Homework 1

Tianyu Liu (301249861)

2023-09-25

1

I have read and acknowledge the SFU student academic integrity policy.

2 (Q2 from Problem Set 3)

(a)

i

There would be less bias, since with a reduced curvature the linear regression would have a better fit than if there is more curvature.

ii

The variance would be about the same, because Figure 6 is also using a linear fit with n = 10 so the variability would be about the same.

(b)

The variance gets smaller as we increase sample size, because the larger the sample, the more likely it is that errors average out above and below the true mean. The bias would not change with increasing sample size because when approximating a curve with a line the bias always exists.

3 (Application, from Problem Set 4)

1

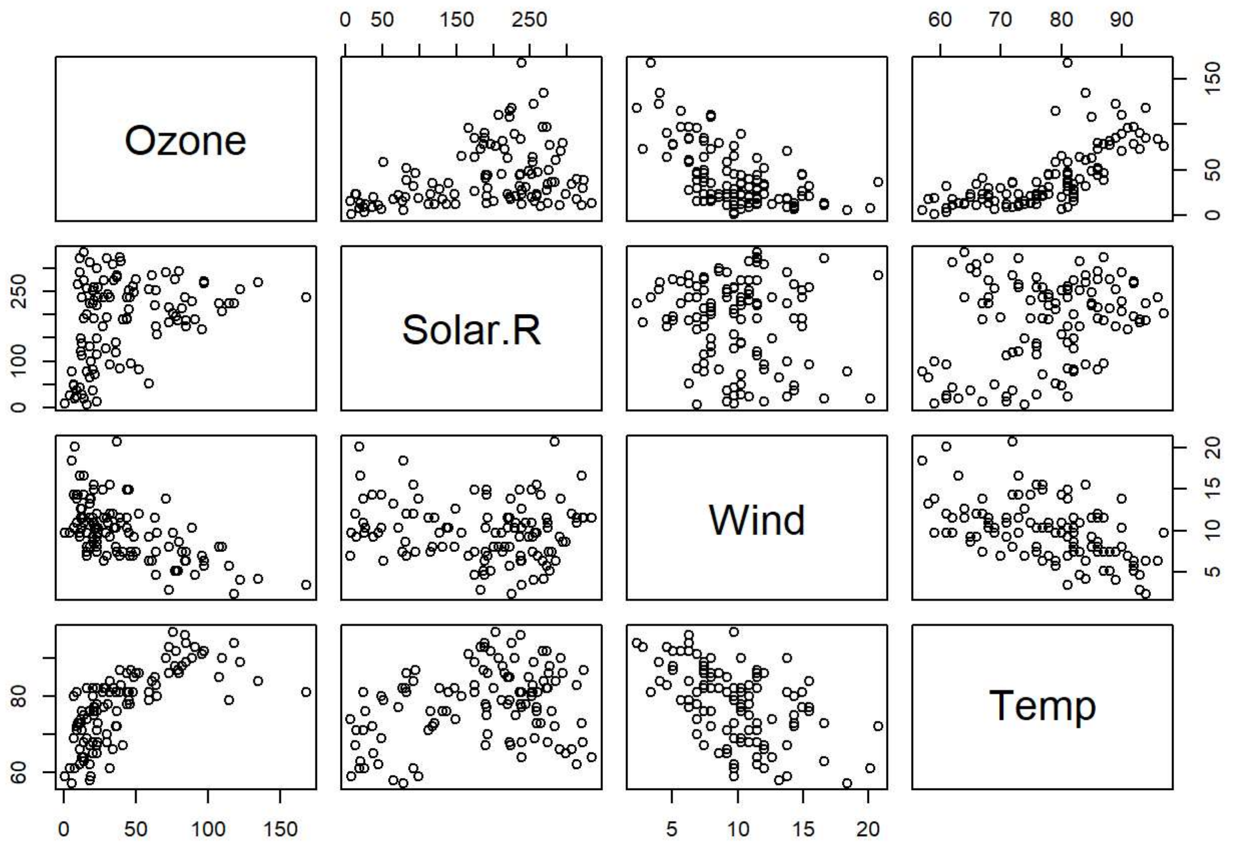
```
AQ <- na.omit(airquality[,1:4])
dim(AQ)
```

```
## [1] 111   4
```

```
head(AQ)
```

```
##   Ozone Solar.R Wind Temp
## 1    41     190  7.4   67
## 2    36     118  8.0   72
## 3    12     149 12.6   74
## 4    18     313 11.5   62
## 7    23     299  8.6   65
## 8    19      99 13.8   59
```

```
pairs(AQ)
```



