Homework 1: Smoothers and loess models

Harvard CS 109B, Spring 2017

Tim Hagmann

February 16, 2017

Contents

Problem 1: Predicting Taxi Pickups

In this problem, the task is to build a regression model that can predict the number of taxi pickups in New York city at any given time of the day. The data set is provided in the files `dataset_1_train.txt` and `dataset_1_test.txt`, and contains details of taxi pickups in the month of Jan 2015. The first column contains the time of the day in minutes, the second column contains information about the day of the week (1 through 7, with 1 denoting Monday) and the last column contains the number of pickups observed at that time.

Visualize the data and check if the pattern of taxi pickups makes intuitive sense.

Initialize

In the following code chunk all the necessary setup for the modelling environment is done.

```
## Options
options(scipen = 10)                          # Disable scientific notation
update_package <- FALSE                       # Use old status of packages
```

```
## Init files (always execute, eta: 10s)
source("scripts/01_init.R")                    # Helper functions to load packages
source("scripts/02_packages.R")                # Load all necessary packages

## Set Java path to C:/Program Files (x86)/Java/jre7
source("scripts/03_functions.R")               # Load project specific functions
```

Load the data

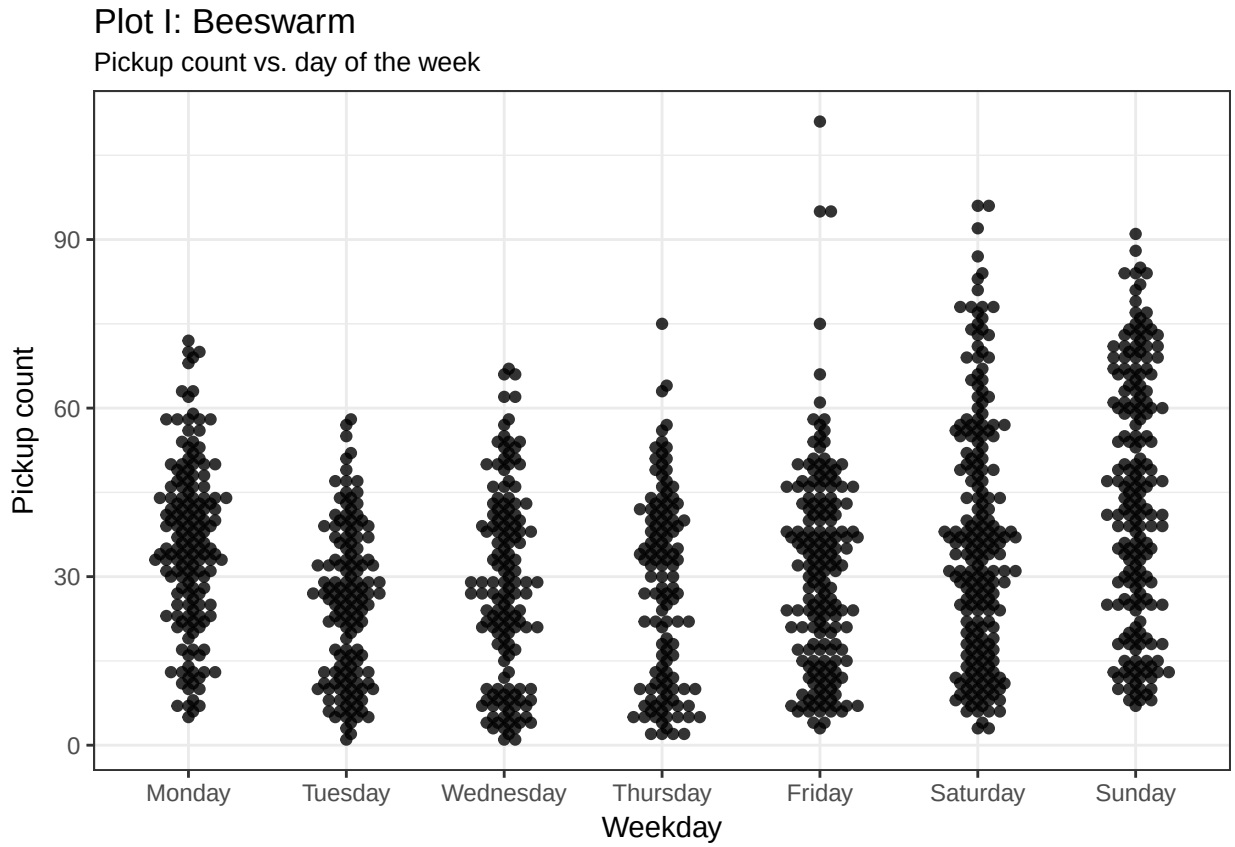
```
## Read data
df_train1 <- read_csv("data/dataset_1_train.txt")
df_test1 <- read_csv("data/dataset_1_test.txt")
```

