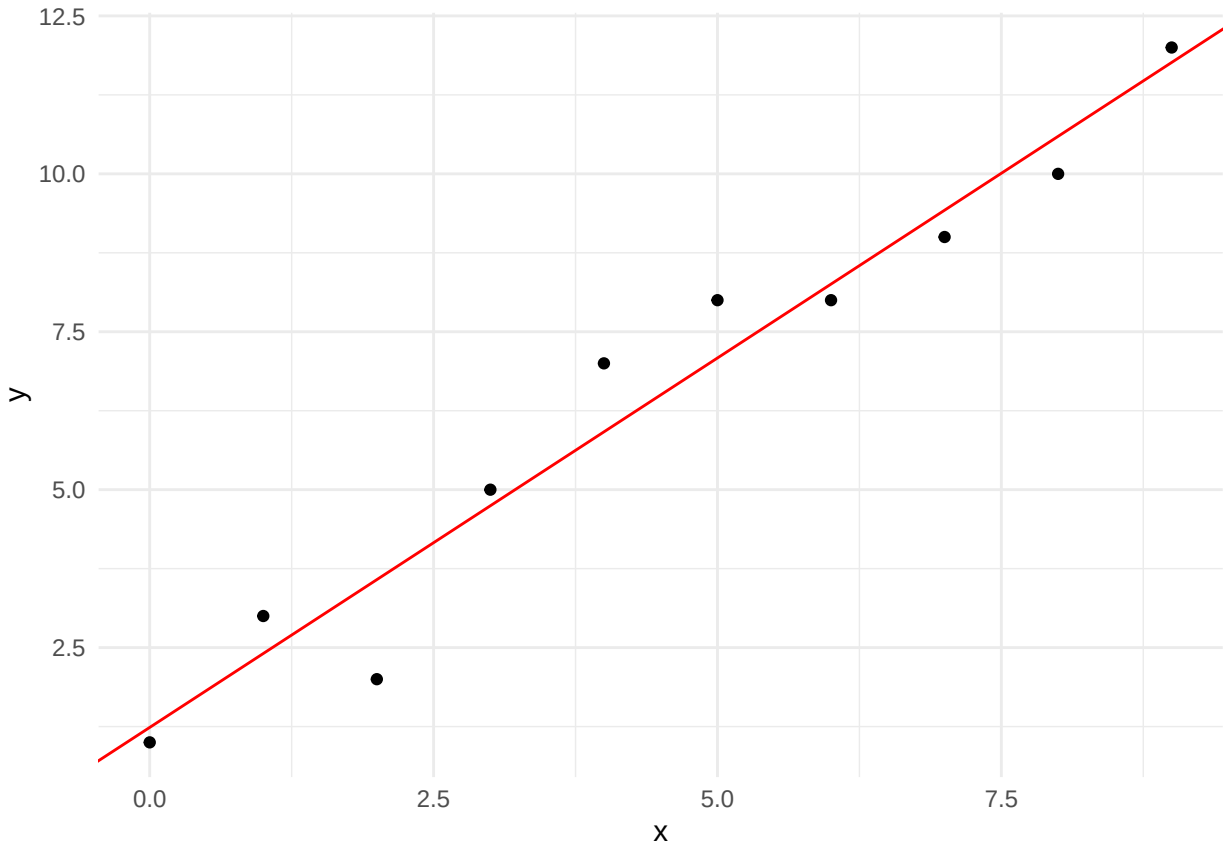
```
estimate_coef <- function(x, y){

  tibble::tibble(
    x = x,
    y = y,
    m_x = mean(x),
    m_y = mean(y),
    S_xy = (x - m_x) * (y - m_y),
    S_xx = (x - m_x)**2) |>
    dplyr::summarise(
      b_1 = sum(S_xy) / sum(S_xx),
      b_0 = m_y - b_1 * m_x) |>
    dplyr::distinct()
}
```

```r
x <- c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
y <- c(1, 3, 2, 5, 7, 8, 8, 9, 10, 12)

(da <- estimate_coef(x, y))
```

```
## # A tibble: 1 x 2
##     b_1   b_0
##   <dbl> <dbl>
## 1  1.17  1.24
```

