Hwk2_nikhila_ravi

Nikhila Ravi

Feb 8th 2017

‘‘‘

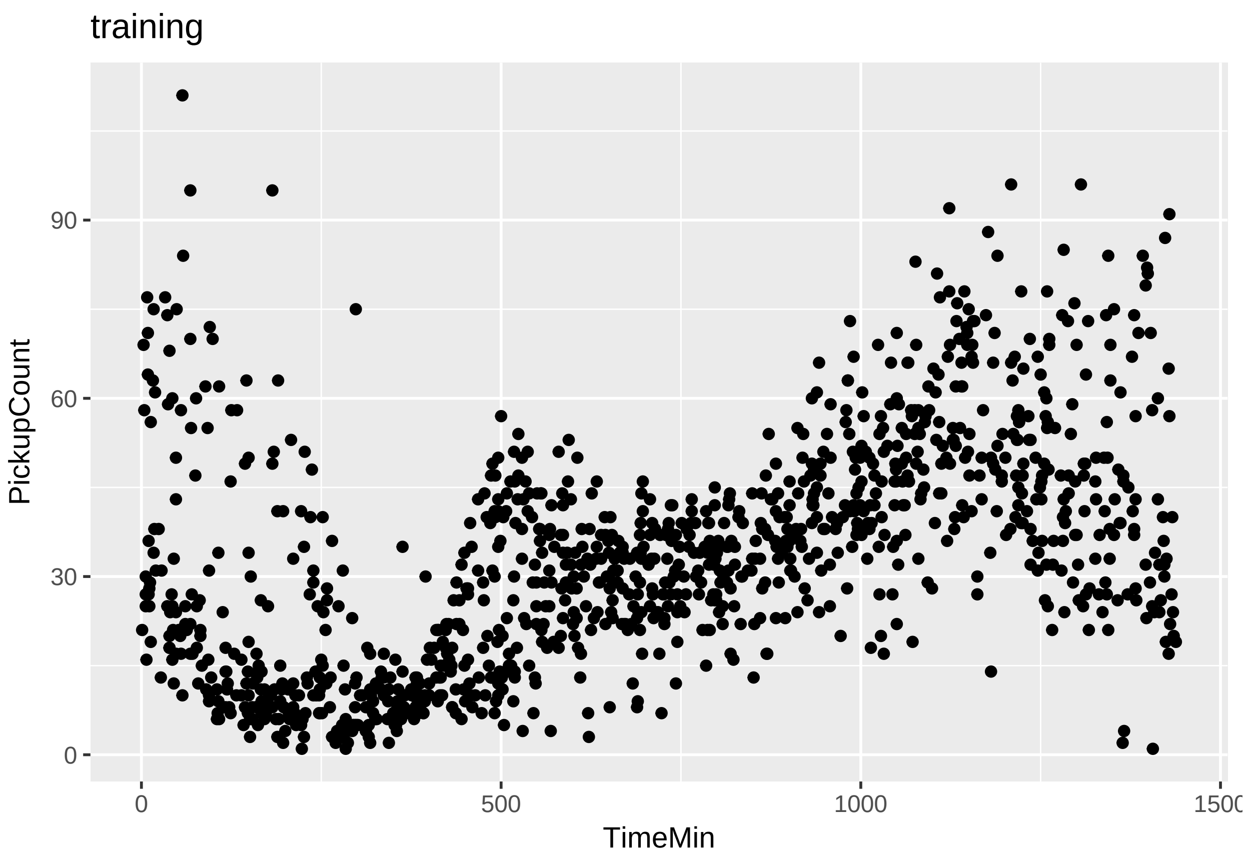
Question 1

Load the data:

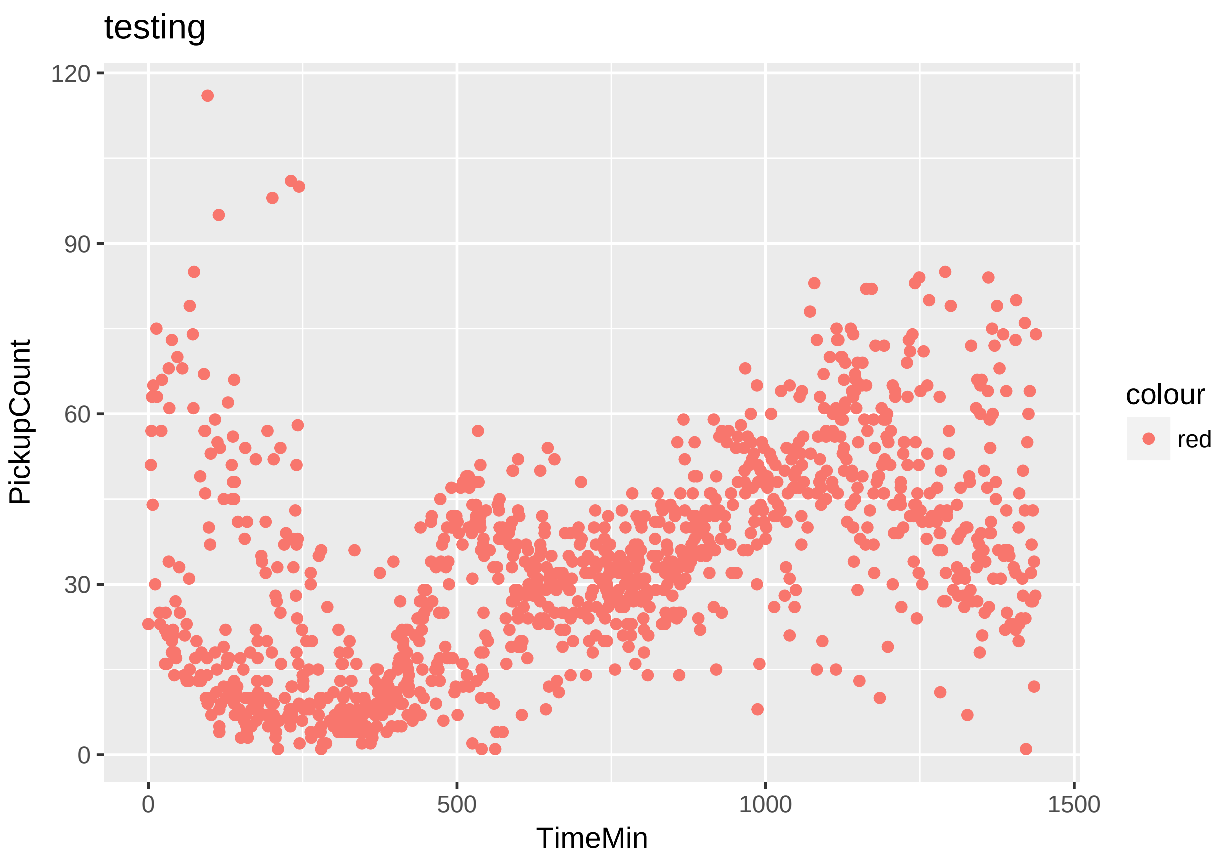
```
taxi_train <- read.table("dataset_1_train.txt", sep = ",", header = TRUE)
taxi_test <- read.table("dataset_1_test.txt", sep = ",", header = TRUE)
```

Visualise the data:

```
library(ggplot2)
ggplot(taxi_train, aes(x = TimeMin, y = PickupCount)) + geom_point() +
    ggtitle("training")
```



```
ggplot(taxi_test, aes(x = TimeMin, y = PickupCount, color = "red")) +
    geom_point() + ggtitle("testing")
```



As expected during the early morning (250 min) there are low number of pickups, with the number of pickups increasing during rushour in the morning (8:30am (500mins)) and peaking in the evening.

Question 1a

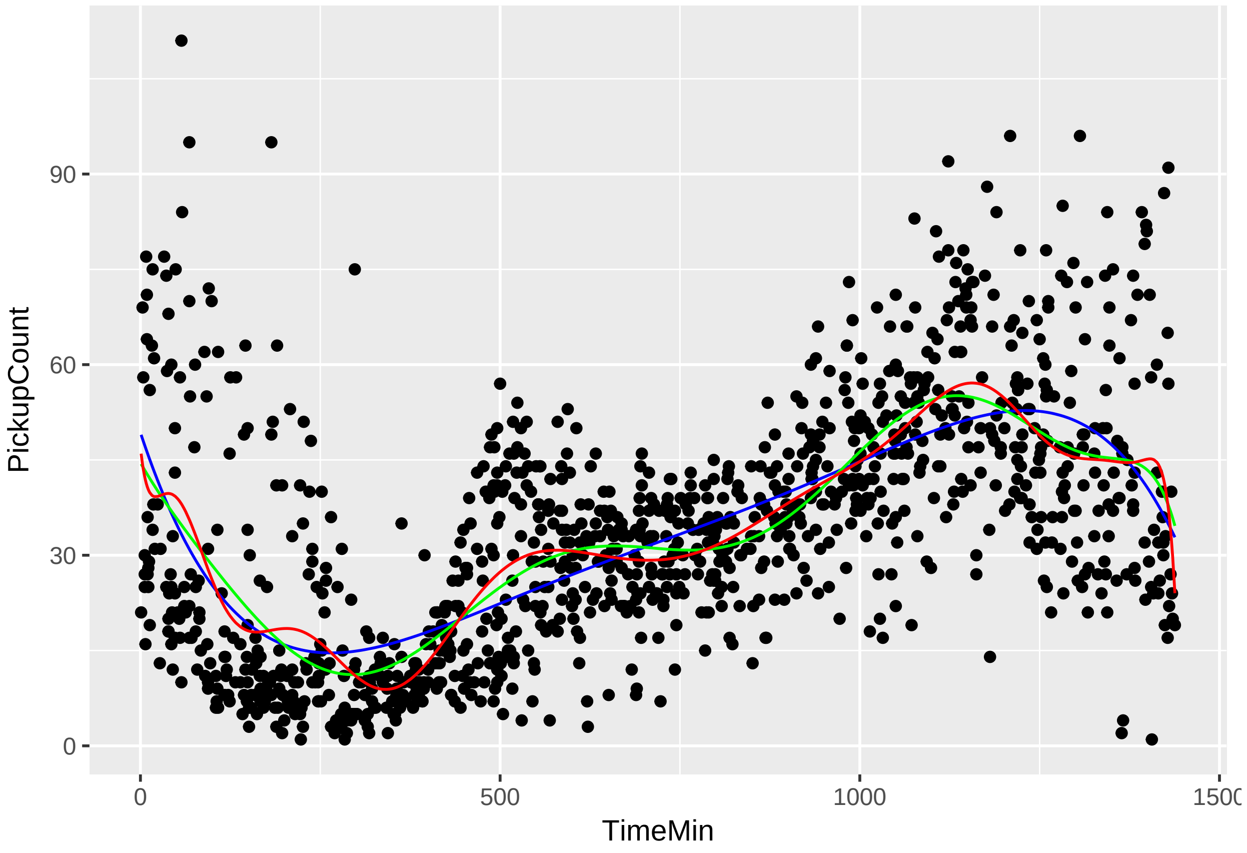
Helper functions

```
rsq = function(y, predict) {
    predict[is.na(predict)] <- 0
    tss = sum((y - mean(y))^2)
    rss = sum((y - predict)^2)
    rsq_ = max(0, 1 - rss/tss)
    return(rsq_)
}
```

Regression models: polynomials

```
poly_5 <- lm(PickupCount ~ poly(TimeMin, degree = 5, raw = TRUE),
    data = taxi_train)
poly_10 <- lm(PickupCount ~ poly(TimeMin, degree = 10, raw = TRUE),
    data = taxi_train)
poly_25 <- lm(PickupCount ~ poly(TimeMin, degree = 25, raw = TRUE),
    data = taxi_train)
```

```r
taxi.predict <- data.frame(taxi_train, pred_5 = predict(poly_5),
    pred_10 = predict(poly_10), pred_25 = predict(poly_25))
ggplot(taxi.predict, aes(x = TimeMin, y = PickupCount)) + geom_point() +
    geom_line(aes(y = pred_5), colour = "blue") + geom_line(aes(y = pred_10),
    colour = "green") + geom_line(aes(y = pred_25), colour = "red") +
    ylab(label = "PickupCount") + xlab("TimeMin")
```



The 5th degree polynomial fits the general trend of the data but misses the morning spike at 500 minutes. The 25th degree polynomial has a very high degree of curvature and appears to overfit to the data.

Analyse the R2 of the models

```r
accuracy <- data.frame(model = "5th degree poly", train_rsquared = rsq(taxi_train$PickupCount,
    predict(poly_5, newdata = taxi_train)), test_rsquared = rsq(taxi_test$PickupCount,
    predict(poly_5, newdata = taxi_test)))


accuracy <- rbind(accuracy, data.frame(model = "10th degree poly",
    train_rsquared = rsq(taxi_train$PickupCount, predict(poly_10,
        newdata = taxi_train)), test_rsquared = rsq(taxi_test$PickupCount,
        predict(poly_10, newdata = taxi_test))))


accuracy <- rbind(accuracy, data.frame(model = "25th degree poly",
    train_rsquared = rsq(taxi_train$PickupCount, predict(poly_25,
        newdata = taxi_train)), test_rsquared = rsq(taxi_test$PickupCount,
        predict(poly_25, newdata = taxi_test))))
```

```
accuracy
```

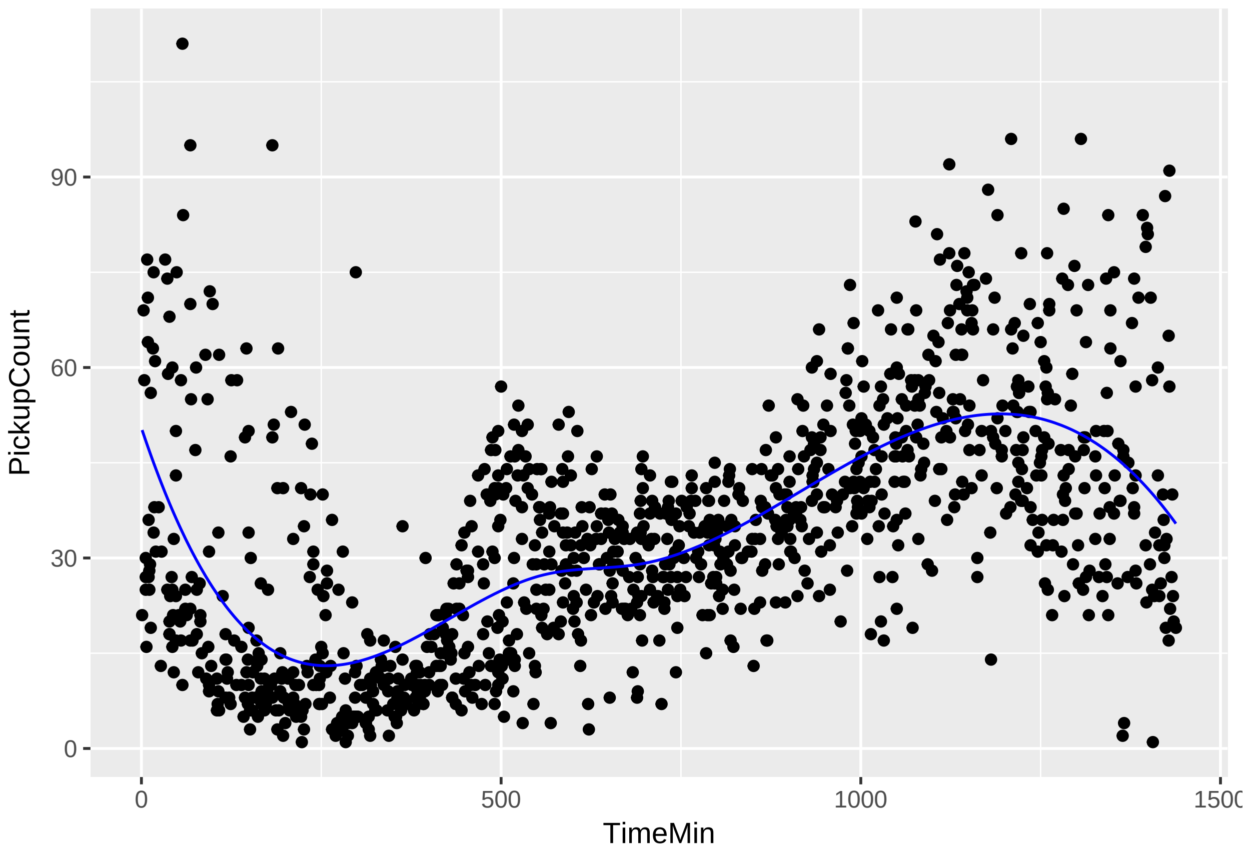
model	train_rsquared	test_rsquared
5th degree poly	0.4240156	0.3855844
10th degree poly	0.4484392	0.4132089
25th degree poly	0.4597791	0.4223648