2

```
set.seed(4099183)
n = nrow(AQ)
reorder = sample.int(n)
size.train <- floor(n * 0.75) ### Number of observations in our training
### set. Use floor() to round down
ind.train <- reorder[1:size.train] ### Indices of observations
### to put in training set
ind.valid <- reorder[(size.train + 1):n] ### Indices of observations
### to put in validation set
data.train <- AQ[ind.train,] ### Keep only observations in ind.train
data.valid <- AQ[ind.valid,] ### Keep only observations in ind.valid
print(ind.valid)
```

```
##  [1]  61  83  63  99  20  13  23  16  68  35  24  51 105   4  76  10  71  22  28
## [20]  79   6  37  70  74  18  25  59 108
```

These are the validation indices.

3

```
### Fit linear models to predict Ozone using each predictor
### individually, all predictors together, and all interactions with curvature
### Note: These models must be fit using data.train so that we can
### evaluate their MSPE on data.valid
fit.temp <- lm(Ozone ~ Temp, data = data.train)
fit.wind <- lm(Ozone ~ Wind, data = data.train)
fit.solar <- lm(Ozone ~ Solar.R, data = data.train)
fit.all <- lm(Ozone ~ Temp + Wind + Solar.R, data = data.train)
fit.intcurv <- lm(Ozone ~ Temp + Wind + Solar.R + I(Temp^2) + I(Wind^2) + I(Solar.R^2)
+ Temp*Wind + Temp*Solar.R + Wind*Solar.R, data = data.train)

### Get predictions on the validation set for each model using the
### predict() function.
pred.temp <- predict(fit.temp, data.valid)
pred.wind <- predict(fit.wind, data.valid)
pred.solar <- predict(fit.solar, data.valid)
pred.all <- predict(fit.all, data.valid)
pred.intcurv <- predict(fit.intcurv, data.valid)

### When we calculate validation set MSPEs for our models, we will
### end up repeating the same calculation 5 times. Let's make a
### function to do this for us.
get.MSPE <- function(Y, Y.hat) {
  residuals <- Y - Y.hat
  resid.sq <- residuals^2
  SSPE <- sum(resid.sq)
  MSPE <- SSPE / length(Y)
  return(MSPE)
}

### Use our get.MSPE() function to calculate the validation set MSPE
### of each model
Y.valid <- data.valid$Ozone
MSPE.temp <- get.MSPE(Y.valid, pred.temp)
MSPE.wind <- get.MSPE(Y.valid, pred.wind)
MSPE.solar <- get.MSPE(Y.valid, pred.solar)
MSPE.all <- get.MSPE(Y.valid, pred.all)
MSPE.intcurv <- get.MSPE(Y.valid, pred.intcurv)

### Let's compare these validation set MSPEs
print(MSPE.temp)
```

```
## [1] 586.7952
```

```
print(MSPE.wind)
```

```
## [1] 574.8944
```

```
print(MSPE.solar)
```

```
## [1] 934.5455
```

```
print(MSPE.all)
```

```
## [1] 387.3352
```

```
print(MSPE.intcurv)
```

```
## [1] 327.3287
```

As expected, the most complicated model wins the competition (The one with interactions and curvature).