Criação da própria reta

```r
tibble::tibble(
  x = x,
  y = y) |>
  ggplot2::ggplot(ggplot2::aes(x,y)) +
  ggplot2::geom_point() +
  ggplot2::geom_abline(intercept = da$b_0, slope = da$b_1, color = 'red') +
  ggplot2::theme_minimal()
```

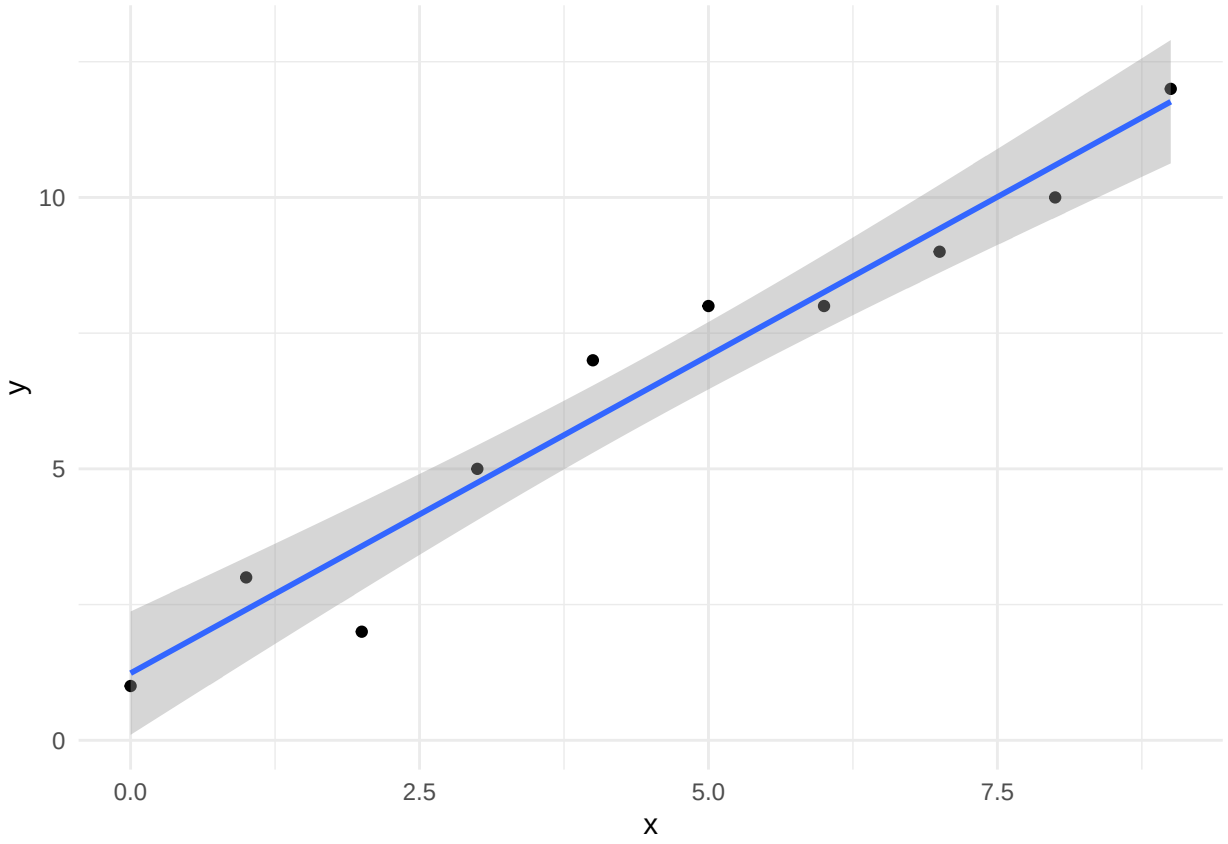


Reta com `geom_smooth`

```r
tibble::tibble(
  x = x,
  y = y) |>
  ggplot2::ggplot(ggplot2::aes(x,y)) +
```

```
  ggplot2::geom_point() +
  ggplot2::theme_minimal() +
  ggplot2::geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Agora, vamos verificar o quão bom é essa reta. Para isso, faremos a variância de Y.

$$V(Y) = V(\beta_0 + \beta_1 X + \epsilon) = \beta_1^2 V(X) + V(\epsilon)$$

$$\frac{V(Y) - V(\epsilon)}{V(Y)} = \frac{\beta_1^2 V(X)}{V(Y)}$$

$$R^2 = \frac{\beta_1^2 V(X)}{V(Y)}$$

$$R^2 = \frac{[Cov(X,Y)]^2}{V(X)V(Y)} = \frac{S_{XY}^2}{S_{XX}S_{YY}}$$

Aplicações em R:

```
R2 <- function(x,y){

  tibble::tibble(
    x = x,
    y = y,
```

```
    m_x = mean(x),
    m_y = mean(y),
    S_xy = (x - m_x) * (y - m_y),
    S_xx = (x - m_x)**2,
    S_yy = (y - m_y)**2) |>
    dplyr::summarise(R2 = sum(S_xy)**2 / (sum(S_xx) * sum(S_yy))) |>
    dplyr::mutate(cat = glue::glue("R2 = {round(R2, 4)}")) |>
    dplyr::pull(cat)

}

R2(x,y)
```

```
## R2 = 0.9525
```

Regressão linaer com `tidymodels`

```
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.3 --
```

```