The 25th degree polynomimal has the highest degree of freedom and can model all the nuances of the training data. It has the highest training and testing R^2 so does not appear to suffer from overfitting issues (this may also be due to the fact that the testing and training data are very similar in appearance)

Regression models: cubic b splines

The knots are set to be at the 40th, 50th and 70th quantile of the data.

```
library(splines)
quarts = quantile(taxi_train$TimeMin, probs = c(0.4, 0.5, 0.7))
cubicbspline = lm(PickupCount ~ bs(TimeMin, knots = quarts),
    data = taxi_train)

accuracy <- rbind(accuracy, data.frame(model = "Cubic spline",
    train_rsquared = rsq(taxi_train$PickupCount, predict(cubicbspline,
        newdata = taxi_train)), test_rsquared = rsq(taxi_test$PickupCount,
        predict(cubicbspline, newdata = taxi_test))))


taxi.predict["cubicsplinepred"] <- predict(cubicbspline)
ggplot(taxi.predict, aes(x = TimeMin, y = PickupCount)) + geom_point() +
    geom_line(aes(y = cubicsplinepred), colour = "blue") + ylab(label = "PickupCount") +
    xlab("TimeMin")
```



```
accuracy
```

model	train_rsquared	test_rsquared
5th degree poly	0.4240156	0.3855844
10th degree poly	0.4484392	0.4132089
25th degree poly	0.4597791	0.4223648
Cubic spline	0.4356485	0.3968863

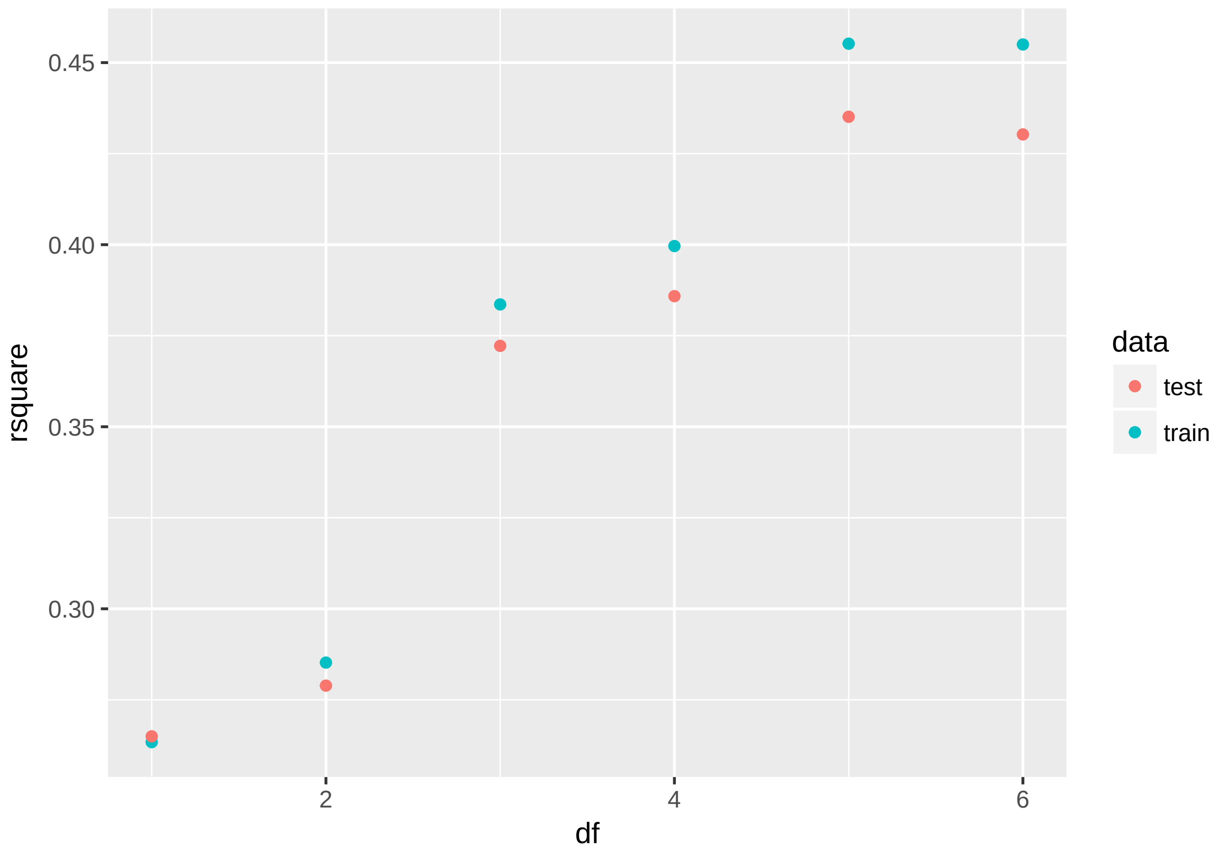
The cubic B spline has comparable performance to a 10th degree polynomial but has lower R^2 compared to the 25th degree polynomial.

Regression models: natural cubic splines

Perform 5 fold cross validation to select the number of degrees of freedom

```
## degrees of freedon

dfs <- 1:6
## create k partitions
taxi_train$splits <- cut(sample(1:nrow(taxi_train), nrow(taxi_train)),
    breaks = 5, labels = FALSE)

model.performance <- function(df, train, test) {
    mod <- lm(PickupCount ~ ns(TimeMin, df = df), data = train)
    c(train.r2 = rsq(train$PickupCount, predict(mod, newdata = train)),
```

```r
        test.r2 = rsq(test$PickupCount, predict(mod, newdata = test)))
}

## iterate over the splits, holding each one out as the test
## set.
perform.5fold <- lapply(unique(taxi_train$splits), function(split) {
    train <- taxi_train$split == split
    mt.train <- taxi_train[train, ]
    mt.test <- taxi_train[-train, ]
    data.frame(t(sapply(dfs, model.performance, train = mt.train,
        test = mt.test)), df = dfs)
})

## collect the k sets of model statistics in a data.frame
perform.5fold <- do.call(rbind, perform.5fold)

## aggregate across the k sets, averaging model statistics for
## each df
perform.5fold <- lapply(split(perform.5fold, perform.5fold$df),
    function(x) {
        data.frame(rsquare = c(mean(x$train.r2), mean(x$test.r2)),
            data = c("train", "test"), df = unique(x$df))
    })

## collect the results
perform.5fold <- do.call(rbind, perform.5fold)

## plot the results
ggplot(perform.5fold, aes(x = df, y = rsquare, color = data)) +
    geom_point()
```



The highest R^2 value on the testing set is obtained with 5 degrees of freedom and this was chosen as the optimal value to be used in the natural spline

```r
nspline <- lm(PickupCount ~ ns(TimeMin, df = 5), data = taxi_train)
accuracy <- rbind(accuracy, data.frame(model = "Natural Cubic spline with 5 degree of freedom",
    train_rsquared = rsq(taxi_train$PickupCount, predict(nspline,
        newdata = taxi_train)), test_rsquared = rsq(taxi_test$PickupCount,
        predict(nspline, newdata = taxi_test))))
accuracy
```

model	train_rsquared	test_rsquared
5th degree poly	0.4240156	0.3855844
10th degree poly	0.4484392	0.4132089
25th degree poly	0.4597791	0.4223648
Cubic spline	0.4356485	0.3968863
Natural Cubic spline with 5 degree of freedom	0.4465072	0.4110788

The natural spline has superior performance to the cubic spline, due to the higher degrees of freedom.

Smoothing Splines

```r
spline = smooth.spline(taxi_train$TimeMin, taxi_train$PickupCount,
    cv = TRUE)
spline["spar"]
```

```
## $spar
## [1] 0.7651076
```

```
fitsmoothspline = smooth.spline(taxi_train$TimeMin, taxi_train$PickupCount,
    spar = spline["spar"])

splinersq = function(pred, y) {
    tss = sum((y - mean(y))^2)
    rss = sum((y - pred)^2)
    rsq_ = max(0, 1 - rss/tss)
    return(rsq_)
}
pred_train <- predict(fitsmoothspline, taxi_train$TimeMin)$y
pred_test <- predict(fitsmoothspline, taxi_test$TimeMin)$y

accuracy <- rbind(accuracy, data.frame(model = "Smoothing spline",
    train_rsquared = splinersq(pred_train, taxi_train$PickupCount),
    test_rsquared = splinersq(pred_test, taxi_test$PickupCount)))
```

```
accuracy
```

model	train_rsquared	test_rsquared
5th degree poly	0.4240156	0.3855844
10th degree poly	0.4484392	0.4132089
25th degree poly	0.4597791	0.4223648
Cubic spline	0.4356485	0.3968863
Natural Cubic spline with 5 degree of freedom	0.4465072	0.4110788
Smoothing spline	0.4569129	0.4254253

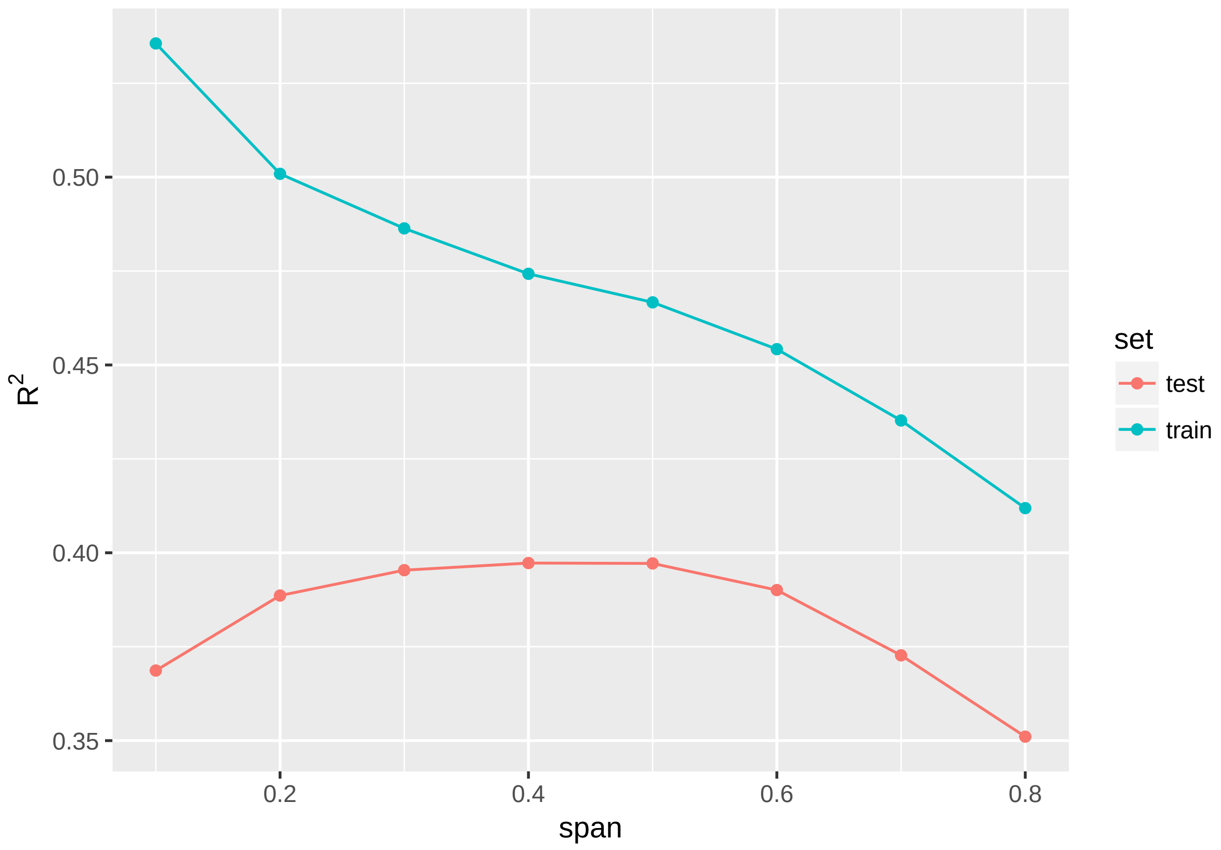
The smoothing spline appears has the highest testing and training R2 compared to the other spline models and is comparable to a 25th degree polynomial.

Locally Weighted regression model

Perform 5 fold cross validation to select the optimal span.

```
library(ggplot2)
loess_5foldcv = function(data_set) {
    data_set$splits <- cut(sample(1:nrow(data_set), nrow(data_set)),
        breaks = 5, labels = FALSE)
    spans <- seq(0.1, 0.8, by = 0.1)

    ## iterate over the splits, holding each one out as the test
    ## set.
    perform.5fold <- list()
    for (split in data_set$splits) {
        train <- data_set$split == split
        mt.train <- data_set[train, ]
        mt.test <- data_set[-train, ]

        train.r2 <- c()
        test.r2 <- c()
        for (sp in spans) {
            mod <- loess(PickupCount ~ TimeMin, span = sp, data = mt.train)
            train.r2 <- c(train.r2, rsq(mt.train$PickupCount,
                predict(mod, newdata = mt.train)))
            test.r2 <- c(test.r2, rsq(mt.test$PickupCount, predict(mod,
                newdata = mt.test)))
        }

        loess.r2 <- data.frame(span = c(spans, spans), set = rep(c("train",
            "test"), each = length(spans)), r2 = c(train.r2,
            test.r2))
        perform.5fold <- c(perform.5fold, list(loess.r2))
    }

    ## collect the k sets of model statistics in a data.frame
    perform.5fold <- do.call(rbind, perform.5fold)

    ## aggregate across the k sets, averaging model statistics for
    ## each df
    perform.5fold <- aggregate(perform.5fold$r2, perform.5fold[c("span",
        "set")], FUN = mean)

    ## plot the results
    ggplot(perform.5fold, aes(x = span, y = x, color = set)) +
        geom_point() + geom_line() + ylab(expression(R^2))
}
```

```
loess_5foldcv(taxi_train)
```



A span of 0.2 has the highest R^2 value on the testing set and this can be chosen as the optimal span.

```
loess_model <- loess(PickupCount ~ TimeMin, span = 0.2, data = taxi_train)
accuracy <- rbind(accuracy, data.frame(model = "Locally weighted regression model",
    train_rsquared = rsq(taxi_train$PickupCount, predict(loess_model,
        newdata = taxi_train)), test_rsquared = rsq(taxi_test$PickupCount,
        predict(loess_model, newdata = taxi_test$TimeMin))))
accuracy
```

model	train_rsquared	test_rsquared
5th degree poly	0.4240156	0.3855844
10th degree poly	0.4484392	0.4132089
25th degree poly	0.4597791	0.4223648
Cubic spline	0.4356485	0.3968863
Natural Cubic spline with 5 degree of freedom	0.4465072	0.4110788
Smoothing spline	0.4569129	0.4254253
Locally weighted regression model	0.4583269	0.4271821

