Chapter 3: Linear Regression

Linear regression is a simple approach for supervised learning when the response is quantitative. Linear regression has a long history and we could actually spend most of this semester talking about it.

Although linear regression is not the newest, shiniest thing out there, it is still a highly used technique out in the real world. It is also useful for talking about more modern techniques that are **generalizations** of it.

We will review some key ideas underlying linear regression and discuss the least squares approach that is most commonly used to fit this model.

Linear regression can help us to answer the following questions about our `Advertising` data:

1 Simple Linear Regression

Simple Linear Regression is an approach for predictiong a quantitative response Y on the basis of a single predictor variable X.

It assumes:

Which leads to the following model:

For example, we may be interested in regressing `sales` onto `TV` by fitting the model

Once we have used training data to produce estimates $\hat{\beta}_0$ and $\hat{\beta}_1$, we can predict future sales on the basis of a particular TV advertising budget.

1.1 Estimating the Coefficients

In practice, β_0 and β_1 are **unknown**, so before we can predict $\hat{y}$, we must use our training data to estimate them.

Let $(x_1, y_1), \ldots, (x_n, y_n)$ represent n observation pairs, each of which consists of a measurement of X and Y.

Goal: Obtain coefficient estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ such that the linear model fits the available data well.

The most common approach involves minimizing the *least squares* criterion.

The least squares approach results in the following estimates:

$$\hat{\beta}_1 =$$
$$\hat{\beta}_0 =$$

We can get these estimates using the following commands in `R` and `tidymodels`:

```
library(tidymodels) ## load library

## load the data in
ads <- read_csv("../data/Advertising.csv", col_select = -1)

## fit the model
lm_spec <- linear_reg() |>
  set_mode("regression") |>
  set_engine("lm")

slr_fit <- lm_spec |>
  fit(sales ~ TV, data = ads)

slr_fit |>
  pluck("fit") |>
  summary()
```

```
##
## Call:
## stats::lm(formula = sales ~ TV, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3860 -1.9545 -0.1913  2.0671  7.2124
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.032594   0.457843   15.36   <2e-16 ***
## TV          0.047537   0.002691   17.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 198 degrees of freedom
## Multiple R-squared:  0.6119, Adjusted R-squared:  0.6099
## F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

1.2 Assessing Accuracy

Recall we assume the *true* relationship between X and Y takes the form

If f is to be approximated by a linear function, we can write this relationship as

and when we fit the model to the training data, we get the following estimate of the *population model*

But how close this this to the truth?

In general, σ^2 is not known, so we estimate it with the *residual standard error*, $RSE = \sqrt{RSS/(n-2)}$.

We can use these standard errors to compute confidence intervals and perform hypothesis tests.

Once we have decided that there is a significant linear relationship between X and Y that is captured by our model, it is natural to ask

> To what extent does the model fit the data?

The quality of the fit is usually measured by the *residual standard error* and the R^2 statistic.

RSE: Roughly speaking, the RSE is the average amount that the response will deviate from the true regression line. This is considered a measure of the *lack of fit* of the model to the data.

R^2: The RSE provides an absolute measure of lack of fit, but is measured in the units of Y. So, we don't know what a "good" RSE value is! R^2 gives the proportion of variation in Y explained by the model.

```r
slr_fit |>
  pluck("fit") |>
  summary()
```

```
##
## Call:
## stats::lm(formula = sales ~ TV, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3860 -1.9545 -0.1913  2.0671  7.2124
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.032594   0.457843   15.36   <2e-16 ***
## TV          0.047537   0.002691   17.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 198 degrees of freedom
## Multiple R-squared:  0.6119, Adjusted R-squared:  0.6099
## F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

2 Multiple Linear Regression

Simple linear regression is useful for predicting a response based on one predictor variable, but we often have **more than one** predictor.

How can we extend our approach to accommodate additional predictors?

We can give each predictor a separate slope coefficient in a single model.

We interpret β_j as the "average effect on Y of a one unit increase in X_j, *holding all other predictors fixed*".