## v broom        0.7.6      v recipes      0.1.16
## v dials        0.0.9      v rsample      0.1.0
## v dplyr        1.0.7      v tibble       3.1.3
## v ggplot2      3.3.5      v tidyr        1.1.3
## v infer        1.0.0      v tune         0.1.6
## v modeldata    0.1.1      v workflows    0.2.3
## v parsnip      0.1.7      v workflowsets 0.1.0
## v purrr        0.3.4      v yardstick    0.0.8
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
data <- tibble::tibble(x = x, y = y)

lm_model <- linear_reg() %>%
            set_engine('lm') %>%
            set_mode('regression')


lm_fit <- lm_model %>%
          fit(y ~ x, data = data)

summary(lm_fit$fit)
```

```
##
## Call:
## stats::lm(formula = y ~ x, data = data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5758 -0.3818  0.0000  0.5091  1.0849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.23636    0.49281   2.509   0.0364 *
## x            1.16970    0.09231  12.671 1.41e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8385 on 8 degrees of freedom
## Multiple R-squared:  0.9525, Adjusted R-squared:  0.9466
## F-statistic: 160.6 on 1 and 8 DF,  p-value: 1.415e-06
```

Nesse `summary` podemos analisar também já os testes de hipóteses, verificando se os coeficientes são relevantes para o modelo.

Modelos multivariados

Temos mais uma variável independentes. Estamos interessados na influência de várias variáveis. Em uma regressão linear múltipla não estamos mais interessados em ajustar uma reta, e sim um *plano*.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ ... \ + \beta_d X_d$$

Aqui também objetivamos minimizar o erro:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$$

Aplicação - predizendo o preço de casas

Feito com base no artigo: https://www.gmudatamining.com/lesson-10-r-tutorial.html