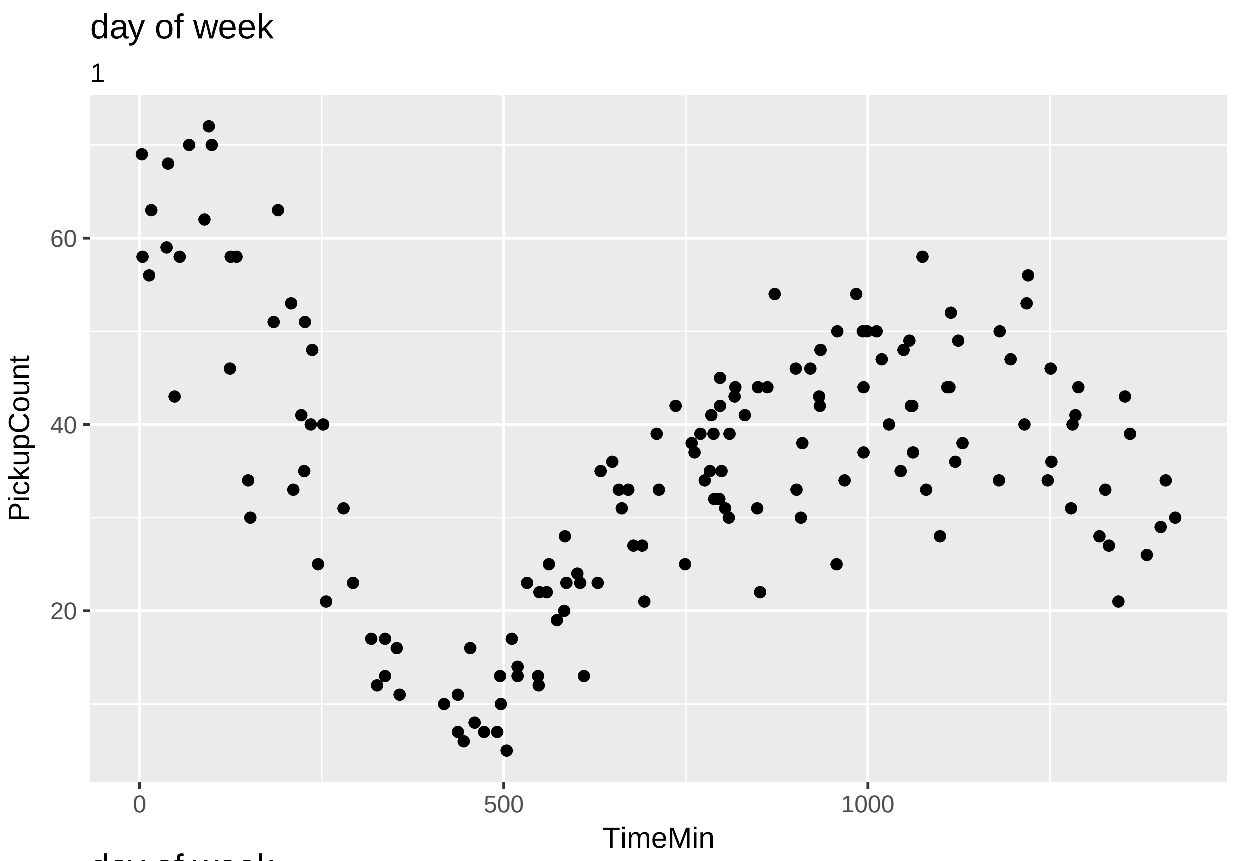
The locally weighted regression model has very similar R^2 to the 25th degree polynomial on both the testing and training sets. Overall, all the models have similar performance on the testing and training sets and no model significantly outperforms any other model. This suggests that for this particular data set the choice of model does not significantly affect the predictive accuracy.

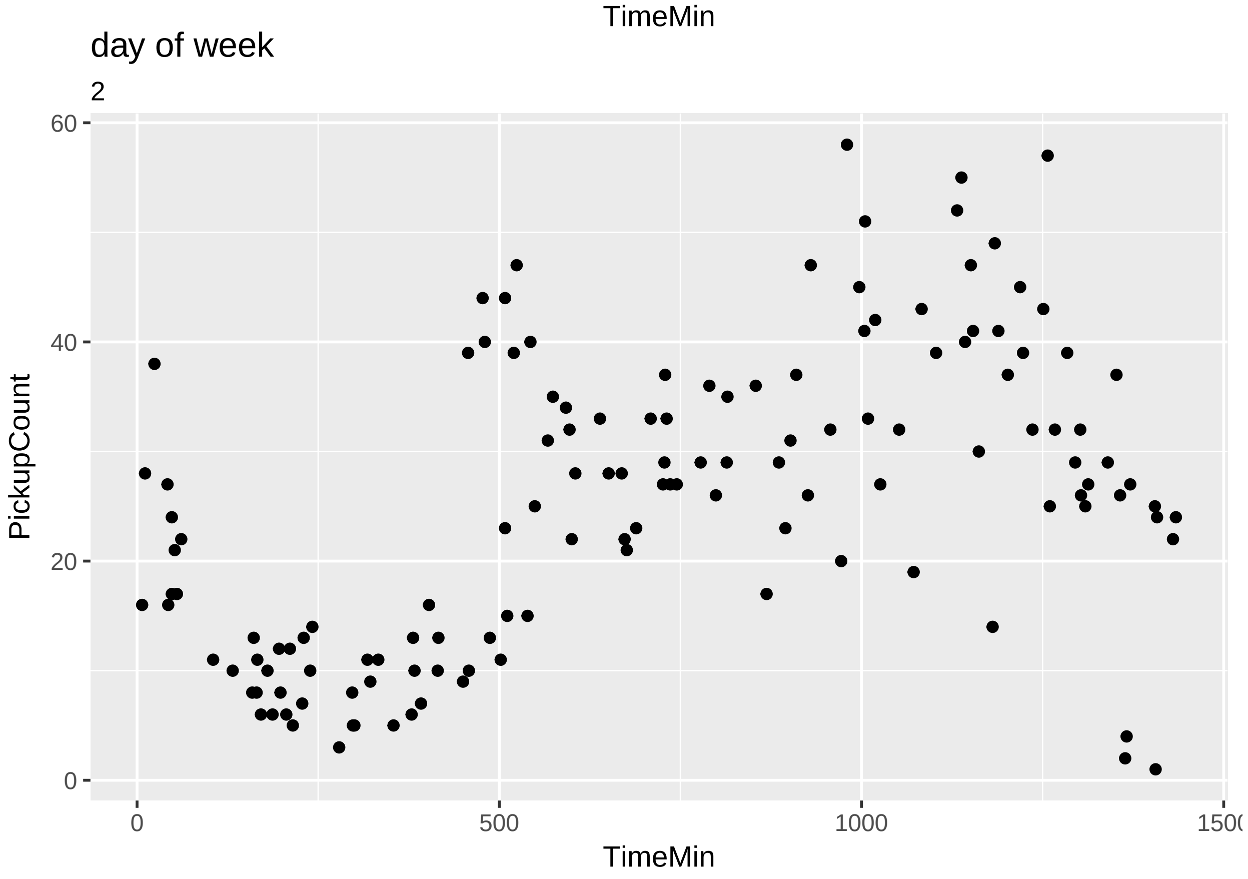
However, splines generally produce more stable estimates compared to polynomial regression as splines increase flexibility by introducing knots and keeping the degree fixed. Extra flexibility of polynomials produces undesirable results at the boundaries while the natural cubic spline can still provide a reasonable fit to the data. Therefore a spline would be preferrable to a very high degree polynomial.

Part 1b: Adapting to Weekends

The time series plots of taxi pickups for each day of the week are shown below:

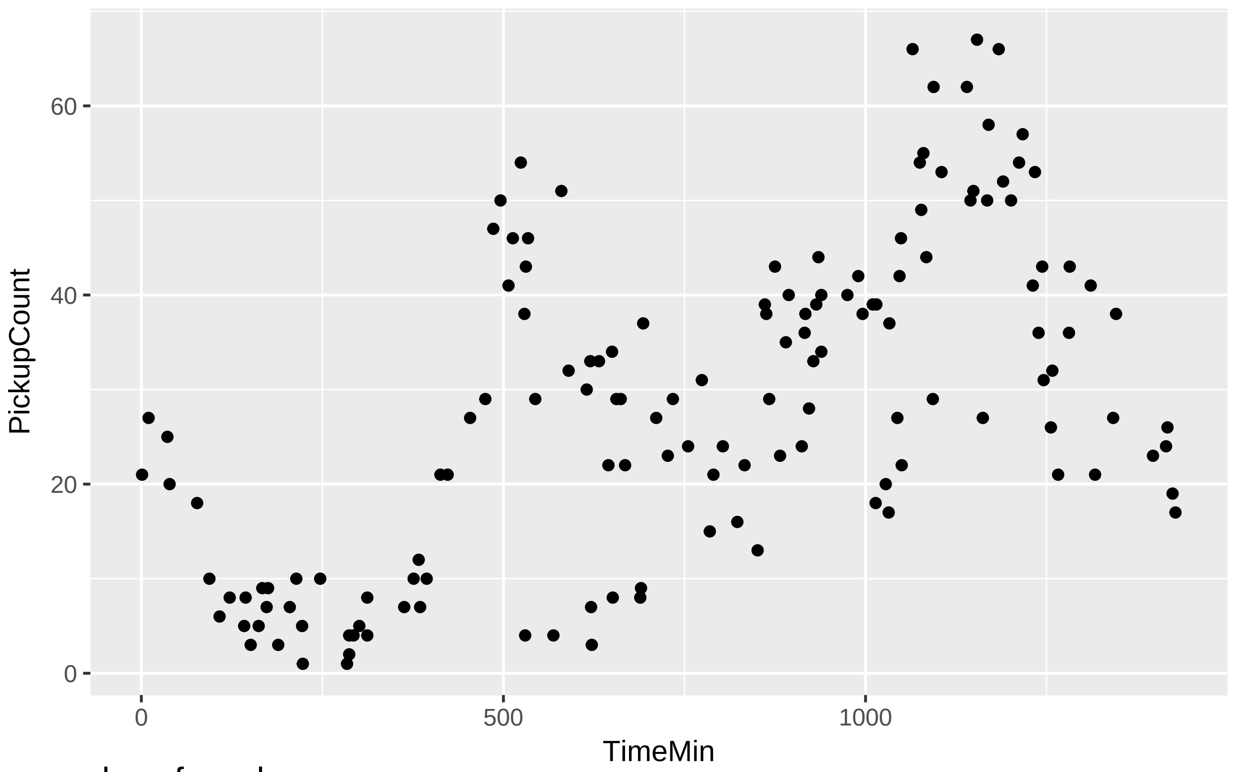
```r
for (day in 1:7) {
    df = taxi_train[taxi_train$DayOfWeek == day, ]
    print(ggplot(df, aes(x = TimeMin, y = PickupCount)) + geom_point() +
        ggtitle("day of week ", day))
}
```