4

```
### We need to divide the dataset into 5 folds.
### We can do this by randomly sampling the numbers from 1 to 10 and
### attaching these to our dataset as fold labels
M = 5
n.fold <- n / M # Number of observations in each fold
n.fold <- ceiling(n.fold) # Round up to make sure we get enough labels
# We can remove any excess later
ordered.ids <- rep(1:M, times = n.fold)
ordered.ids <- ordered.ids[1:n] # Remove excess label(s)
shuffle <- sample.int(n) # Randomly permute the numbers 1 to n
shuffled.ids <- ordered.ids[shuffle] # Use shuffle to permute
# the fold labels
data.CV <- AQ # Create a copy of our dataset
data.CV$fold <- shuffled.ids # Add a column to our new dataset containing
# the fold labels


### Next, let's actually do the cross validation. This will be easier
### with a for loop than with the replicate function. First, we will
### need to make an array to store the MSPEs
CV.MSPEs <- array(0, dim = c(M, 5))
colnames(CV.MSPEs) <- c("temp", "wind", "solar", "all", "intcurv")
### Set model names

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
for (i in 1:M) {
  ### Use fold i for validation and the rest for training
  data.train <- filter(data.CV, fold != i)
  data.valid <- filter(data.CV, fold == i)

  ### Remove fold from training and validation sets since it
  ### isn't a real predictor
  data.train <- select(data.train, -fold)
  data.valid <- select(data.valid, -fold)

  ### Fit linear models to predict Ozone using each predictor
  ### individually, all predictors together, and all interactions
  ### Note: These models must be fit using data.train so that we can
  ### evaluate their MSPE on data.valid
  fit.temp <- lm(Ozone ~ Temp, data = data.train)
  fit.wind <- lm(Ozone ~ Wind, data = data.train)
  fit.solar <- lm(Ozone ~ Solar.R, data = data.train)
  fit.all <- lm(Ozone ~ Temp + Wind + Solar.R, data = data.train)
  fit.intcurv <- lm(Ozone ~ Temp + Wind + Solar.R + I(Temp^2) + I(Wind^2) + I(Solar.R^2)
  + Temp*Wind + Temp*Solar.R + Wind*Solar.R, data = data.train)


  ### Get predictions on the validation set for each model using the
  ### predict() function.
  pred.temp <- predict(fit.temp, data.valid)
  pred.wind <- predict(fit.wind, data.valid)
  pred.solar <- predict(fit.solar, data.valid)
  pred.all <- predict(fit.all, data.valid)
  pred.intcurv <- predict(fit.intcurv, data.valid)

  ### Use our get.MSPE() function to calculate the validation set MSPE
  ### of each model
  Y.valid <- data.valid$Ozone
  MSPE.temp <- get.MSPE(Y.valid, pred.temp)
  MSPE.wind <- get.MSPE(Y.valid, pred.wind)
  MSPE.solar <- get.MSPE(Y.valid, pred.solar)
  MSPE.all <- get.MSPE(Y.valid, pred.all)
  MSPE.intcurv <- get.MSPE(Y.valid, pred.intcurv)

  ### Store MSPEs
  CV.MSPEs[i, 1] <- MSPE.temp
  CV.MSPEs[i, 2] <- MSPE.wind
  CV.MSPEs[i, 3] <- MSPE.solar
  CV.MSPEs[i, 4] <- MSPE.all
  CV.MSPEs[i, 5] <- MSPE.intcurv
}

MSPE.mean <- apply(X = CV.MSPEs, MARGIN = 2 , FUN = mean)
MSPE.mean #means
```

```
##     temp     wind    solar      all  intcurv
## 570.7978 703.3529 984.2164 462.2584 365.9409
```

```
MSPE.sd <- apply(X = CV.MSPEs, MARGIN = 2 , FUN = sd)
MSPE.CIl <- MSPE.mean - qt (p = .975 , df = M-1) * MSPE.sd/sqrt(M)
MSPE.CIu <- MSPE.mean + qt (p = .975 , df = M-1) * MSPE.sd/sqrt(M)
round(cbind(MSPE.CIl,MSPE.CIu), 2) #confidence intervals
```

```
##         MSPE.CIl MSPE.CIu
## temp      283.95   857.65
## wind      587.57   819.13
## solar     807.92  1160.52
## all       263.05   661.47
## intcurv   229.59   502.29
```