In our `Advertising` example,

2.1 Estimating the Coefficients

As with the case of simple linear regression, the coefficients $\beta_0, \beta_1, \ldots, \beta_p$ are unknown and must be estimated. Given estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, we can make predictions using the formula

The parameters are again estimated using the same least squares approach that we saw in the context of simple linear regression.

```
# mlr_fit <- lm_spec |> fit(sales ~ TV + radio + newspaper, data = ads)
mlr_fit <- lm_spec |>
  fit(sales ~ ., data = ads)

mlr_fit |>
  pluck("fit") |>
  summary()
```

```
##
## Call:
## stats::lm(formula = sales ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## TV           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

2.2 Some Important Questions

When we perform multiple linear regression we are usually interested in answering a few important questions:

1.

2.

3.

4.

2.2.1 Is there a relationship between response and predictors?

We need to ask whether all of the regression coefficients are zero, which leads to the following hypothesis test.

$$H_0 :$$

$$H_a :$$

This hypothesis test is performed by computing the F-statistic

$$F =$$

2.2.2 Deciding on Important Variables

After we have computed the F-statistic and concluded that there is a relationship between predictor and response, it is natural to wonder

> Which predictors are related to the response?

We could look at the p-values on the individual coefficients, but if we have many variables this can lead to false discoveries.

Instead we could consider *variable selection*. We will revisit this in Ch. 6.

2.2.3 Model Fit

Two of the most common measures of model fit are the RSE and R^2. These quantities are computed and interpreted in the same way as for simple linear regression.

Be careful with using these alone, because R^2 will **always increase** as more variables are added to the model, even if it's just a small increase.

```r
# model with TV, radio, and newspaper
mlr_fit |> pluck("fit") |> summary()
```

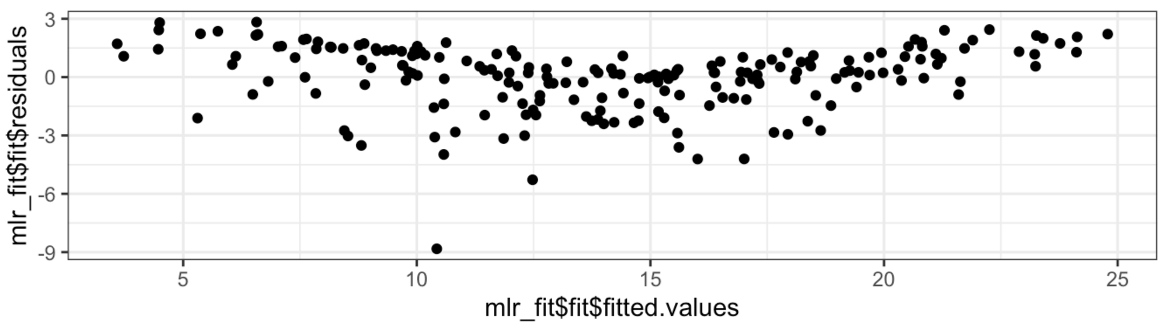
```
##
## Call:
## stats::lm(formula = sales ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8277 -0.8908  0.2418  1.1893  2.8292
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.938889   0.311908   9.422   <2e-16 ***
## TV           0.045765   0.001395  32.809   <2e-16 ***
## radio        0.188530   0.008611  21.893   <2e-16 ***
## newspaper   -0.001037   0.005871  -0.177     0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.686 on 196 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8956
## F-statistic: 570.3 on 3 and 196 DF,  p-value: < 2.2e-16
```

```
# model without newspaper
lm_spec |> fit(sales ~ TV + radio, data = ads) |>
  pluck("fit") |> summary()
```

```
##
## Call:
## stats::lm(formula = sales ~ TV + radio, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7977 -0.8752  0.2422  1.1708  2.8328
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.92110    0.29449   9.919   <2e-16 ***
## TV           0.04575    0.00139  32.909   <2e-16 ***
## radio        0.18799    0.00804  23.382   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.681 on 197 degrees of freedom
## Multiple R-squared:  0.8972, Adjusted R-squared:  0.8962
## F-statistic: 859.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