day of week
1
PickupCount
60
40
20
0
500
1000
TimeMin


day of week
2
PickupCount
60
40
20
0
500
1000
TimeMin

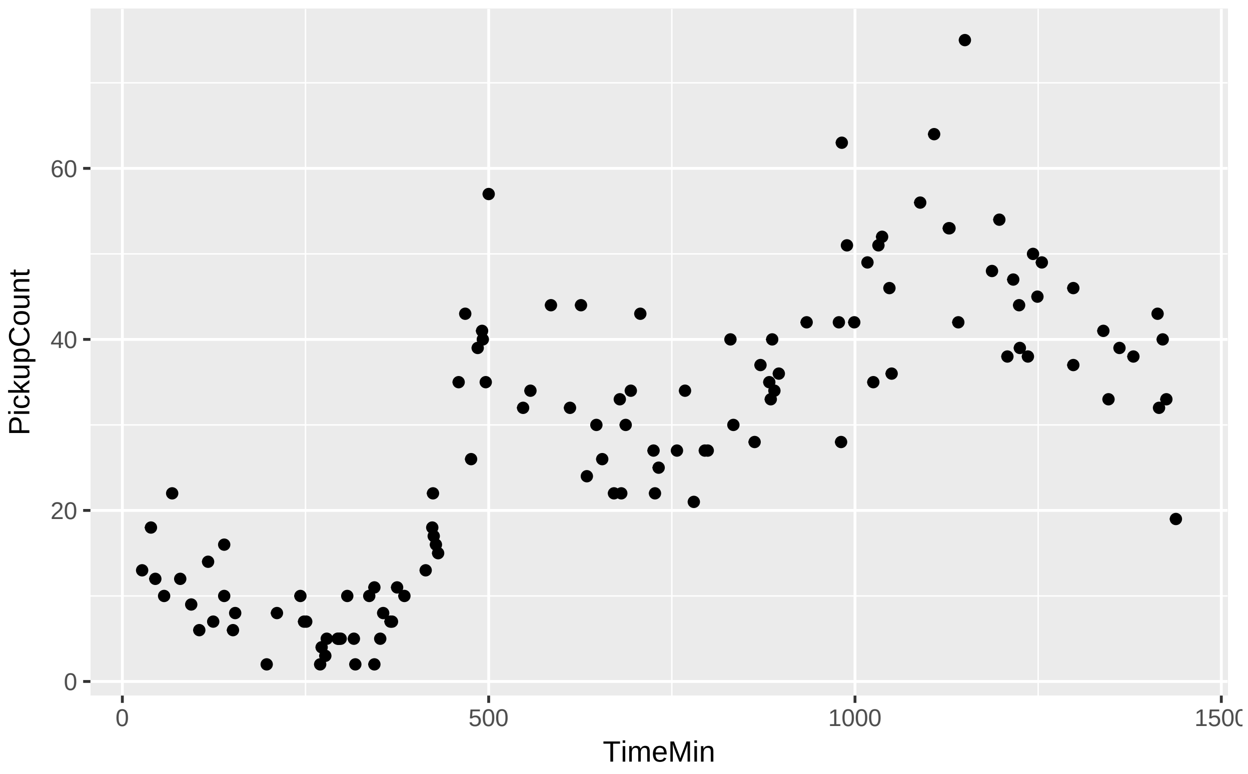
day of week

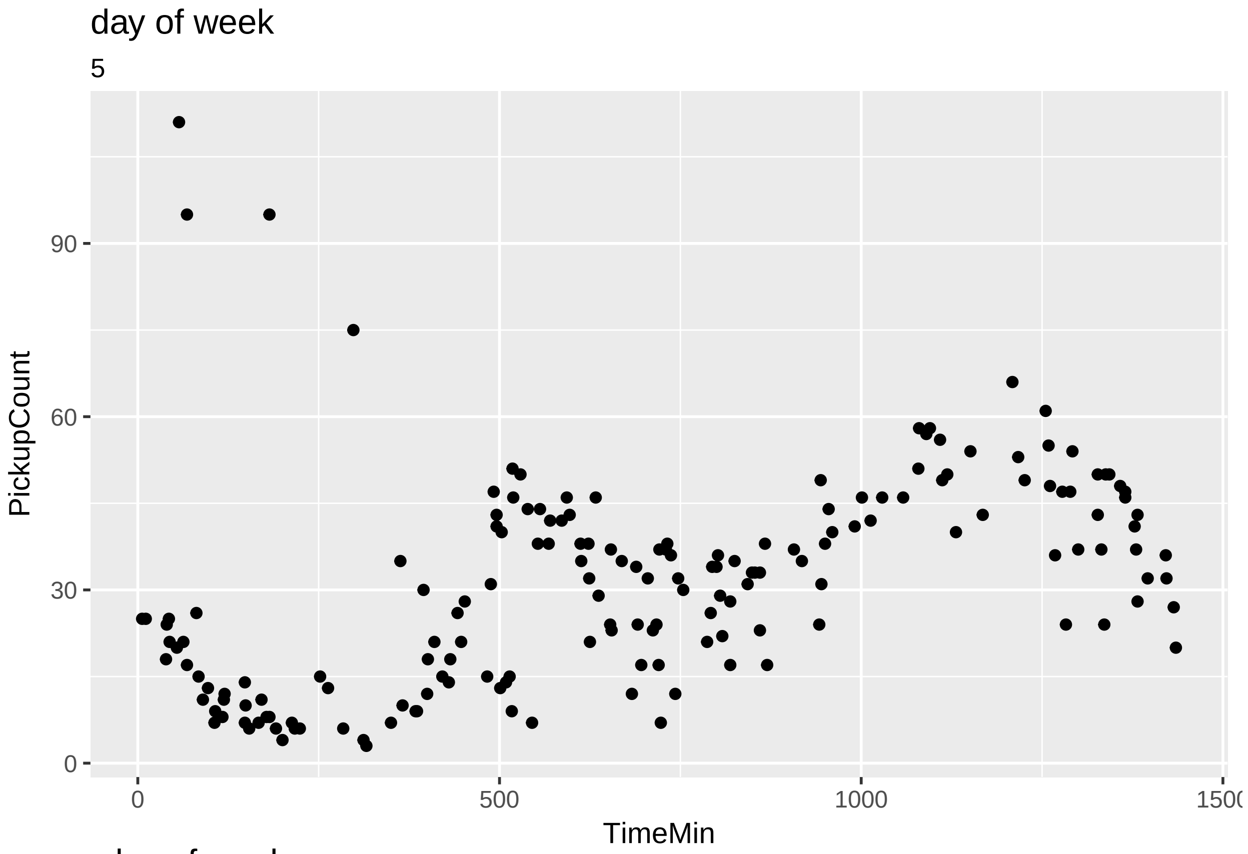
3

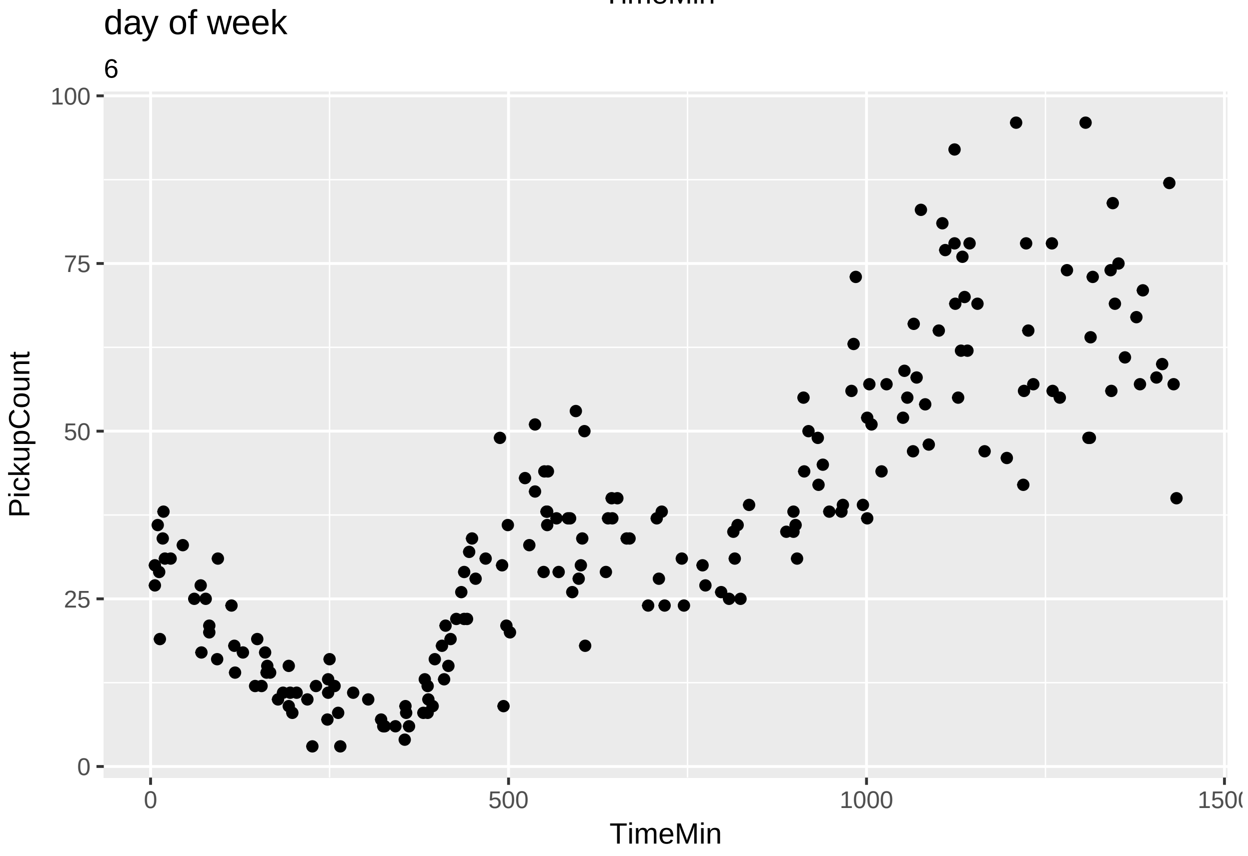


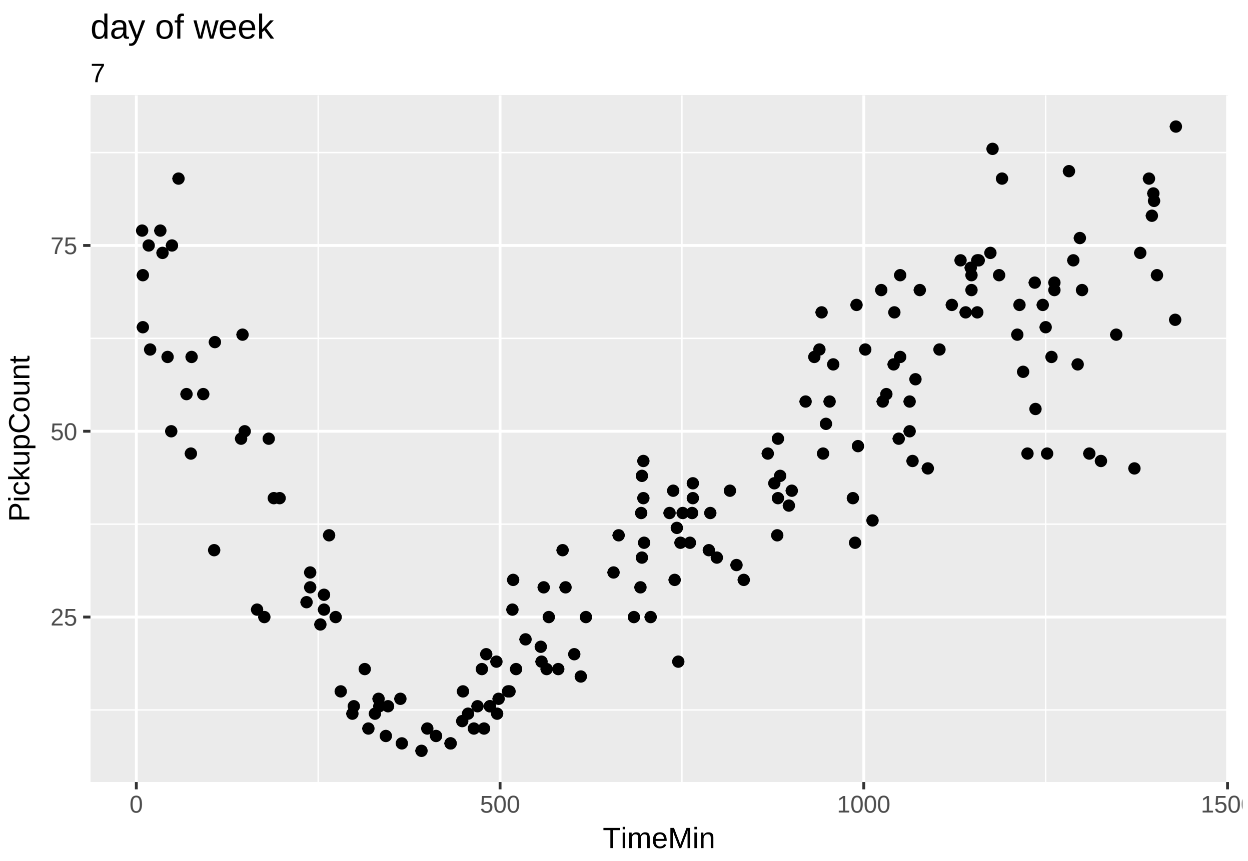
day of week

4




day of week
5
PickupCount
TimeMin
0
30
60
90
0
500
1000
1500

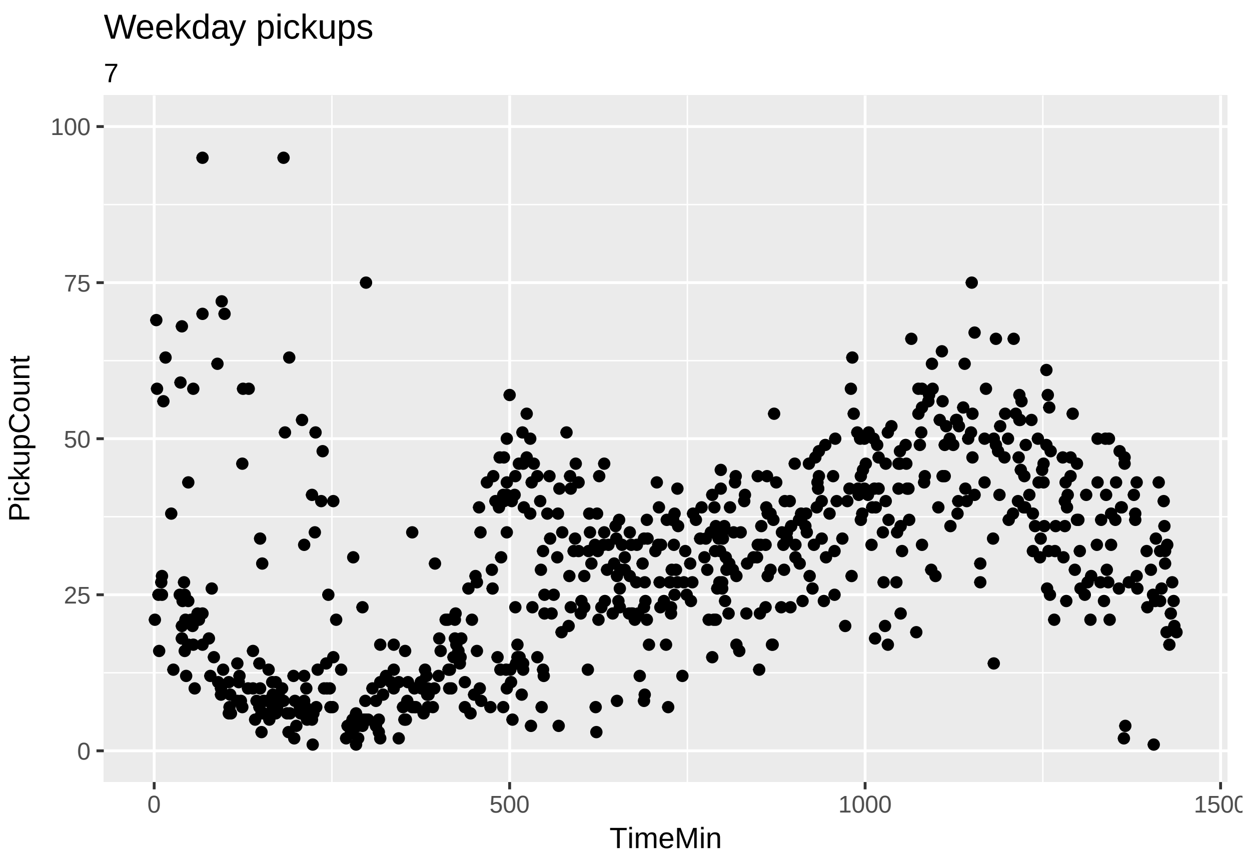

day of week
6
PickupCount
TimeMin
0
25
50
75
100
0
500
1000
1500



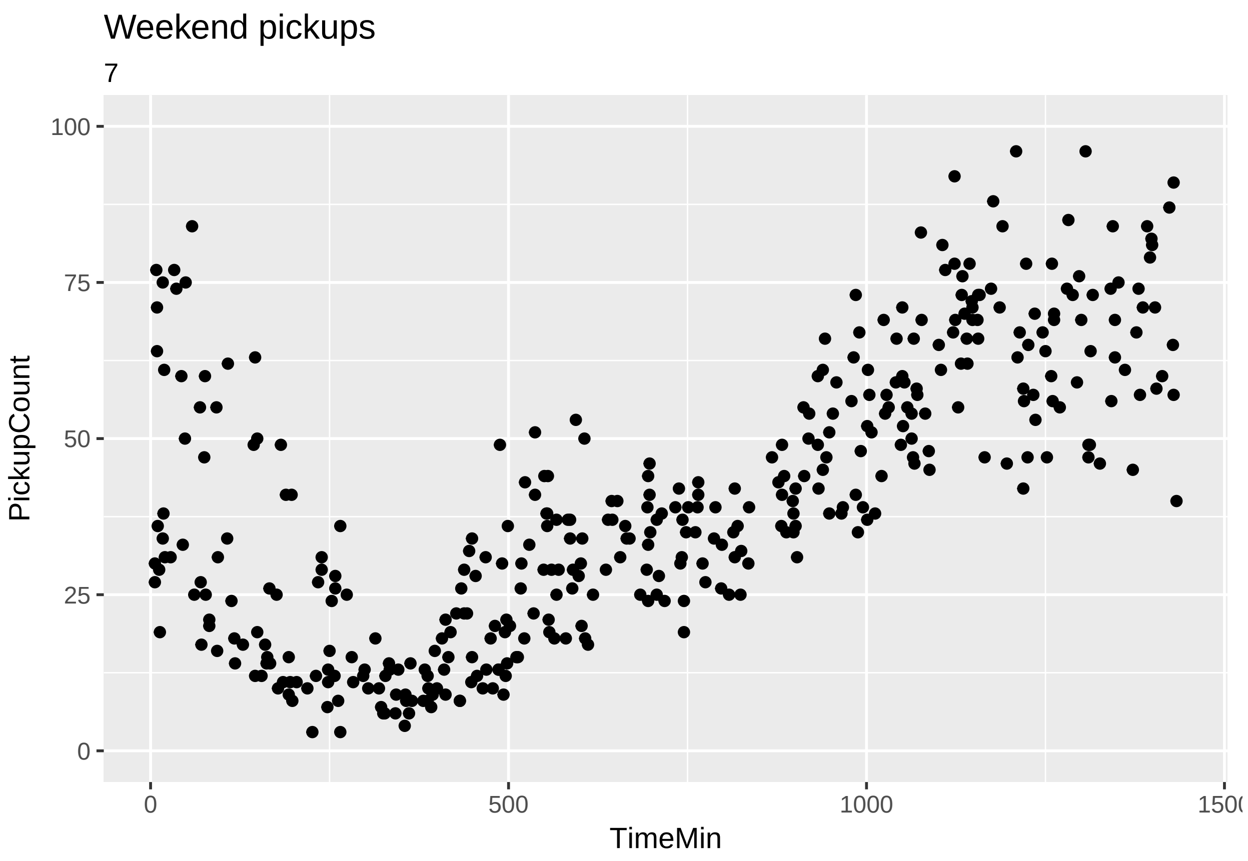
The dataset can be split into weekday and weekend pickups and the time series of weekday vs weekend is also shown below:

```
weekday_train_df = taxi_train[taxi_train$DayOfWeek != 6 & taxi_train$DayOfWeek !=
    7, ]
weekend_train_df = taxi_train[taxi_train$DayOfWeek == 6 | taxi_train$DayOfWeek ==
    7, ]
weekday_test_df = taxi_test[taxi_test$DayOfWeek != 6 & taxi_test$DayOfWeek !=
    7, ]
weekend_test_df = taxi_test[taxi_test$DayOfWeek == 6 | taxi_test$DayOfWeek ==
    7, ]
```

```
ggplot(weekday_train_df, aes(x = TimeMin, y = PickupCount)) +
    geom_point() + ggtitle("Weekday pickups ", day) + ylim(0,
    100)
```



```r
ggplot(weekend_train_df, aes(x = TimeMin, y = PickupCount)) +
    geom_point() + ggtitle("Weekend pickups", day) + ylim(0,
    100)
```



The pattern of pickups is different on the weekends compared to the weekdays, with a much more pronounced peak during the morning (500 min) on weekdays compared to the weekend and much higher pickup count in the evenings on weekends compared to weekdays.

Fit locally weighted least squares to each subset with the span set to 0.2

```
weekend_model = loess(PickupCount ~ TimeMin, span = 0.2, data = weekend_train_df)
weekday_model = loess(PickupCount ~ TimeMin, span = 0.2, data = weekday_train_df)
```

Comparison of R2 values of the new models on the separated test sets compared to model fit on the entire test set

```
loess_accuracy <- data.frame(model = "loess", test_rsquared = rsq(taxi_test$PickupCount,
    predict(loess_model, newdata = taxi_test)))

loess_accuracy <- rbind(loess_accuracy, data.frame(model = "Loess weekday model on weekday test set",
    test_rsquared = rsq(weekday_test_df$PickupCount, predict(weekday_model,
        newdata = weekday_test_df))))

loess_accuracy <- rbind(loess_accuracy, data.frame(model = "Loess weekend model on weekend test set",
    test_rsquared = rsq(weekend_test_df$PickupCount, predict(weekend_model,
        newdata = weekend_test_df))))

loess_accuracy <- rbind(loess_accuracy, data.frame(model = "Loess model on weekend test set",
    test_rsquared = rsq(weekend_test_df$PickupCount, predict(loess_model,
        newdata = weekend_test_df))))
```

```
loess_accuracy <- rbind(loess_accuracy, data.frame(model = "Loess model on weekday test set",
    test_rsquared = rsq(weekday_test_df$PickupCount, predict(loess_model,
        newdata = weekday_test_df))))
```

```
loess_accuracy
```

model	test_rsquared
loess	0.4271821
Loess weekday model on weekday test set	0.3586341
Loess weekend model on weekend test set	0.6998729
Loess model on weekend test set	0.5349427
Loess model on weekday test set	0.2921371

Predictions for weekdays and weekend made using the locally weighted regression model fit to the entire dataset has very low R2 on the weekday test set, but much higher accuracy on the weekend test set. In comparison, the weekday model fit only on the weekday training set has a higher R^2 (0.358), but this is still lower than the R^2 of the loess model predicting on the entire data set (0.427). The weekend model however has a much higher R^2 for predictions on the weekend test set (0.699) than the full loess model predicting on the weekend test set (0.534).

