7. Simple Linear Regression, $y = mx + b$

e.g. `cars` is a built-in data.frame: `cars`, `?cars`, `str(cars)`, `head(cars)`

- (Recall) `plot(x, y)` makes a (base graphics) scatterplot of data in the vectors `x` and `y`; e.g.

  ```
  plot(x=cars$speed, y=cars$dist)
  ```

- `cor(x, y)` gives the correlation of vectors `x` and `y`; e.g. `r = cor(x = cars$speed, y = cars$dist)`

  For data frame `x`, `A = cor(x)` gives a matrix `A` such that `A[i, j] == cor(x[ , i], x[ , j])`, the correlation of `A`'s i^{th} and j^{th} columns; e.g. `cor(mtcars[ , 1:3])`

- `lm(y ~ x, data)` calculates a linear regression model $y = mx + b$ from the `y` and `x` variables in the data.frame `data` (this uses the "`formula, data`" interface mentioned earlier); e.g.

  ```
  m = lm(dist ~ speed, data = cars) # "m" is for "model"
  str(m)
  summary(m) # summary
  anova(m) # ANOVA table
  ```

- `m$coefficients` is a vector containing y-intercept b and slope m:

  ```
  y.intercept = m$coefficients[1]
  slope = m$coefficients[2]
  ```

- `abline(a, b)` adds a line $y =$ `a` $+$ `b`x, and `abline(reg)` adds the line from model `reg`; e.g.

  ```
  abline(a = y.intercept, b = slope) # add regression line
  abline(reg = m) # same as previous line
  abline(a = mean(cars$dist), b = 0, lty = "dashed") # horizontal line through mean y
  ```

- `predict(model, newdata)` gives $\hat{y}$ from `model` evaluated at x (or at $x_1, \ldots, x_p$ in the multiple regression case) in data.frame `newdata`; e.g. Our model's x is `speed`; so put speeds for which we want predictions in a data.frame with a `speed` column:

  ```
  d = data.frame(speed = seq(from=5, to = 25, by = 5))
  y.hat = predict(m, newdata = d)
  # add (x, y) pairs to graph with plotting character 19, scaled by 3
  points(x=d$speed, y=y.hat, pch=19, cex=3)
  ```

- In the simple regression model $y_i = mx_i + b + \varepsilon_i$, errors ε_i are assumed to be random and independent, with $\varepsilon_i \sim N(0, \sigma)$. To check these assumptions, a *residual plot* of points $\{(\text{fitted value} = \hat{y}_i, \text{residual} = e_i = y_i - \hat{y}_i)\}$ should show no pattern (if errors are random and independent) or varying vertical spread (if errors have the same standard deviation σ); e.g.

  ```
  plot(m$fitted.values, m$residuals)
  abline(0, 0) # y = 0 + 0x; errors should have mean 0
  ```

- A *QQ plot* shows quantiles of a data distribution, like our residuals, on the y-axis against the same quantiles of a reference distribution, like $N(\mu =$`mean(residuals)`$, \sigma =$ `sd(residuals)`$)$. If the assumption of normal errors is met, these points should be close to a line. `qqline(x)` adds a line through the first and third quantile pairs. e.g.

```
x = rnorm(n=100); qqnorm(x);               qqline(x) # 100 random N(0, 1) points
w = rexp(100);    qqnorm(w, ylim=c(-1, 5)); qqline(w) # 100 random Exp(1)  points

qqnorm(m$residuals); qqline(m$residuals) # our "dist vs. speed" model
```

Or use `plot(m)` to see the residual and QQ plots, and two others, in one step:

```
layout(matrix(data=1:4, nrow=2, ncol=2, byrow=TRUE))
plot(m)
layout(matrix(data=1, nrow=1, ncol=1)) # reset graphics device
```

Multiple Linear Regression, $y = a_0 + a_1 x_1 + \cdots + a_p x_p$

e.g. `y ~ x1 + x2 + x3 + x1*x2` indicates that y depends linearly on x_1, x_2, x_3, and $x_1 \cdot x_2$, as in the multiple linear regression model, $y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_1 \cdot x_2$.

```
n = 100 # simulate n points, (y, x1, x2, x3), for a "sanity check" example
x1 = rnorm(n=n, mean=0, sd=1); x2 = rnorm(n); x3 = rnorm(n)
y = 3 + 4*x1 + 5*x2 + 6*x3 + 7*x1*x2
m = lm(y ~ x1 + x2 + x3 + x1*x2) # use lm() to discover coefficients from data
summary(m)

y = 3 + 4*x1 + 5*x2 + 6*x3 + 7*x1*x2 + rnorm(n) # add noise to make it harder
m2 = lm(y ~ x1 + x2 + x3 + x1*x2)
summary(m2)

m3 = lm(mpg ~ hp + wt + gear, data=mtcars) # real data from mtcars:
summary(m3)
anova(m3)
```

Inference on the coefficients is facilitated by `summary(model)`, which gives

- estimated coefficients $a_0, a_1, \cdots, a_p$
- estimated standard deviations of coeffiecients, $s_{a_0}, \cdots, s_{a_p}$
- the F statistic and P-value for $H_0 : a_1 = \cdots = a_p = 0$
- for each coefficient a_i, the t statistic and P-value for $H_0 : a_i = 0$

`confint(m, level = .95)` gives confidence intervals for the coefficients