It may also be useful to plot residuals to get a sense of the model fit.

```
ggplot() +
  geom_point(aes(mlr_fit$fit$fitted.values, mlr_fit$fit$residuals))
```



3 Other Considerations

3.1 Categorical Predictors

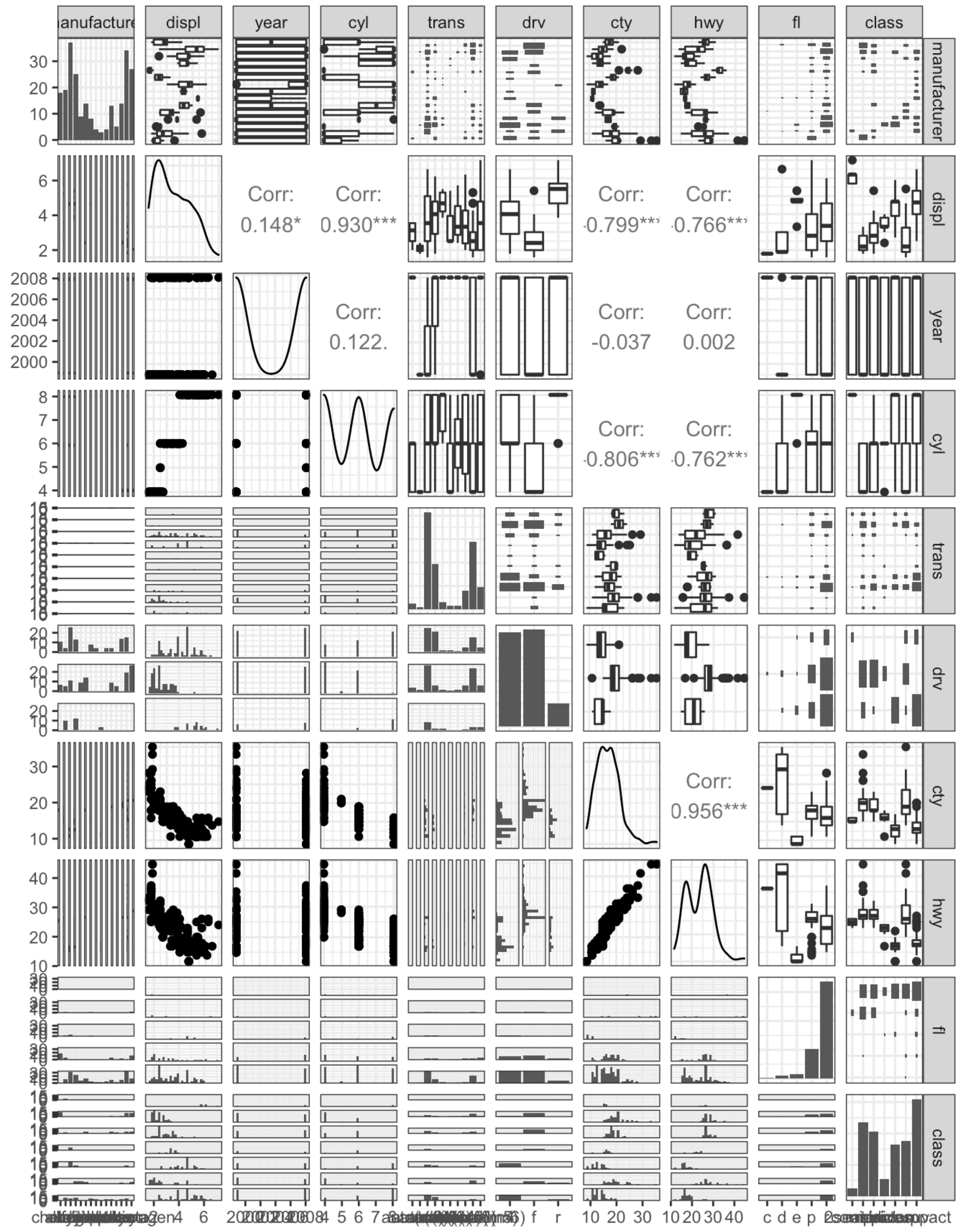
So far we have assumed all variables in our linear model are quantitiative.