This suggests that the data for weekends is similar between testing and training data and hence a more specific model can have higher test accuracy. However the same does not apply for weekdays where the R^2 predictions made using a more specific model are lower than the loess model fit to the entire data set.

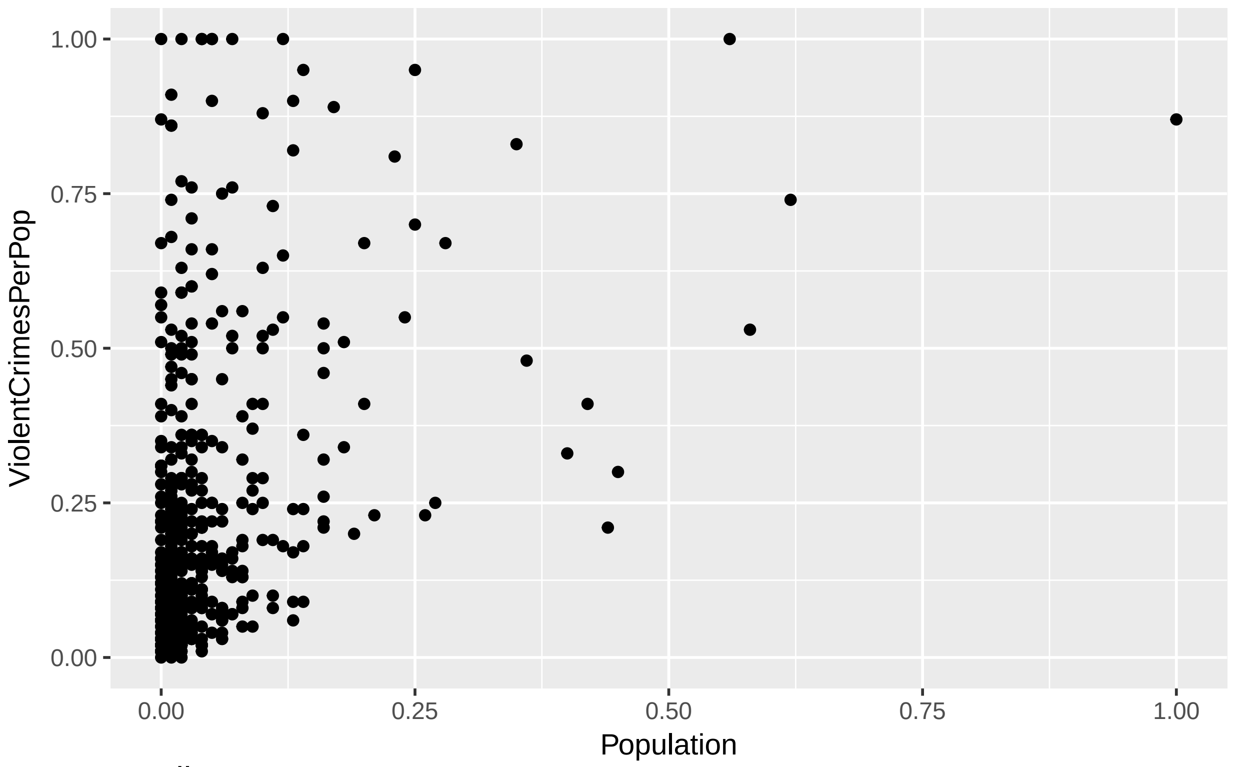
Problem 2: Predicting Crime in the City

```
crime_train <- read.table("dataset_2_train.txt", sep = "\t",
    header = TRUE)
crime_test <- read.table("dataset_2_test.txt", sep = "\t", header = TRUE)
```

```
library(ggplot2)
for (pred in names(crime_train)) {
    if (pred != "ViolentCrimesPerPop") {
        print(ggplot(crime_train, aes_string(x = pred, y = "ViolentCrimesPerPop")) +
            geom_point() + ggtitle("Predictor ", pred))
    }
}
```