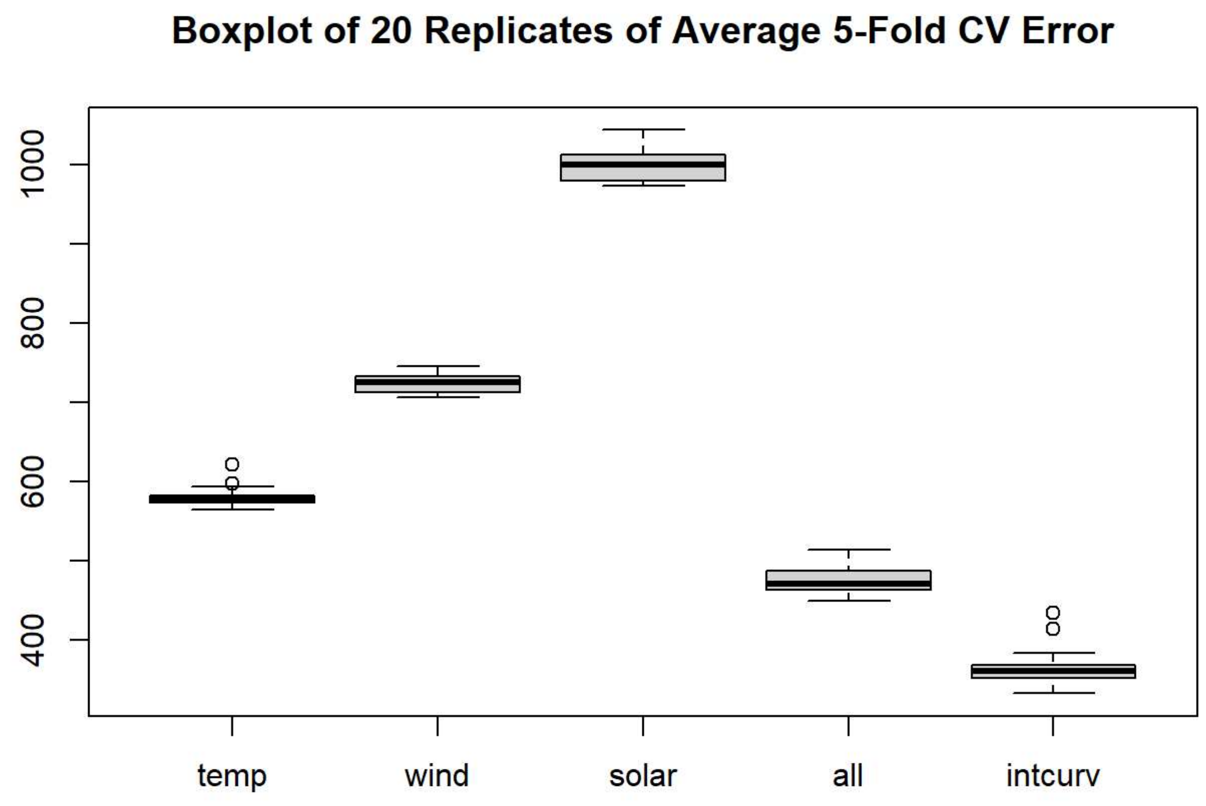
The model involving solar has a very high MSPE, while the model involving temp and all predictors have high variance. Taking a balance between variability and the actual MSPE, wind seems like a much better model. Only the model for interaction and curvature has low variance and low MSPE. It’s clearly the best model but that’s intuitive since it considers all the factors.