```
da <- readr::read_csv("../dados/BostonHousing.csv")
```

```
## Rows: 506 Columns: 14
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): crim, zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, b, ls...
```

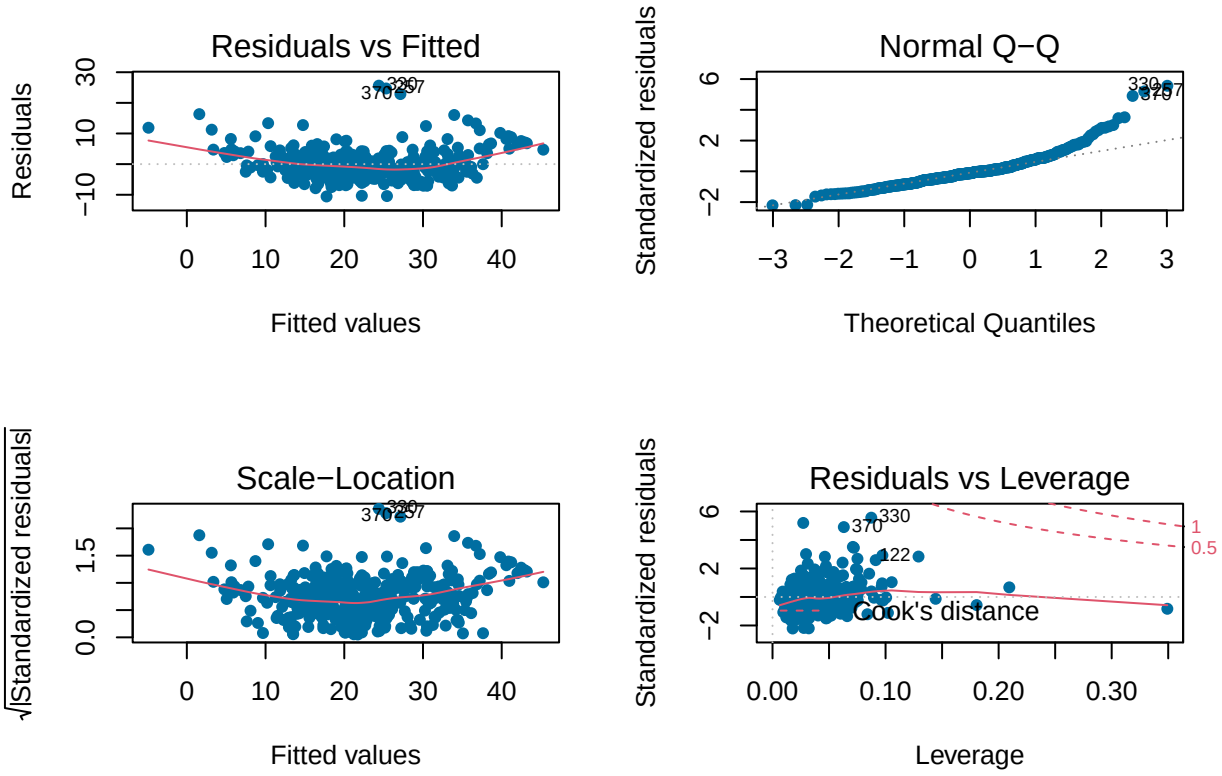
```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# train test split
da_split <- rsample::initial_split(da, prop = 0.75)
da_training <- da_split |> rsample::training()
da_test <- da_split |> rsample::testing()

# model specification
lm_model <- parsnip::linear_reg() |>
            parsnip::set_engine("lm") |>
            parsnip::set_mode("regression")
```

```r
# fitting the model
lm_fit <- lm_model |>
          parsnip::fit(medv ~ ., data = da_training)

summary(lm_fit$fit)
```

```
##
## Call:
## stats::lm(formula = medv ~ ., data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5519  -2.6820  -0.5741   1.8024  25.6074
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.550e+01  6.144e+00   5.777 1.63e-08 ***
## crim        -1.188e-01  3.540e-02  -3.355 0.000876 ***
## zn           4.786e-02  1.543e-02   3.101 0.002081 **
## indus        2.129e-02  7.029e-02   0.303 0.762174
## chas         3.406e+00  9.957e-01   3.420 0.000696 ***
## nox         -1.792e+01  4.506e+00  -3.977 8.41e-05 ***
## rm           3.829e+00  4.927e-01   7.771 8.02e-14 ***
## age          4.725e-04  1.535e-02   0.031 0.975452
## dis         -1.433e+00  2.308e-01  -6.208 1.46e-09 ***
## rad          3.174e-01  7.589e-02   4.183 3.61e-05 ***
## tax         -1.120e-02  4.214e-03  -2.658 0.008202 **
## ptratio     -8.983e-01  1.568e-01  -5.728 2.13e-08 ***
## b            8.092e-03  3.142e-03   2.575 0.010412 *
## lstat       -5.571e-01  5.909e-02  -9.428  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.817 on 365 degrees of freedom
## Multiple R-squared:  0.7431, Adjusted R-squared:  0.7339
## F-statistic: 81.21 on 13 and 365 DF,  p-value: < 2.2e-16
```