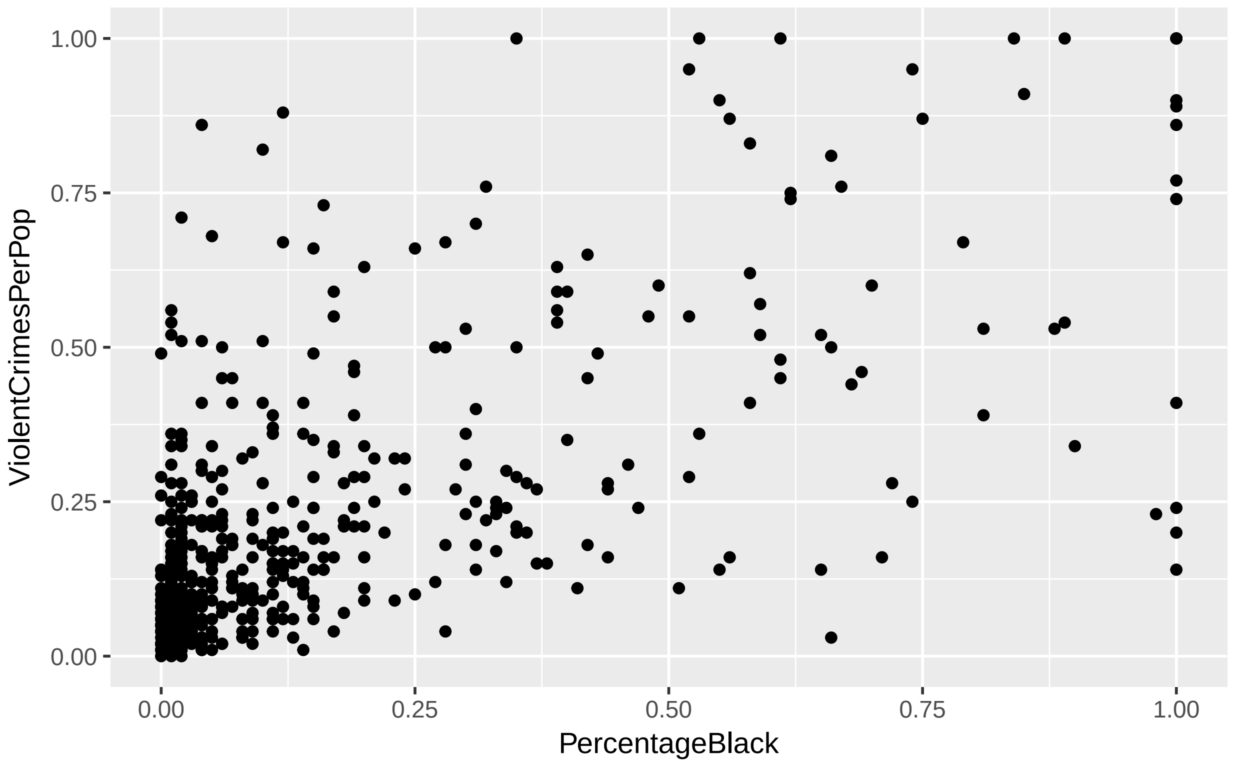
Predictor

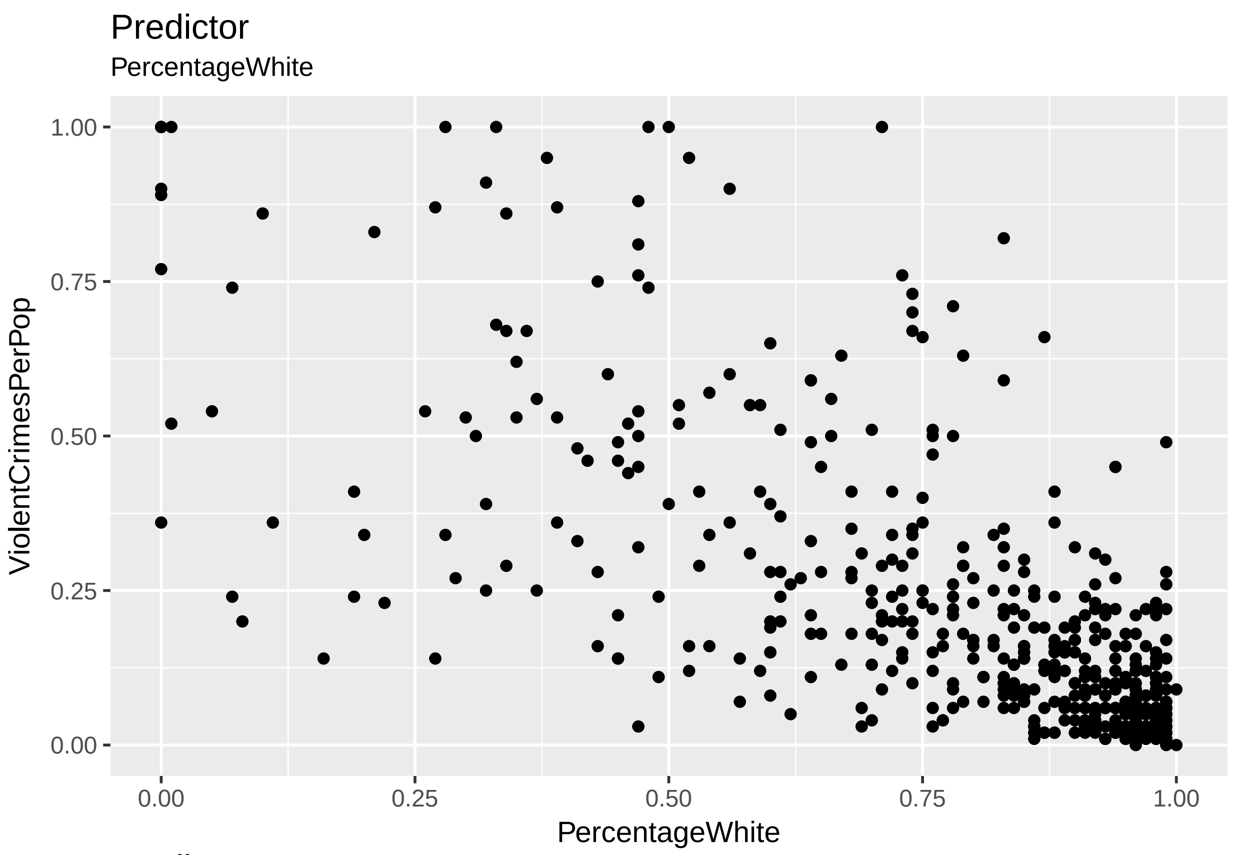
Population

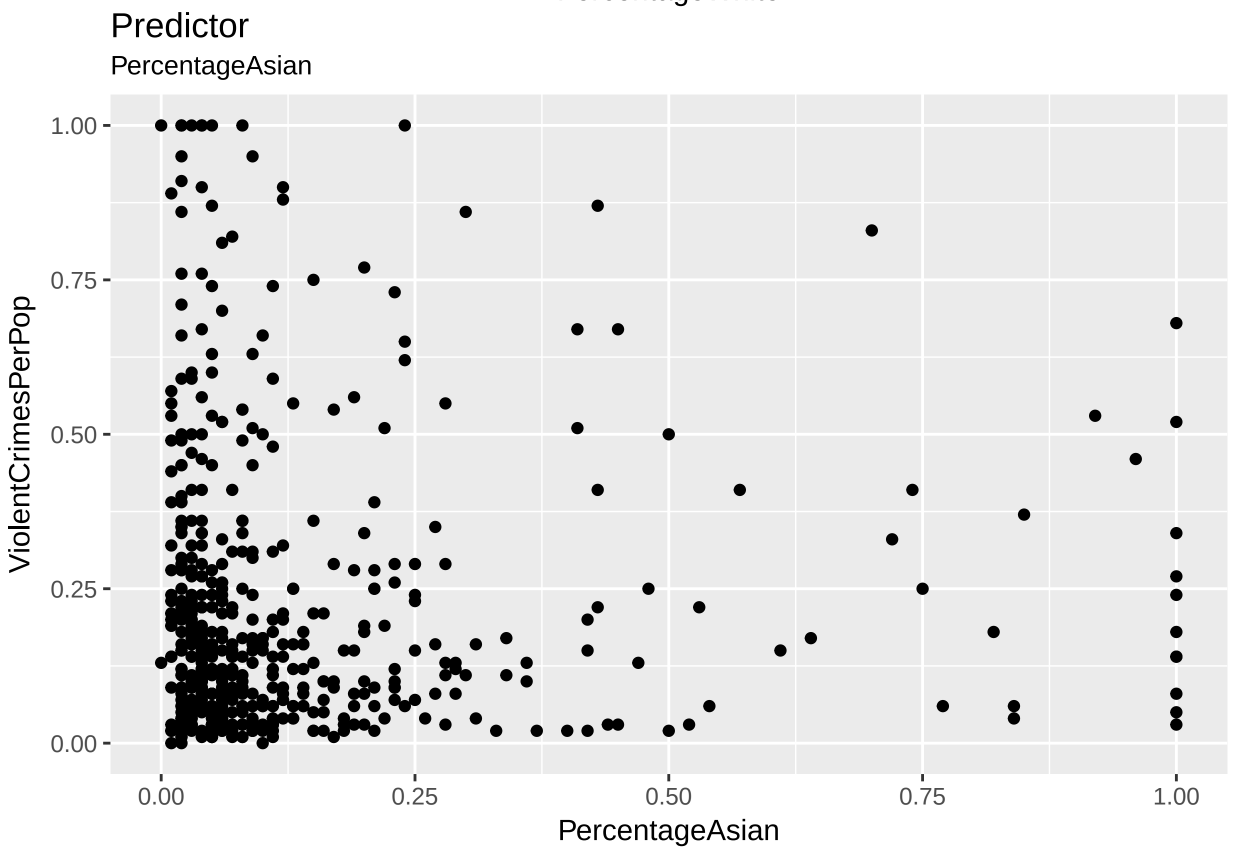


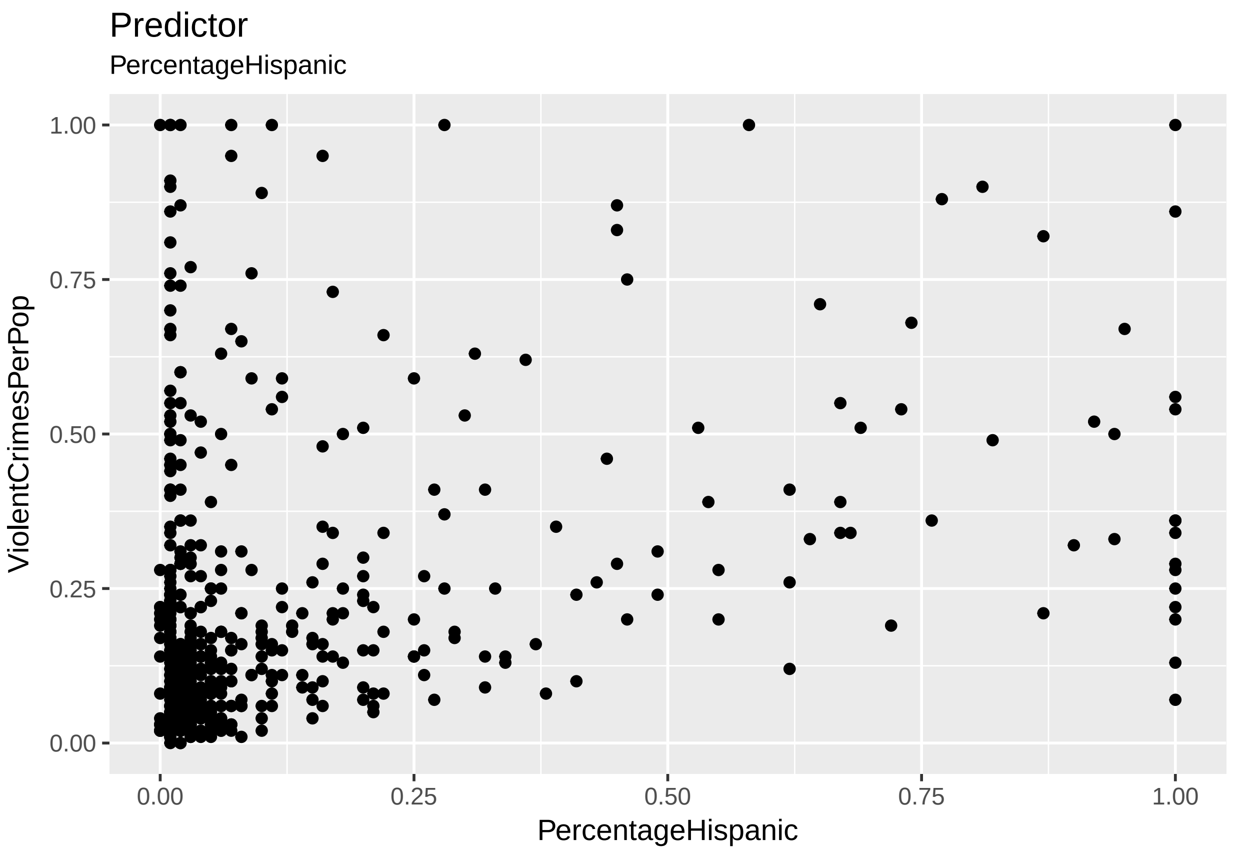
Predictor

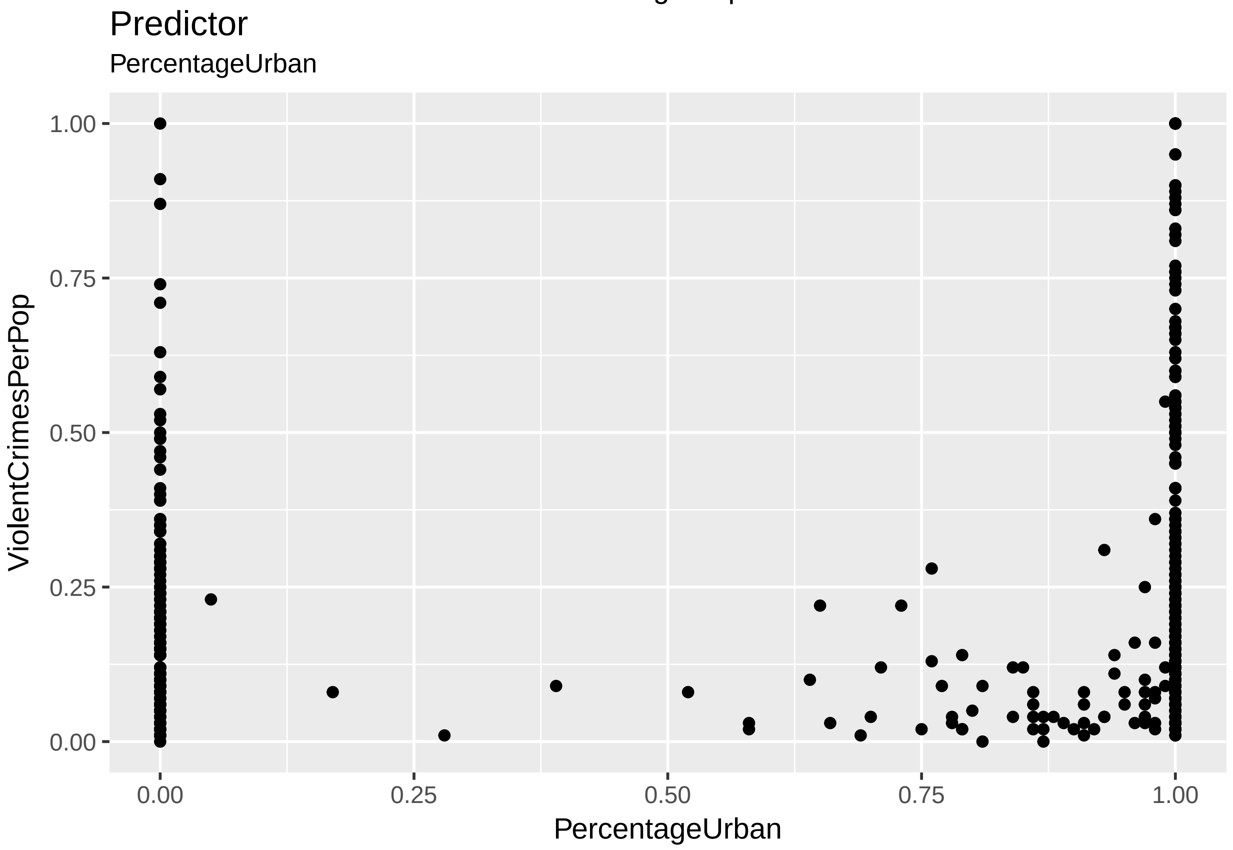
PercentageBlack

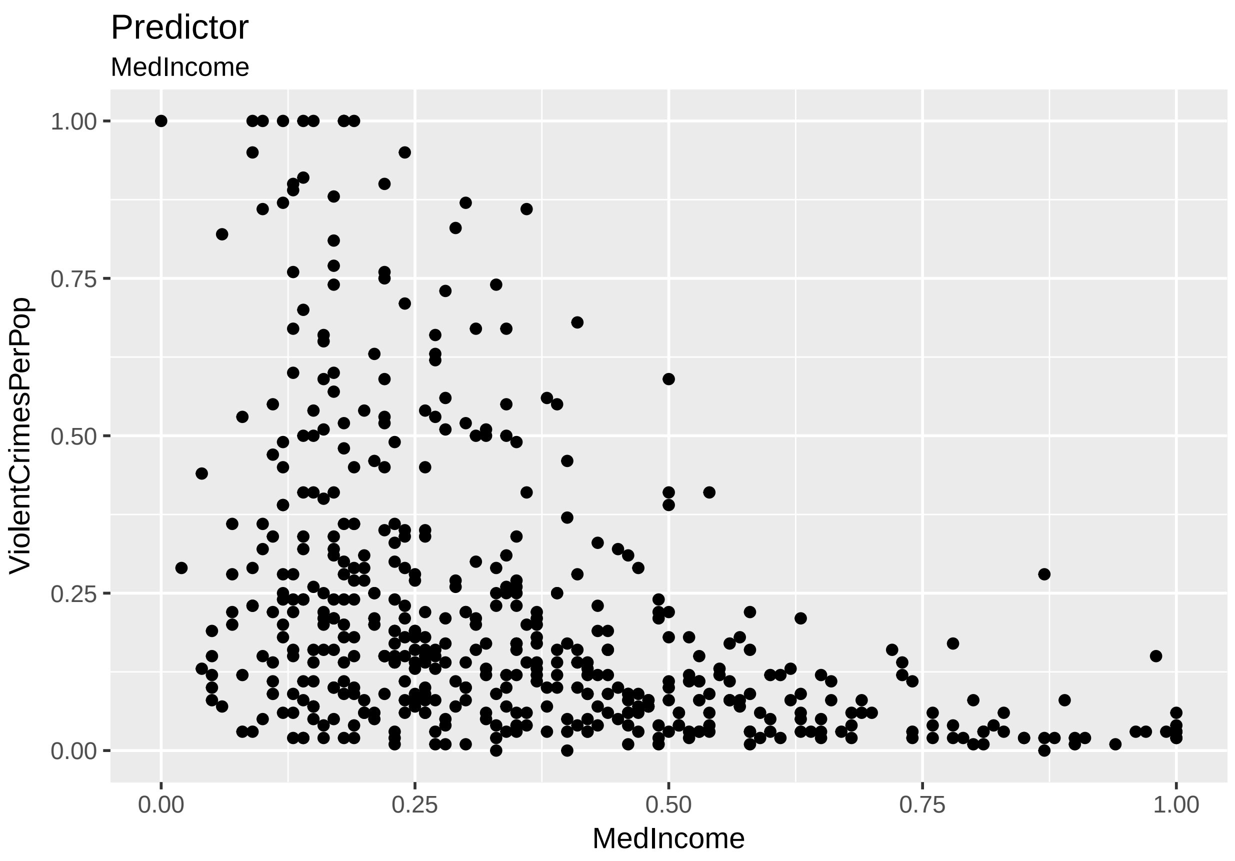



Predictor
PercentageWhite
ViolentCrimesPerPop
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
PercentageWhite


Predictor
PercentageAsian
ViolentCrimesPerPop
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
PercentageAsian


Predictor
PercentageHispanic
ViolentCrimesPerPop
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
PercentageHispanic


Predictor
PercentageUrban
ViolentCrimesPerPop
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
PercentageUrban



Examining the relationship between the crime rate and the individual predictors it is clear that the predictors all have different effects on the response variable. In particular the PercentageUrban has a highly non linear effect. Most communities are either completely urban or completely rural and there are few communities in the spectrum between the extremes. There is also a wide distribution of violent crimes in each of these groups of communities.

A non linear regression model is required is likely to be preferrable to model this data.

Polynomial Regression

```
library(splines)

lin_model <- lm(ViolentCrimesPerPop ~ ., data = crime_train)

poly_2 <- lm(ViolentCrimesPerPop ~ poly(Population, degree = 2,
    raw = TRUE) + poly(PercentageBlack, degree = 2, raw = TRUE) +
    poly(PercentageWhite, degree = 2, raw = TRUE) + poly(PercentageAsian,
    degree = 2, raw = TRUE) + poly(PercentageHispanic, degree = 2,
    raw = TRUE) + poly(PercentageUrban, degree = 2, raw = TRUE) +
    poly(MedIncome, degree = 2, raw = TRUE), data = crime_train)


poly_3 <- lm(ViolentCrimesPerPop ~ poly(Population, degree = 3,
    raw = TRUE) + poly(PercentageBlack, degree = 3, raw = TRUE) +
    poly(PercentageWhite, degree = 3, raw = TRUE) + poly(PercentageAsian,
    degree = 3, raw = TRUE) + poly(PercentageHispanic, degree = 3,
```

```r
    raw = TRUE) + poly(PercentageUrban, degree = 3, raw = TRUE) +
    poly(MedIncome, degree = 3, raw = TRUE), data = crime_train)


bspline = lm(ViolentCrimesPerPop ~ bs(Population, df = 3) + bs(PercentageBlack,
    df = 3) + bs(PercentageWhite, df = 3) + bs(PercentageAsian,
    df = 3) + bs(PercentageHispanic, df = 3) + bs(PercentageUrban,
    df = 3) + bs(MedIncome, df = 3), data = crime_train)


crime.model.performance <- function(model) {
    c(train.r2 = rsq(crime_train$ViolentCrimesPerPop, predict(model,
        newdata = crime_train)), test.r2 = rsq(crime_test$ViolentCrimesPerPop,
        predict(model, newdata = crime_test)))
}

crime_accuracy <- data.frame(t(sapply(list(lin_model, poly_2,
    poly_3, bspline), crime.model.performance)))
crime_accuracy["models"] = c("lin_model", "poly_2", "poly_3",
    "bspline")
crime_accuracy
```

train.r2	test.r2	models
0.6184731	0.5553841	lin_model
0.6327907	0.5753024	poly_2
0.6437397	0.5734467	poly_3
0.6437397	0.5734467	bspline

The bspline and 3rd degree polynomial models have the best performance on the testing set due to their ability to model the highly non linear effects of the predictors on the outcome variable. However there are no significant differences in performance between the models.

Part 2b: GAM models

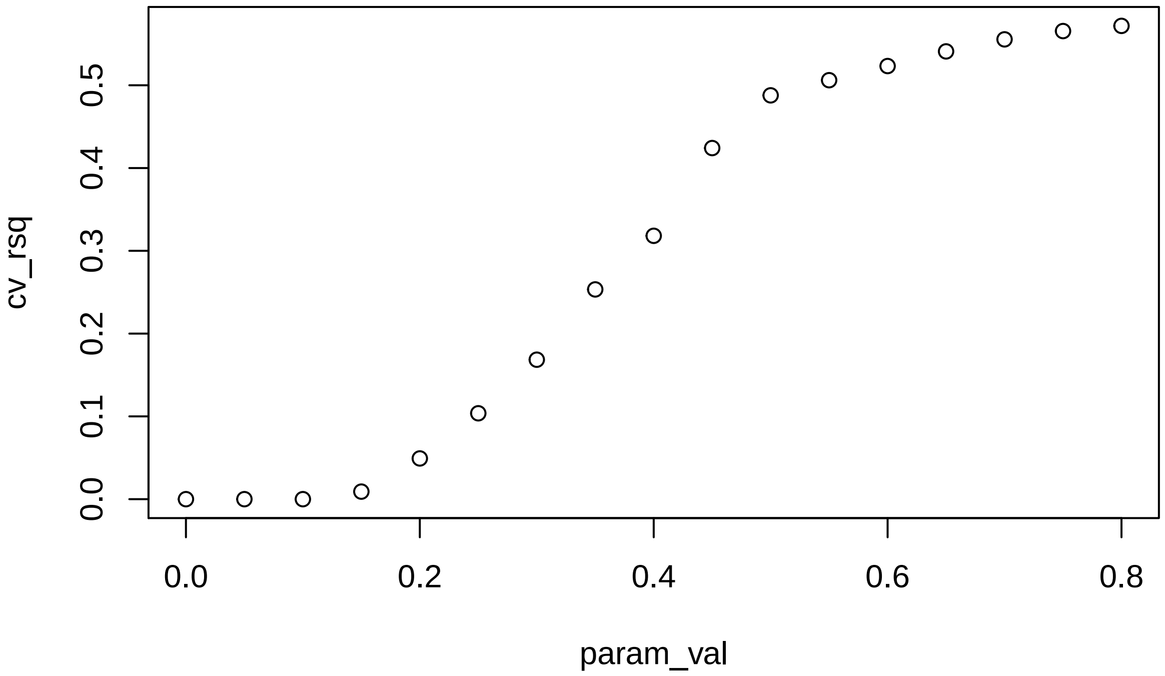
A GAM model has the advantage of providing a regularized and interpretable solution for data that contains highly nonlinear effects.