5

```r
n.rep <- 20 # Number of times to repeat CV

### First, we need a container to store the average CV errors
ave.CV.MSPEs <- array(0, dim = c(n.rep, 5))
colnames(ave.CV.MSPEs) <- colnames(CV.MSPEs)

### We will put the entire CV section from above inside another
### for loop. This will repeat the entire CV process
for (j in 1:n.rep) {
  ordered.ids <- rep(1:M, times = n.fold)
  ordered.ids <- ordered.ids[1:n]
  shuffle <- sample.int(n)
  shuffled.ids <- ordered.ids[shuffle]

  data.CV <- AQ
  data.CV$fold <- shuffled.ids

  CV.MSPEs <- array(0, dim = c(M, 5))
  colnames(CV.MSPEs) <- c("temp", "wind", "solar", "all", "intcurv")

  for (i in 1:M) {
    data.train <- filter(data.CV, fold != i)
    data.valid <- filter(data.CV, fold == i)

    ### In tutorial, I was getting an error because I wrote -folds
    ### instead of -fold. Whoops!
    data.train <- select(data.train, -fold)
    data.valid <- select(data.valid, -fold)

    fit.temp <- lm(Ozone ~ Temp, data = data.train)
    fit.wind <- lm(Ozone ~ Wind, data = data.train)
    fit.solar <- lm(Ozone ~ Solar.R, data = data.train)
    fit.all <- lm(Ozone ~ Temp + Wind + Solar.R, data = data.train)
    fit.intcurv <- lm(Ozone ~ Temp + Wind + Solar.R + I(Temp^2) + I(Wind^2) + I(Solar.R^2)
    + Temp*Wind + Temp*Solar.R + Wind*Solar.R, data = data.train)

    pred.temp <- predict(fit.temp, data.valid)
    pred.wind <- predict(fit.wind, data.valid)
    pred.solar <- predict(fit.solar, data.valid)
    pred.all <- predict(fit.all, data.valid)
    pred.intcurv <- predict(fit.intcurv, data.valid)

    Y.valid <- data.valid$Ozone
    MSPE.temp <- get.MSPE(Y.valid, pred.temp)
    MSPE.wind <- get.MSPE(Y.valid, pred.wind)
    MSPE.solar <- get.MSPE(Y.valid, pred.solar)
    MSPE.all <- get.MSPE(Y.valid, pred.all)
    MSPE.intcurv <- get.MSPE(Y.valid, pred.intcurv)
    CV.MSPEs[i, 1] <- MSPE.temp
    CV.MSPEs[i, 2] <- MSPE.wind
    CV.MSPEs[i, 3] <- MSPE.solar
    CV.MSPEs[i, 4] <- MSPE.all
    CV.MSPEs[i, 5] <- MSPE.intcurv
  }

  ### We now have MSPEs for each fold of one iteration of CV. Let's
  ### get the average error across these folds (think of each fold
  ### as a data split), and store the result in ave.CV.MSPEs
  this.ave.MSPEs <- apply(CV.MSPEs, 2, mean)
  ave.CV.MSPEs[j, ] <- this.ave.MSPEs # We are replacing a whole
  # row at once
}
```

(a)

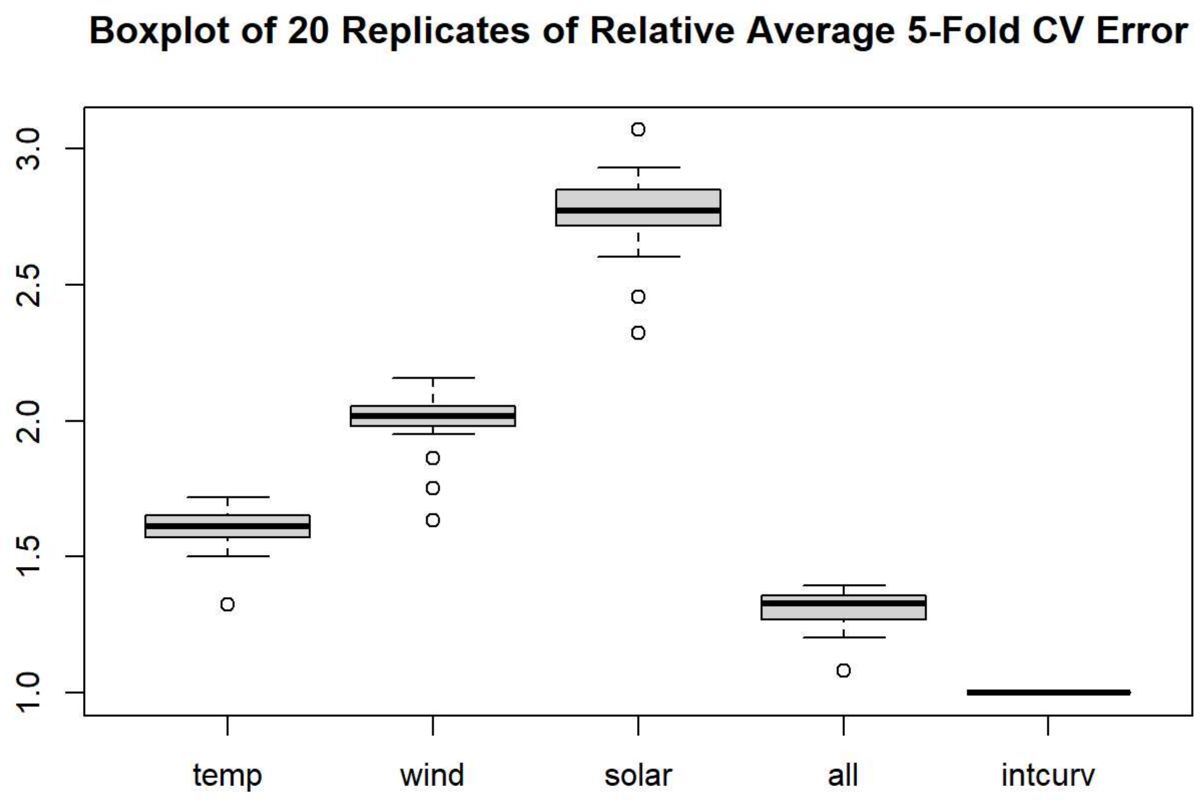
```
boxplot(ave.CV.MSPEs,
  main = "Boxplot of 20 Replicates of Average 5-Fold CV Error"
)
```