```r
par(mfrow=c(2,2)) # plot all 4 plots in one
plot(lm_fit$fit,
     pch = 16,
     col = '#006EA1')
```



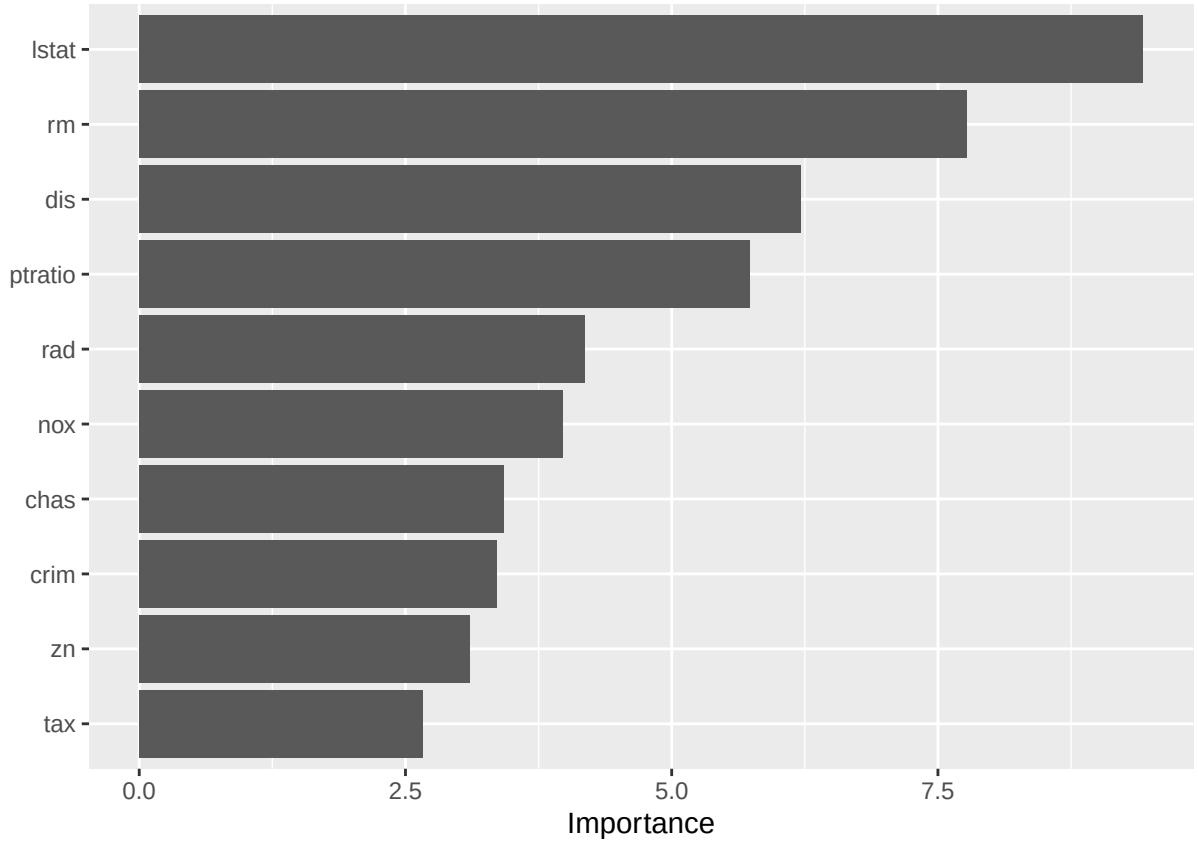