All the predictors can be used in the model each fit with a smoothing spline basis function. The smoothing parameter for these splines can be chosen by cross validation on the training set:

```r
library(gam)
loess_gam = function(data_set) {
    data_set$splits <- cut(sample(1:nrow(data_set), nrow(data_set)),
        breaks = 5, labels = FALSE)
    spans <- seq(0, 0.8, by = 0.05)

    ## iterate over the splits, holding each one out as the test
    ## set.
    perform.5fold <- list()
    for (split in data_set$splits) {
        train <- data_set$split == split
        mt.train <- data_set[train, ]
        mt.test <- data_set[-train, ]

        train.r2 <- c()
        test.r2 <- c()

        for (sp in spans) {
            mod <- gam(ViolentCrimesPerPop ~ s(Population, spar = sp) +
                s(PercentageBlack, spar = sp) + s(PercentageWhite,
                spar = sp) + s(PercentageAsian, spar = sp) +
                s(PercentageHispanic, spar = sp) + s(PercentageUrban,
                spar = sp) + s(MedIncome, spar = sp), data = mt.train)
            train.r2 <- c(train.r2, rsq(mt.train$ViolentCrimesPerPop,
                predict(mod, newdata = mt.train)))
            test.r2 <- c(test.r2, rsq(mt.test$ViolentCrimesPerPop,
                predict(mod, newdata = mt.test)))
        }

        loess.r2 <- data.frame(span = c(spans, spans), set = rep(c("train",
            "test"), each = length(spans)), r2 = c(train.r2,
            test.r2))
        perform.5fold <- c(perform.5fold, list(loess.r2))
    }

    ## collect the k sets of model statistics in a data.frame
    perform.5fold <- do.call(rbind, perform.5fold)

    ## aggregate across the k sets, averaging model statistics for
    ## each df
    perform.5fold <- aggregate(perform.5fold$r2, perform.5fold[c("span",
        "set")], FUN = mean)

    ## plot the results
    ggplot(perform.5fold, aes(x = span, y = x, color = set)) +
        geom_point() + geom_line() + ylab(expression(R^2))
}
```

```
train = crime_train
param_val <- seq(0, 0.8, by = 0.05)
k = 5

# Input: Training data frame: 'train', Output: Vector of R^2
# values for the provided parameters: 'cv_rsq'

num_param = length(param_val)  # Number of parameters set.seed(109) # Set seed for random number genera

# Divide training set into k folds by sampling uniformly at
# random # folds[s] has the fold index for train instance 's'
folds = sample(1:k, nrow(train), replace = TRUE)

cv_rsq = rep(0, num_param)  # Store cross-validated R^2 for different parameter values

# Iterate over parameter values
for (i in 1:num_param) {
    # Iterate over folds to compute R^2 for parameter
    for (j in 1:k) {
        # Fit model on all folds other than 'j' with parameter value
        # param_val[i] model.loess = loess(PickupCount ~ TimeMin,
        # span = param_val[i], data = train[folds!=j, ],
        model_1 <- gam(ViolentCrimesPerPop ~ s(Population, spar = param_val[i]) +
            s(PercentageBlack, spar = param_val[i]) + s(PercentageWhite,
            spar = param_val[i]) + s(PercentageAsian, spar = param_val[i]) +
            s(PercentageHispanic, spar = param_val[i]) + s(PercentageUrban,
            spar = param_val[i]) + s(MedIncome, spar = param_val[i]),
            data = train[folds != j, ])

        # Make prediction on fold 'j'
        pred = predict(model_1, train[folds == j, ])

        # Compute R^2 for predicted values
        cv_rsq[i] = cv_rsq[i] + rsq(train[folds == j, ]$ViolentCrimesPerPop,
            pred)
    }
    # Average R^2 across k folds
    cv_rsq[i] = cv_rsq[i]/k
}
```

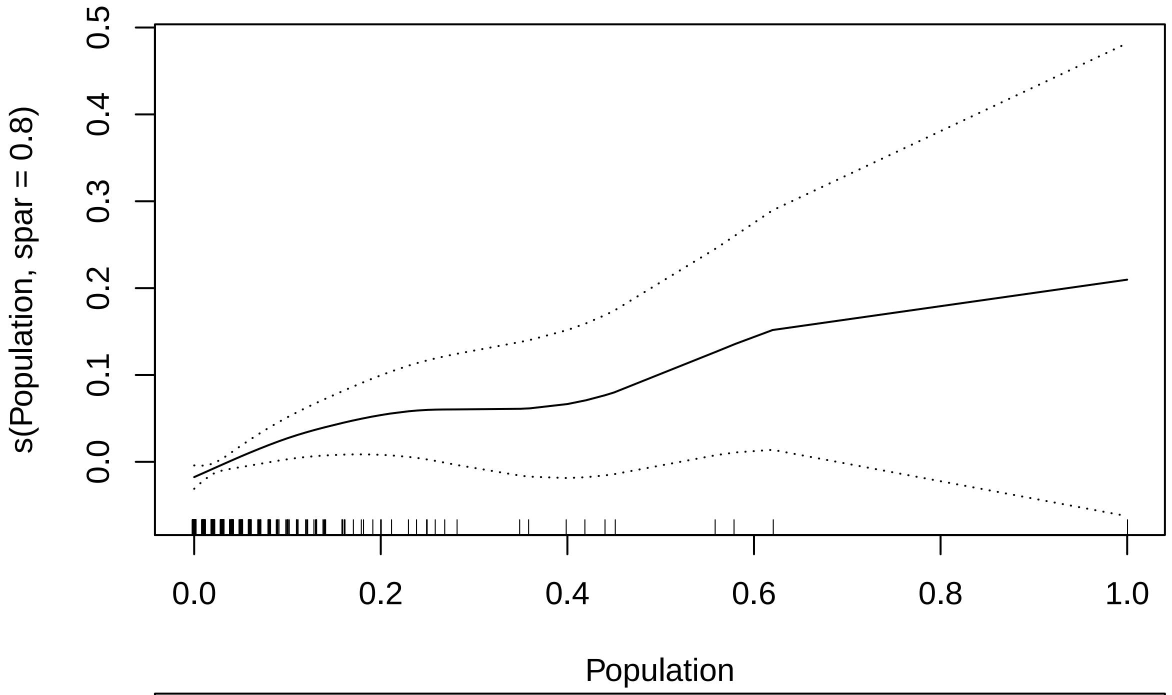
```
plot(param_val, cv_rsq)
```

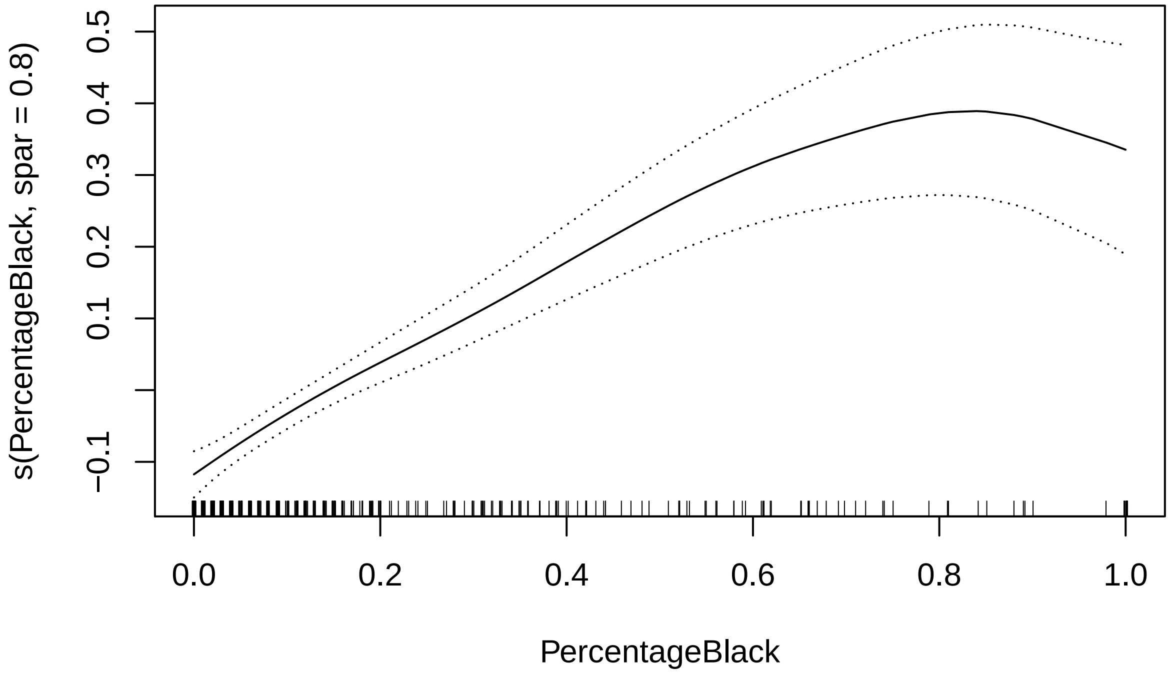


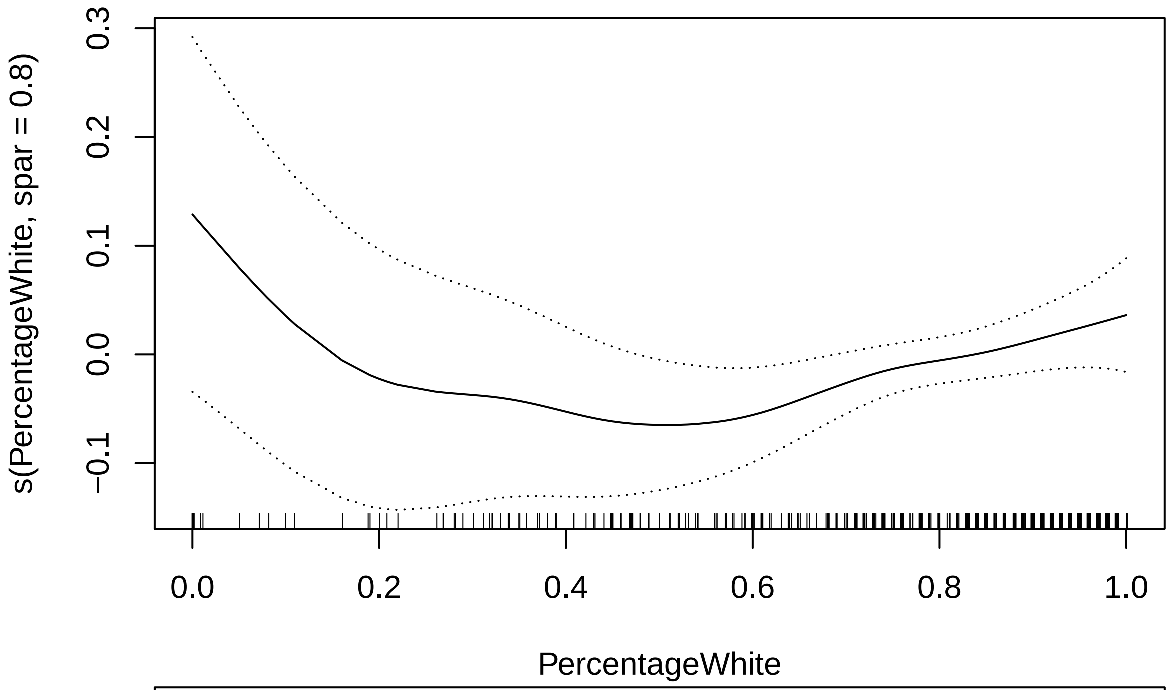
Based on cross validation an optimum spar of 0.8 can be selected as this results in the highest testing R2. This value of span can be used to fit the GAM.

```
gam_0 = gam(ViolentCrimesPerPop ~ 1, data = crime_train)

gam_model = gam(ViolentCrimesPerPop ~ s(Population, spar = 0.8) +
    s(PercentageBlack, spar = 0.8) + s(PercentageWhite, spar = 0.8) +
    s(PercentageAsian, spar = 0.8) + s(PercentageHispanic, spar = 0.8) +
    s(PercentageUrban, spar = 0.8) + s(MedIncome, spar = 0.8),
    data = crime_train)

crime_accuracy <- rbind(crime_accuracy, data.frame(models = "Gam model",
    train.r2 = rsq(crime_train$ViolentCrimesPerPop, predict(gam_model,
        newdata = crime_train)), test.r2 = rsq(crime_test$ViolentCrimesPerPop,
        predict(gam_model, newdata = crime_test))))
```

