3 Other Considerations

3.1 Categorical Predictors

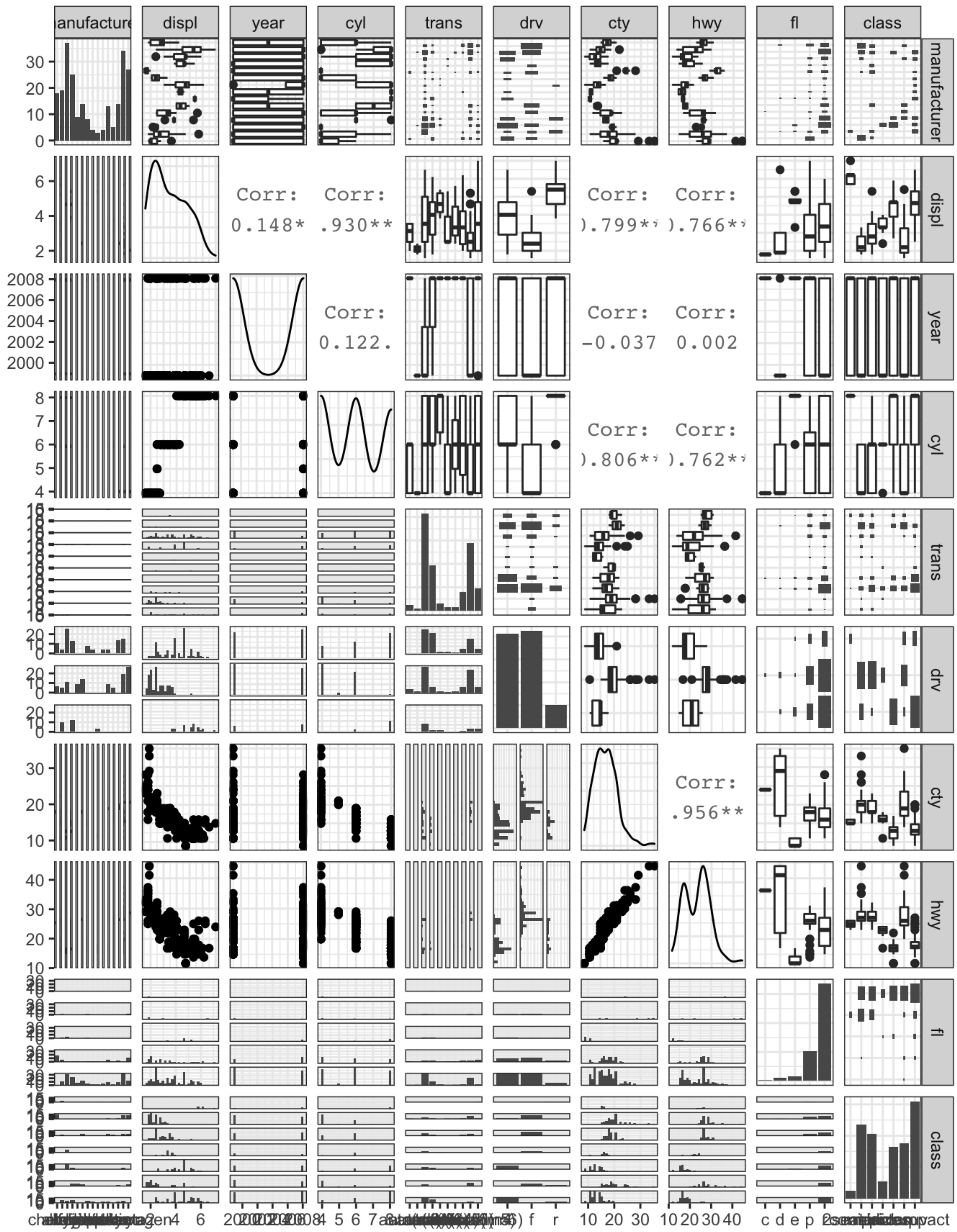
So far we have assumed all variables in our linear model are quantitiative.

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

```r
head(mpg)
```

```
## # A tibble: 6 x 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
```

```r
library(GGally)

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



anufacture
displ
year
cyl
trans
drv
cty
hwy
fl
class
manufacturer
Corr: 0.148*
Corr: .930**
Corr:).799**
Corr:).766**
Corr: 0.122.
Corr: -0.037
Corr: 0.002
Corr:).806**
Corr:).762**
Corr: .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`.

```r
lm(hwy ~ displ + cty + drv, data = mpg) %>%
  summary()
```

```
##
## Call:
## lm(formula = hwy ~ displ + cty + drv, data = mpg)
##
## 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?