For example, consider building a model to predict highway gas mileage from the `mpg` data set.

```
head(mpg)
```

```
## # A tibble: 6 × 11
##   manufacturer model displ  year   cyl trans      drv     cty   hwy
fl    class
##   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int>
<chr> <chr>
## 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29
p     compa…
## 2 audi         a4      1.8  1999     4 manual(m5) f        21    29
p     compa…
## 3 audi         a4      2    2008     4 manual(m6) f        20    31
p     compa…
## 4 audi         a4      2    2008     4 auto(av)   f        21    30
p     compa…
## 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26
p     compa…
## 6 audi         a4      2.8  1999     6 manual(m5) f        18    26
p     compa…
```

```
library(GGally)

mpg %>%
  select(-model) %>% # too many models
  ggpairs() # plot matrix
```


manufacturer
displ
year
cyl
trans
drv
cty
hwy
fl
class
Corr: 0.148*
Corr: 0.930***
Corr: -0.799***
Corr: -0.766***
Corr: 0.122.
Corr: -0.037
Corr: 0.002
Corr: 0.806***
Corr: 0.762***
Corr: 0.956***

To incorporate these categorical variables into the model, we will need to introduce $k - 1$ dummy variables, where $k =$ the number of levels in the variable, for each qualitative variable.

For example, for `drv`, we have 3 levels: `4`, `f`, and `r`.

```
lm_spec |>
  fit(hwy ~ displ + cty + drv, data = mpg) |>
  pluck("fit") |>
  summary()
```

```
## 
## Call:
## stats::lm(formula = hwy ~ displ + cty + drv, data = data)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -4.6499 -0.8764 -0.3001  0.9288  4.8632 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  3.42413    1.09313   3.132  0.00196 ** 
## displ       -0.20803    0.14439  -1.441  0.15100    
## cty          1.15717    0.04213  27.466  < 2e-16 ***
## drvf         2.15785    0.27348   7.890 1.23e-13 ***
## drvr         2.35970    0.37013   6.375 9.95e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.49 on 229 degrees of freedom
## Multiple R-squared:  0.9384, Adjusted R-squared:  0.9374 
## F-statistic: 872.7 on 4 and 229 DF,  p-value: < 2.2e-16
```

3.2 Extensions of the Model

The standard regression model provides interpretable results and works well in many problems. However it makes some very strong assumptions that may not always be reasonable.

Additive Assumption

The additive assumption assumes that the effect of each predictor on the response is not affected by the value of the other predictors. What if we think the effect should depend on the value of another predictor?

```
lm_spec |>
  fit(sales ~ TV + radio + TV*radio, data = ads) |>
  pluck("fit") |>
  summary()
```

```
##
## Call:
## stats::lm(formula = sales ~ TV + radio + TV * radio, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3366 -0.4028  0.1831  0.5948  1.5246
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.750e+00  2.479e-01  27.233   <2e-16 ***
## TV          1.910e-02  1.504e-03  12.699   <2e-16 ***
## radio       2.886e-02  8.905e-03   3.241   0.0014 **
## TV:radio    1.086e-03  5.242e-05  20.727   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9435 on 196 degrees of freedom
## Multiple R-squared:  0.9678, Adjusted R-squared:  0.9673
## F-statistic:  1963 on 3 and 196 DF,  p-value: < 2.2e-16
```

Alternatively:

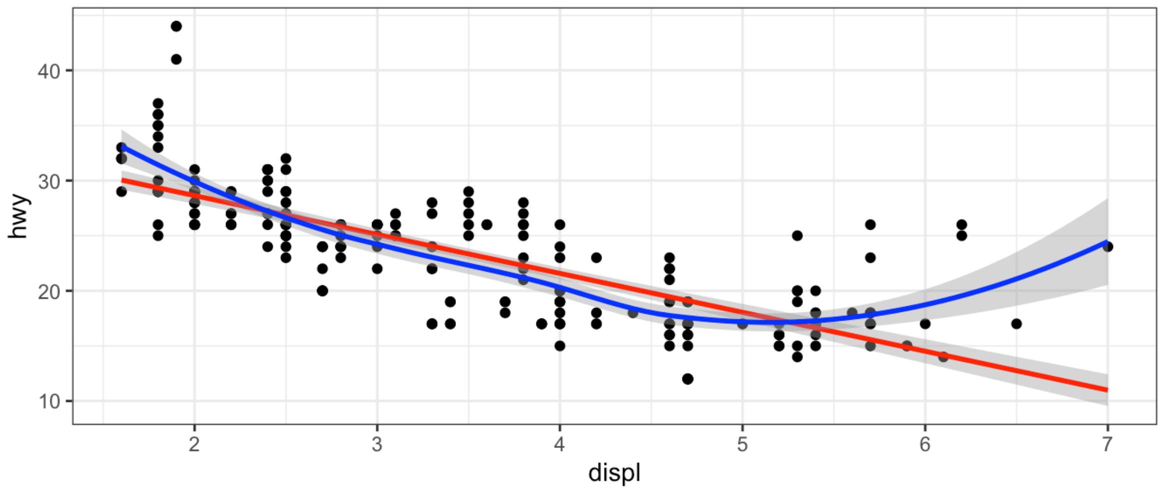
```
rec_spec_interact <- recipe(sales ~ TV + radio, data = ads) |>
  step_interact(~ TV:radio)

lm_wf_interact <- workflow() |>
  add_model(lm_spec) |>
  add_recipe(rec_spec_interact)

lm_wf_interact |> fit(ads)
```

```
## ══ Workflow [trained] ══════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: linear_reg()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────
## 1 Recipe Step
##
## • step_interact()
##
## ── Model ───────────────────────────────────────────────────────────────────────
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
## (Intercept)           TV        radio   TV_x_radio
##    6.750220     0.019101     0.028860     0.001086
```

Linearity Assumption

The linear regression model assumes a linear relationship between response and predictors. In some cases, the true relationship may be non-linear.

```r
ggplot(data = mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth(method = "lm", colour = "red") +
  geom_smooth(method = "loess", colour = "blue")
```



```r
lm_spec |>
  fit(hwy ~ displ + I(displ^2), data = mpg) |>
  pluck("fit") |> summary()
```

```
##
## Call:
## stats::lm(formula = hwy ~ displ + I(displ^2), data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6258 -2.1700 -0.7099  2.1768 13.1449
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  49.2450     1.8576  26.510  < 2e-16 ***
## displ       -11.7602     1.0729 -10.961  < 2e-16 ***
## I(displ^2)    1.0954     0.1409   7.773 2.51e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.423 on 231 degrees of freedom
## Multiple R-squared:  0.6725, Adjusted R-squared:  0.6696
## F-statistic: 237.1 on 2 and 231 DF,  p-value: < 2.2e-16
```