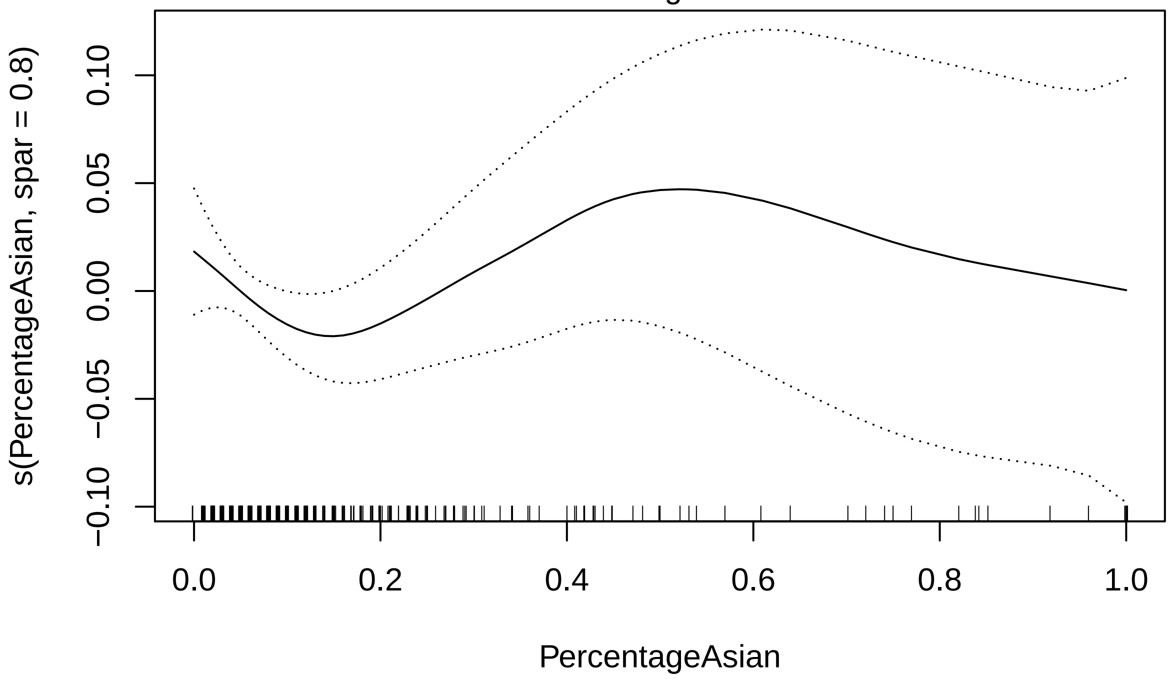
```
plot(gam_model, se = TRUE)
```



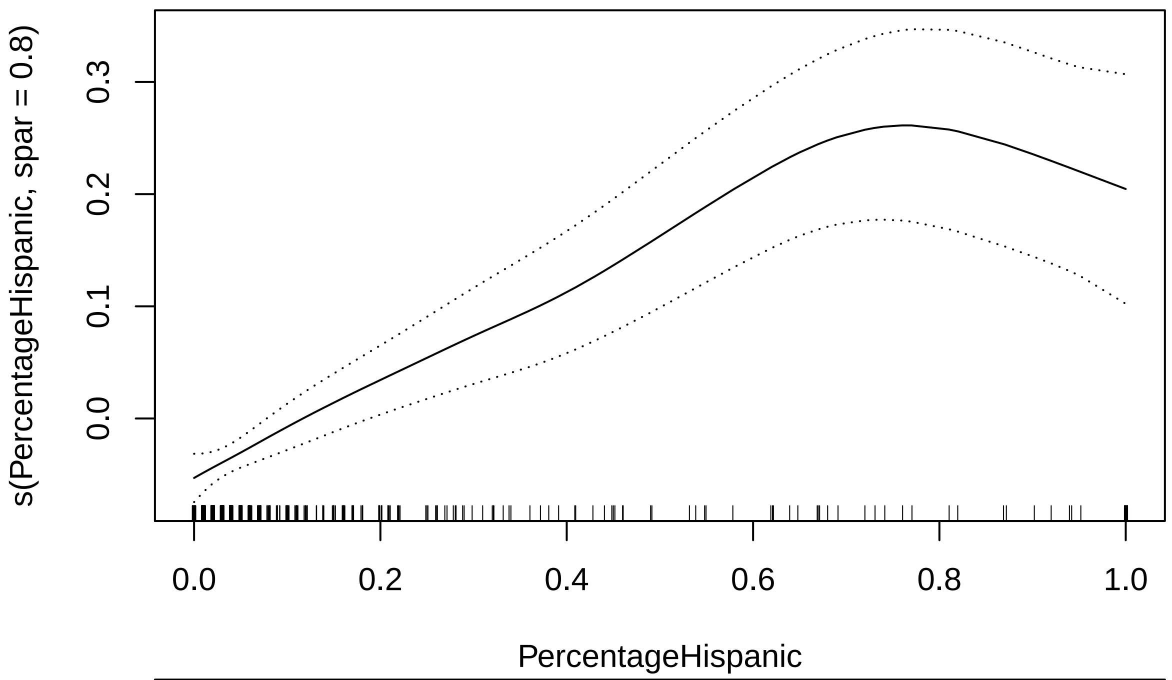


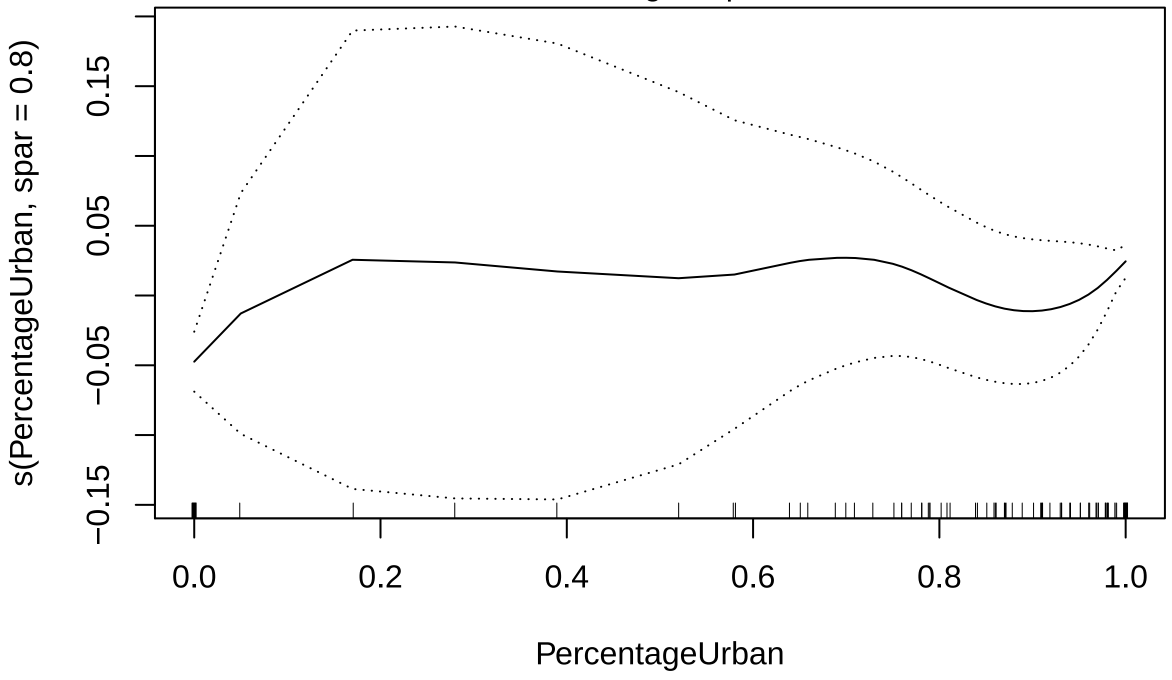


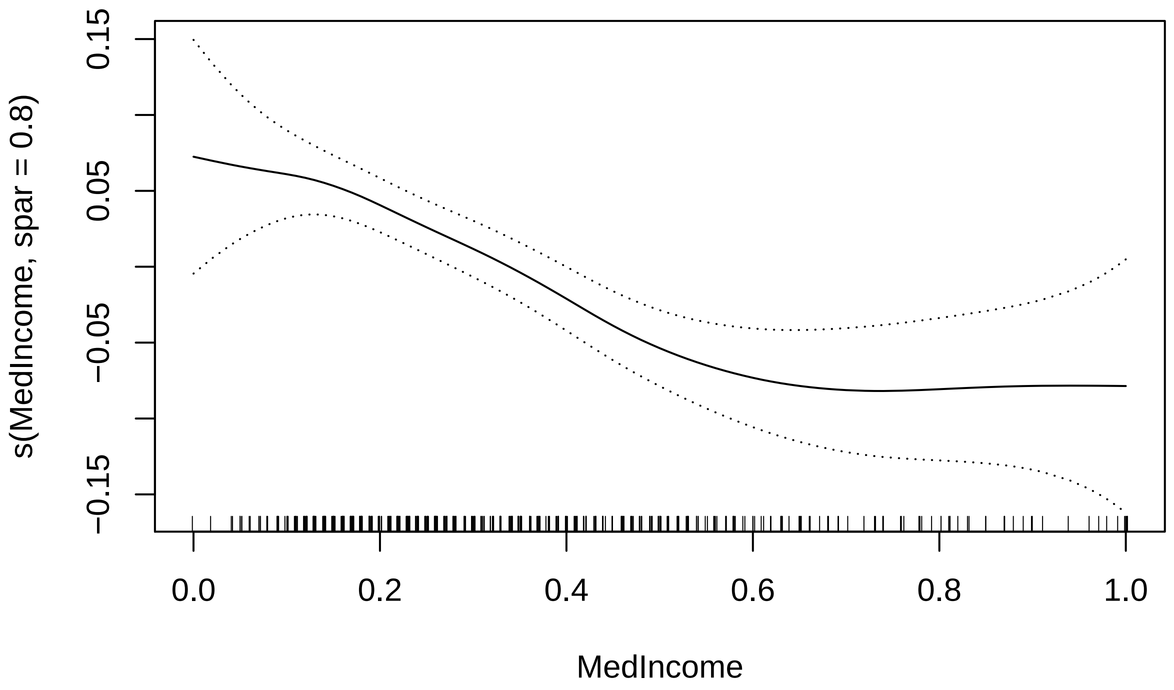
s(PercentageWhite, spar = 0.8)
0.3
0.2
0.1
0.0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
PercentageWhite



s(PercentageAsian, spar = 0.8)
0.10
0.05
0.00
−0.05
−0.10
0.0
0.2
0.4
0.6
0.8
1.0
PercentageAsian


s(PercentageHispanic, spar = 0.8)
0.0
0.1
0.2
0.3
0.0
0.2
0.4
0.6
0.8
1.0
PercentageHispanic


s(PercentageUrban, spar = 0.8)
−0.15
−0.05
0.05
0.15
0.0
0.2
0.4
0.6
0.8
1.0
PercentageUrban



```
gam_model$coefficients
```

```
##                      (Intercept)        s(Population, spar = 0.8) 
##                       0.05636462                       0.26707549 
##    s(PercentageBlack, spar = 0.8)    s(PercentageWhite, spar = 0.8) 
##                       0.57950849                       0.05541054 
##    s(PercentageAsian, spar = 0.8) s(PercentageHispanic, spar = 0.8) 
##                       0.01671072                       0.32929122 
##    s(PercentageUrban, spar = 0.8)         s(MedIncome, spar = 0.8) 
##                       0.06904004                      -0.21524599
```

For most of the predictors, the error curves diverge as the predictor value increases, however for Percentage-White and PercentageUrban this is not the case. The smallest error bars are seen for the PercentageBlack and PercentageHispanic predictors. Considering the coefficients, the PercentageBlack predictor has the highest predictive power followed by PercentageHispanic and Population. This is in accordance with the error associated with each coefficient.

Likelihood ratio to compare GAM with linear regression model

```
print(anova(lin_model, gam_model, test = "Chi"))
```

```
## Analysis of Variance Table
## 
## Model 1: ViolentCrimesPerPop ~ Population + PercentageBlack + PercentageWhite + 
##     PercentageAsian + PercentageHispanic + PercentageUrban + 
##     MedIncome
## Model 2: ViolentCrimesPerPop ~ s(Population, spar = 0.8) + s(PercentageBlack, 
##     spar = 0.8) + s(PercentageWhite, spar = 0.8) + s(PercentageAsian, 
##     spar = 0.8) + s(PercentageHispanic, spar = 0.8) + s(PercentageUrban, 
##     spar = 0.8) + s(MedIncome, spar = 0.8)
##   Res.Df     RSS     Df Sum of Sq Pr(>Chi)
## 1 490.00 10.1458
## 2 466.68  9.1439 23.324    1.0019 0.000754 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Considering the likelihood of the linear model to the GAM model, it can be seen that the GAM model is better than the linear model with a significance of 0.001.

```r
gam_model_refined = gam(ViolentCrimesPerPop ~ s(Population, spar = 0.8) +
    s(PercentageBlack, spar = 0.8) + s(PercentageWhite, spar = 0.8) +
    s(PercentageHispanic, spar = 0.8) + s(MedIncome, spar = 0.8),
    data = crime_train)
print(anova(gam_model_refined, gam_model, test = "Chi"))
```

```
## Analysis of Deviance Table
##
## Model 1: ViolentCrimesPerPop ~ s(Population, spar = 0.8) + s(PercentageBlack,
##     spar = 0.8) + s(PercentageWhite, spar = 0.8) + s(PercentageHispanic,
##     spar = 0.8) + s(MedIncome, spar = 0.8)
## Model 2: ViolentCrimesPerPop ~ s(Population, spar = 0.8) + s(PercentageBlack,
##     spar = 0.8) + s(PercentageWhite, spar = 0.8) + s(PercentageAsian,
##     spar = 0.8) + s(PercentageHispanic, spar = 0.8) + s(PercentageUrban,
##     spar = 0.8) + s(MedIncome, spar = 0.8)
##   Resid. Df Resid. Dev     Df Deviance Pr(>Chi)
## 1    475.87     9.6441
## 2    466.68     9.1439 9.1893   0.5002 0.002742 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Considering the likelihood of the refined GAM to the original GAM model, the more complex model is seen to be more significant. It can be seen that the original GAM model is better than the refined GAM model with a significance of 0.001.

Part 2c: Including Interaction Terms:

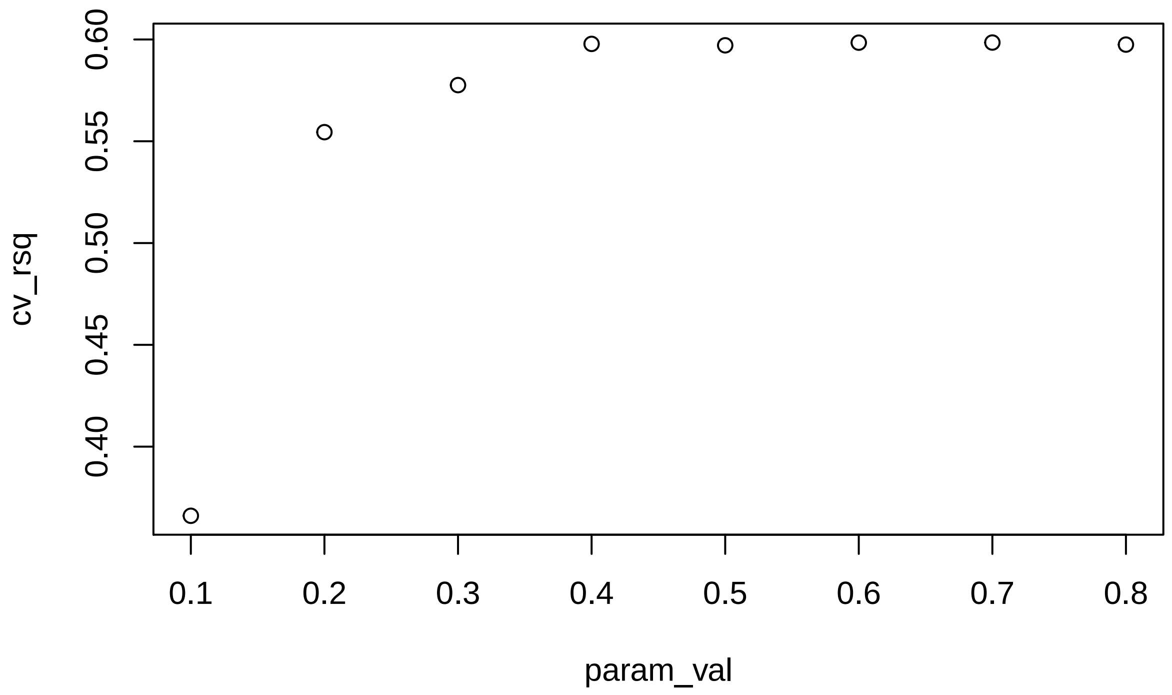
Term 1: Local regression basis function invovling population, percentage urban and medincome

```r
# Function for k-fold cross-validation to tune span parameter
# in GAM
train = crime_train
param_val <- seq(0.1, 0.8, by = 0.1)
k = 5

# Input: Training data frame: 'train', Output: Vector of R^2
# values for the provided parameters: 'cv_rsq'

num_param = length(param_val)  # Number of parameters set.seed(109) # Set seed for random number genera

# Divide training set into k folds by sampling uniformly at
# random # folds[s] has the fold index for train instance 's'
folds = sample(1:k, nrow(train), replace = TRUE)

cv_rsq = rep(0, num_param)  # Store cross-validated R^2 for different parameter values

# Iterate over parameter values
for (i in 1:num_param) {
    # Iterate over folds to compute R^2 for parameter
    for (j in 1:k) {
        # Fit model on all folds other than 'j' with parameter value
        # param_val[i] model.loess = loess(PickupCount ~ TimeMin,
        # span = param_val[i], data = train[folds!=j, ],


        model <- gam(ViolentCrimesPerPop ~ s(Population, spar = 0.8) +
            s(PercentageBlack, spar = 0.8) + s(PercentageWhite,
            spar = 0.8) + s(PercentageAsian, spar = 0.8) + s(PercentageHispanic,
            spar = 0.8) + s(PercentageUrban, spar = 0.8) + s(MedIncome,
            spar = 0.8) + lo(Population, PercentageUrban, MedIncome,
            span = param_val[i]) + lo(MedIncome, PercentageBlack,
            span = param_val[i]), data = train[folds != j, ])


        # Make prediction on fold 'j'
        pred = predict(model, train[folds == j, ])

        # Compute R^2 for predicted values
        cv_rsq[i] = cv_rsq[i] + rsq(train[folds == j, ]$ViolentCrimesPerPop,
            pred)
    }
    # Average R^2 across k folds
    cv_rsq[i] = cv_rsq[i]/k
}
# Return cross-validated R^2 values
```

Interaction Terms in the GAM model

Use cross validation to tune the span parameter of the local regression interaction term

```
plot(param_val, cv_rsq)
```



A span of 0.4 has the highest testing R2 from cross validation and this span value was used to train the final GAM model with interaction terms.

```r
gam_model_refined_tuned = gam(ViolentCrimesPerPop ~ s(Population,
    spar = 0.8) + s(PercentageBlack, spar = 0.8) + s(PercentageWhite,
    spar = 0.8) + s(PercentageHispanic, spar = 0.8) + s(MedIncome,
    spar = 0.8) + lo(Population, PercentageUrban, MedIncome,
    span = 0.4) + lo(MedIncome, PercentageBlack, span = 0.4),
    data = crime_train)
```

```r
print(anova(gam_model_refined_tuned, gam_model, test = "Chi"))
```

```
## Analysis of Deviance Table
##
## Model 1: ViolentCrimesPerPop ~ s(Population, spar = 0.8) + s(PercentageBlack,
##     spar = 0.8) + s(PercentageWhite, spar = 0.8) + s(PercentageHispanic,
##     spar = 0.8) + s(MedIncome, spar = 0.8) + lo(Population, PercentageUrban,
##     MedIncome, span = 0.4) + lo(MedIncome, PercentageBlack, span = 0.4)
## Model 2: ViolentCrimesPerPop ~ s(Population, spar = 0.8) + s(PercentageBlack,
##     spar = 0.8) + s(PercentageWhite, spar = 0.8) + s(PercentageAsian,
##     spar = 0.8) + s(PercentageHispanic, spar = 0.8) + s(PercentageUrban,
##     spar = 0.8) + s(MedIncome, spar = 0.8)
##   Resid. Df Resid. Dev      Df Deviance  Pr(>Chi)
## 1    458.03     8.5665
## 2    466.68     9.1439 -8.6437 -0.57741 0.0002407 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The more complex model (in this case the GAM model with interaction) is more significant than the simpler model (without interaction terms) with a significance of 0.001.

Compare GAM models:

The testing R2 of the the original GAM model and the refined GAM model can be compared:

```
cat("Original GAM model", rsq(crime_test$ViolentCrimesPerPop,
    predict(gam_model, newdata = crime_test)))
```

```
## Original GAM model 0.5751515
```

```
cat("GAM model without PercentageUrban and Percentage Asian terms",
    rsq(crime_test$ViolentCrimesPerPop, predict(gam_model_refined,
        newdata = crime_test)))
```

```
## GAM model without PercentageUrban and Percentage Asian terms 0.5738545
```

```
cat("GAM model without PercentageUrban and Percentage Asian terms and with interaction terms",
    rsq(crime_test$ViolentCrimesPerPop, predict(gam_model_refined_tuned,
        newdata = crime_test)))
```

```
## GAM model without PercentageUrban and Percentage Asian terms and with interaction terms 0.5937981
```

The final GAM model with the incusion of the interaction terms has a higher testing R2 compared to the previous two GAM models. However there is no significant improvement in the predictive accuracy (increase of only 0.02 in the testing R2). Compared to the linear and polynomial models, the GAM model has not had a significant improvement in the predictive accuracy on the test set. This suggests that a more complex model may not be justified for modeling the incidence of violent crimes in this data set.