```
parsnip::tidy(lm_fit)
```

```
## # A tibble: 14 x 5
##    term          estimate std.error statistic  p.value
##    <chr>            <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)  35.5        6.14       5.78   1.63e- 8
##  2 crim         -0.119      0.0354    -3.36   8.76e- 4
##  3 zn            0.0479     0.0154     3.10   2.08e- 3
##  4 indus         0.0213     0.0703     0.303  7.62e- 1
##  5 chas          3.41       0.996      3.42   6.96e- 4
##  6 nox         -17.9        4.51      -3.98   8.41e- 5
##  7 rm            3.83       0.493      7.77   8.02e-14
##  8 age           0.000473   0.0153     0.0308 9.75e- 1
##  9 dis          -1.43       0.231     -6.21   1.46e- 9
## 10 rad           0.317      0.0759     4.18   3.61e- 5
## 11 tax          -0.0112     0.00421   -2.66   8.20e- 3
## 12 ptratio      -0.898      0.157     -5.73   2.13e- 8
## 13 b             0.00809    0.00314    2.58   1.04e- 2
## 14 lstat        -0.557      0.0591    -9.43   4.97e-19
```

```
# metrics on training data
parsnip::glance(lm_fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.743         0.734  4.82      81.2 3.97e-99    13 -1127. 2283. 2342.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
# variable importance
vip::vip(lm_fit)
```




```
# evaluating test set accuracy
predict(lm_fit, new_data = da_test)
```

```
## # A tibble: 127 x 1
##    .pred
##    <dbl>
##  1  27.9
##  2  18.5
##  3  21.2
##  4  19.6
##  5  19.3
##  6  20.6
##  7  16.7
##  8  17.5
##  9  15.5
## 10  13.4
## # ... with 117 more rows
```

```
# joining all

da_test_results <- predict(lm_fit, new_data = da_test) |>
                   dplyr::bind_cols(da_test)

da_test_results
```

```
## # A tibble: 127 x 15
##    .pred   crim    zn indus  chas   nox    rm   age   dis   rad   tax ptratio
```