3.3 Potential Problems

1. Non-linearity of response-predictor relationships

2. Correlation of error terms

3. Non-constant variance of error terms

4. Outliers

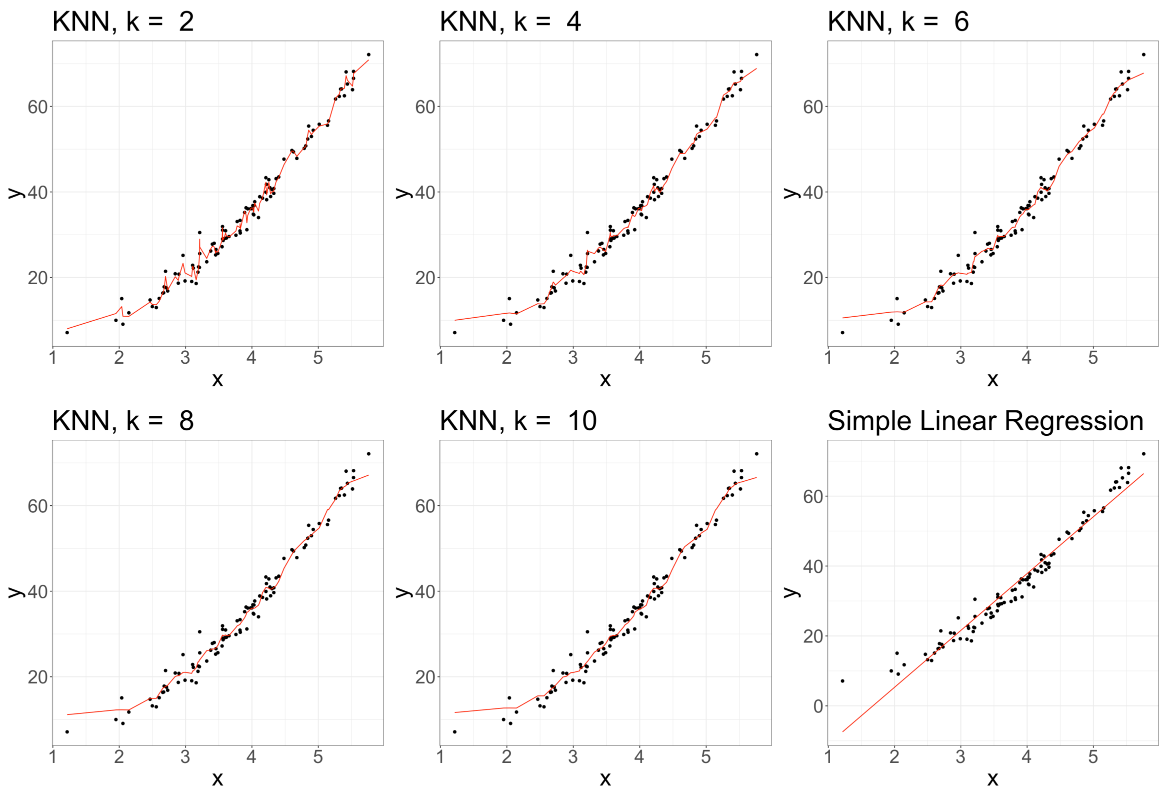
4 K-Nearest Neighbors

In Ch. 2 we discuss the differences between *parametric* and *nonparametric* methods. Linear regression is a parametric method because it assumes a linear functional form for $f(X)$.

A simple and well-known non-parametric method for regression is called K-nearest neighbors regression (KNN regression).

Given a value for K and a prediction point x_0, KNN regression first identifies the K training observations that are closest to x_0 ($\mathcal{N}_0$). It then estimates $f(x_0)$ using the average of all the training responses in $\mathcal{N}_0$,

```r
set.seed(445) #reproducibility

## generate data
x <- rnorm(100, 4, 1) # pick some x values
y <- 0.5 + x + 2*x^2 + rnorm(100, 0, 2) # true relationship
df <- data.frame(x = x, y = y) # data frame of training data

for (k in seq(2, 10, by = 2)) {
  nearest_neighbor(mode = "regression", neighbors = k) |>
    fit(y ~ x, data = df) |>
    augment(new_data = df) |>
    ggplot() +
    geom_point(aes(x, y)) +
    geom_line(aes(x, .pred), colour = "red") +
    ggtitle(paste("KNN, k = ", k)) +
    theme(text = element_text(size = 30)) -> p

  print(p)
}

lm_spec |>
  fit(y ~ x, df) |>
  augment(new_data = df) |>
  ggplot() +
    geom_point(aes(x, y)) +
    geom_line(aes(x, .pred), colour = "red") +
    ggtitle("Simple Linear Regression") +
    theme(text = element_text(size = 30)) # slr plot
```


KNN, k = 2
KNN, k = 4
KNN, k = 6
KNN, k = 8
KNN, k = 10
Simple Linear Regression
y
x