The models for all variables and the interactions are still good, but the model for temp is also decent, with low variance compared to other models. The model for wind and solar are much worse.

(b)

```
rel.ave.CV.MSPEs <- apply(ave.CV.MSPEs, 1, function(W) {
  best <- min(W)
  return(W / best)
})
rel.ave.CV.MSPEs <- t(rel.ave.CV.MSPEs)

boxplot(rel.ave.CV.MSPEs,
  main = "Boxplot of 20 Replicates of Relative Average 5-Fold CV Error"
)
```



Relative to the best model, the model for all predictors has the lowest MSPE and would be the next best model. The model for temp is also a good model with relatively low MSPE and low variability.

6

The interactive model is not too practical as it has too many predictors and can lead to high variance when predicting the true model. Considering the MSPE, variance, and practicality, I would choose the all predictors model, or if I'm restricted to only using 1 predictor then I would pick the model with temperature.

4 (Problem Set 5B Categorical Explanatories)

```
ins = read.csv("Insurance.csv", header = TRUE)
ins <- ins[ins$claims>0,]
ins$zone <- as.factor(ins$zone)
ins$make <- as.factor(ins$make)
levels(ins$zone)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7"
```

```
levels(ins$make)
```

```
## [1] "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

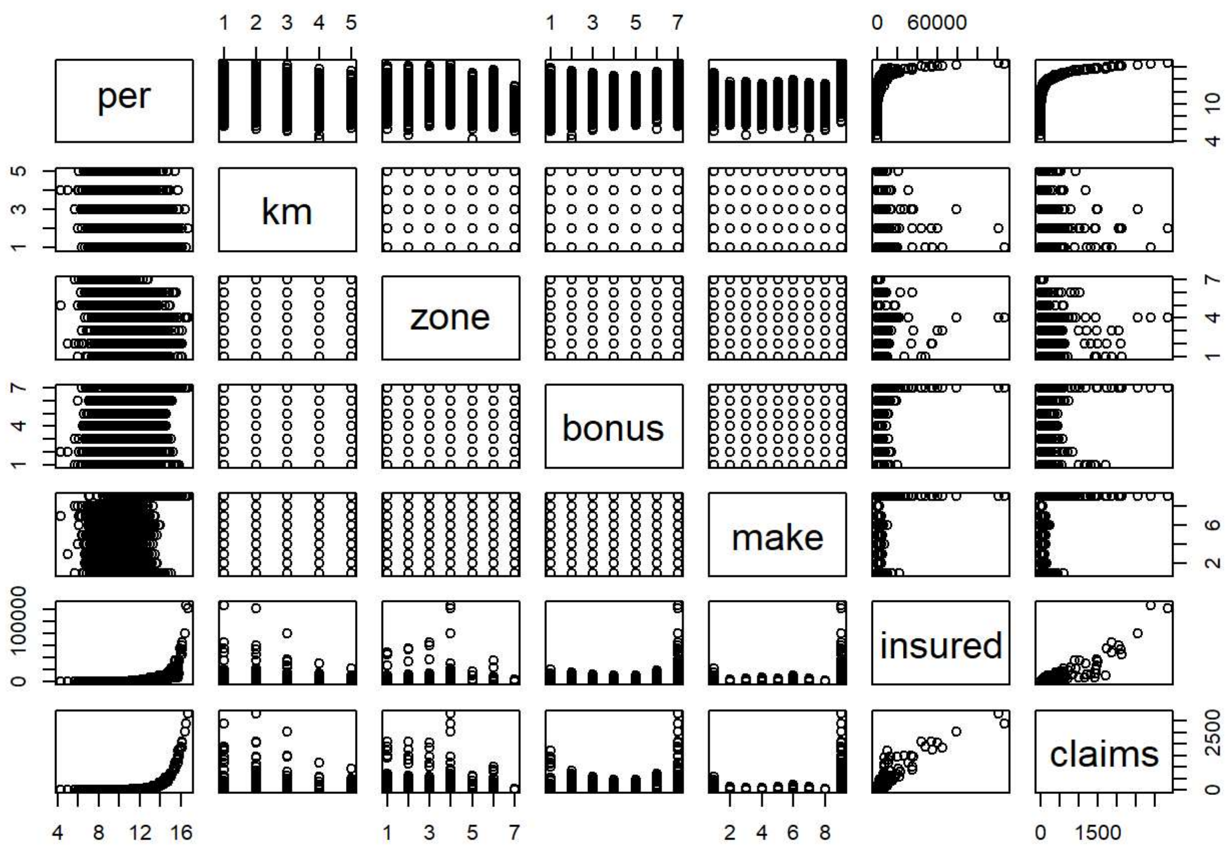
```
dim(ins)
```