```
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1  27.9 0.0690   0    2.18     0 0.458  7.15  54.2  6.06     3   222    18.7
##  2  18.5 0.170   12.5  7.87     0 0.524  6.00  85.9  6.59     5   311    15.2
##  3  21.2 0.117   12.5  7.87     0 0.524  6.01  82.9  6.23     5   311    15.2
##  4  19.6 0.630    0    8.14     0 0.538  5.95  61.8  4.71     4   307    21
##  5  19.3 0.638    0    8.14     0 0.538  6.10  84.5  4.46     4   307    21
##  6  20.6 1.05     0    8.14     0 0.538  5.94  29.3  4.50     4   307    21
##  7  16.7 0.784    0    8.14     0 0.538  5.99  81.7  4.26     4   307    21
##  8  17.5 0.852    0    8.14     0 0.538  5.96  89.2  4.01     4   307    21
##  9  15.5 1.23     0    8.14     0 0.538  6.14  91.7  3.98     4   307    21
## 10  13.4 0.988    0    8.14     0 0.538  5.81 100    4.10     4   307    21
## # ... with 117 more rows, and 3 more variables: b <dbl>, lstat <dbl>,
## #   medv <dbl>
```

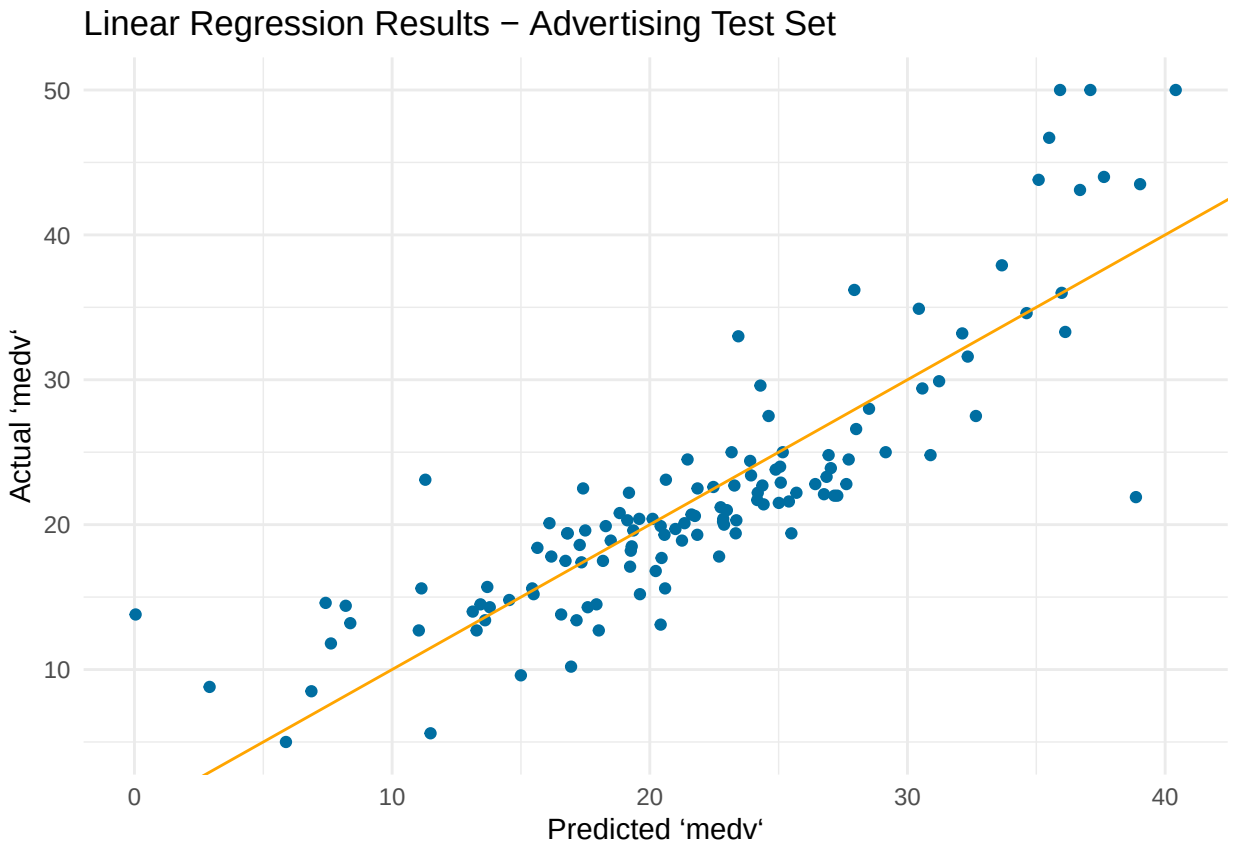
```
# RMSE on test set

yardstick::rmse(da_test_results,
                truth = medv,
                estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        4.59
```

```
# R2 on test set

yardstick::rsq(da_test_results,
               truth = medv,
               estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.727
```

```
# R2 PLOT

ggplot2::ggplot(data = da_test_results,
                ggplot2::aes(x = .pred, y = medv)) +
  ggplot2::geom_point(color = "#006EA1") +
  ggplot2::geom_abline(intercept = 0, slope = 1, color = "orange") +
  ggplot2::theme_minimal() +
  ggplot2::labs(title = "Linear Regression Results - Advertising Test Set",
                x = "Predicted `medv`",
                y = "Actual `medv`")
```


Linear Regression Results − Advertising Test Set
Actual ‘medv‘
Predicted ‘medv‘
50
40
30
20
10
0
10
20
30
40