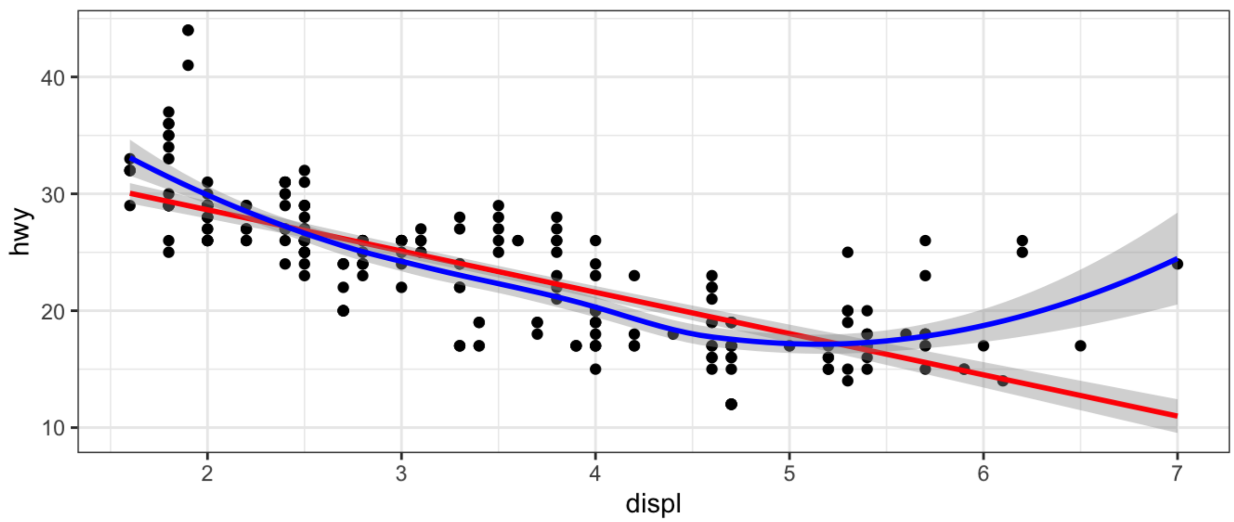
```
lm(sales ~ TV + radio + TV*radio, data = ads) %>%
  summary()
```

```
##
## Call:
## lm(formula = sales ~ TV + radio + TV * radio, data = ads)
##
## 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
```

Linearity Assumption

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

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



```r
lm(hwy ~ displ + I(displ^2), data = mpg) %>%
  summary()
```

```
##
## Call:
## lm(formula = hwy ~ displ + I(displ^2), data = mpg)
##
## 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

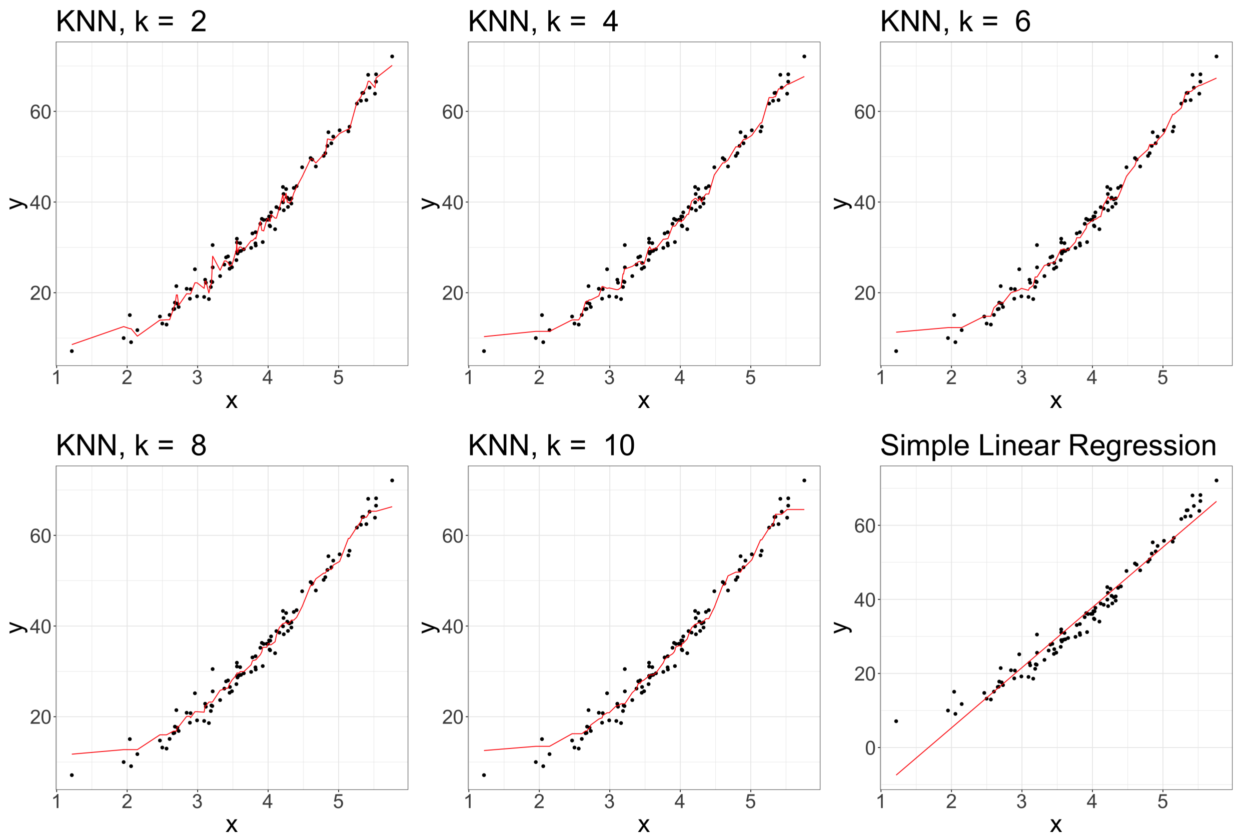
4 K-Nearest Neighbors

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
library(caret) # package for knn
set.seed(445) #reproducibility

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)) {
  knn_model <- knnreg(y ~ x, data = df, k = k) # fit knn model

  ggplot(df) +
    geom_point(aes(x, y)) +
    geom_line(aes(x, predict(knn_model, df)), colour = "red") +
    ggtitle(paste("KNN, k = ", k)) +
    theme(text = element_text(size = 30)) -> p

  print(p) # knn plots
}

ggplot(df) +
    geom_point(aes(x, y)) +
    geom_line(aes(x, lm(y ~ x, df)$fitted.values), 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
x
y