```
## [1] 1797    7
```

```
head(ins)
```

```
##         per km zone bonus make insured claims
## 1 11.649460  2    3     2    6  669.34     42
## 2  9.615939  3    3     1    2   55.70      6
## 4  8.028129  2    7     7    5   81.78      3
## 5 11.222640  1    4     5    5  200.68     14
## 7 11.290310  3    5     2    6   64.73      6
## 8 10.163730  2    2     5    7  176.70     10
```

```
pairs(ins)
```



1

```
ins.fit = lm(per ~ . , data = ins)
summary(ins.fit)
```

```
##
## Call:
## lm(formula = per ~ ., data = ins)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0994 -0.7170  0.0734  0.8393  3.7574
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.186e+01  1.321e-01  89.770  < 2e-16 ***
## km          -3.434e-01  2.064e-02 -16.641  < 2e-16 ***
## zone2       -1.376e-01  9.717e-02  -1.416    0.157
## zone3       -2.143e-02  9.753e-02  -0.220    0.826
## zone4        4.317e-01  9.692e-02   4.454 8.95e-06 ***
## zone5       -1.042e+00  1.043e-01  -9.983  < 2e-16 ***
## zone6       -4.440e-01  1.009e-01  -4.401 1.14e-05 ***
## zone7       -2.862e+00  1.378e-01 -20.767  < 2e-16 ***
## bonus        2.301e-01  1.405e-02  16.381  < 2e-16 ***
## make2       -1.403e+00  1.140e-01 -12.314  < 2e-16 ***
## make3       -1.710e+00  1.189e-01 -14.382  < 2e-16 ***
## make4       -1.834e+00  1.240e-01 -14.789  < 2e-16 ***
## make5       -1.317e+00  1.138e-01 -11.568  < 2e-16 ***
## make6       -8.253e-01  1.129e-01  -7.312 3.95e-13 ***
## make7       -1.716e+00  1.153e-01 -14.878  < 2e-16 ***
## make8       -2.070e+00  1.199e-01 -17.260  < 2e-16 ***
## make9        1.459e+00  1.209e-01  12.071  < 2e-16 ***
## insured     -5.724e-05  1.151e-05  -4.975 7.15e-07 ***
## claims       3.029e-03  3.519e-04   8.608  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.179 on 1778 degrees of freedom
## Multiple R-squared:  0.6477, Adjusted R-squared:  0.6442
## F-statistic: 181.6 on 18 and 1778 DF,  p-value: < 2.2e-16
```

i

There are 6 variables, but R is treating each level of categorical variable differently, thus lm produced the estimate for 19 parameters: 1 for the intercept, 1 for each of the 4 numerical variables (km, bonus, insured, and claims), 6 for zone (1 less than the 7 levels - baseline dropped), and 8 for make (same logic - dropped 1 level)

ii

This is the default baseline that R uses, so the intercept is just 11.186

iii

The intercept would be the Beta-0 + the coefficient for zone7 + the coefficient for make9, which is

```
11.186 - 2.862 + 1.459
```

```
## [1] 9.783
```