Prepare data for visualization

```
## Transform day numbers to characters
weekdays <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
              "Sunday")
df_train1$DayOfWeek <- factor(df_train1$DayOfWeek, labels=weekdays,
                              ordered=TRUE)

rm(weekdays)

## Transform to time in hours
df_train1$TimeHours <- round((df_train1$TimeMin / 60), 0)
```

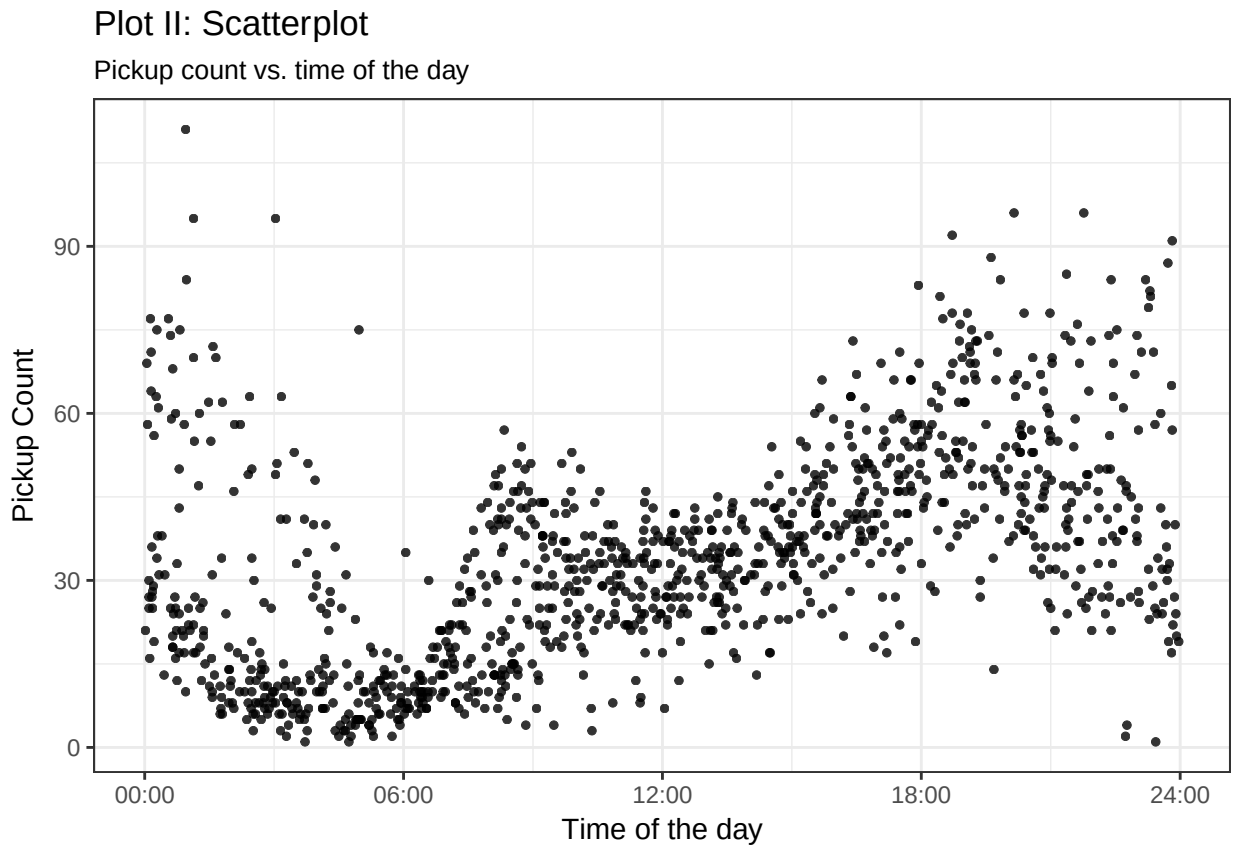
Visualise the data:

```
## Beeswarm plot
ggplot(df_train1, aes(DayOfWeek, PickupCount)) +
  labs(title="Plot I: Beeswarm",
       subtitle="Pickup count vs. day of the week") +
  geom_beeswarm(color="black", alpha = 0.8) +
  xlab("Weekday") +
  ylab("Pickup count") +
  theme_bw()
```



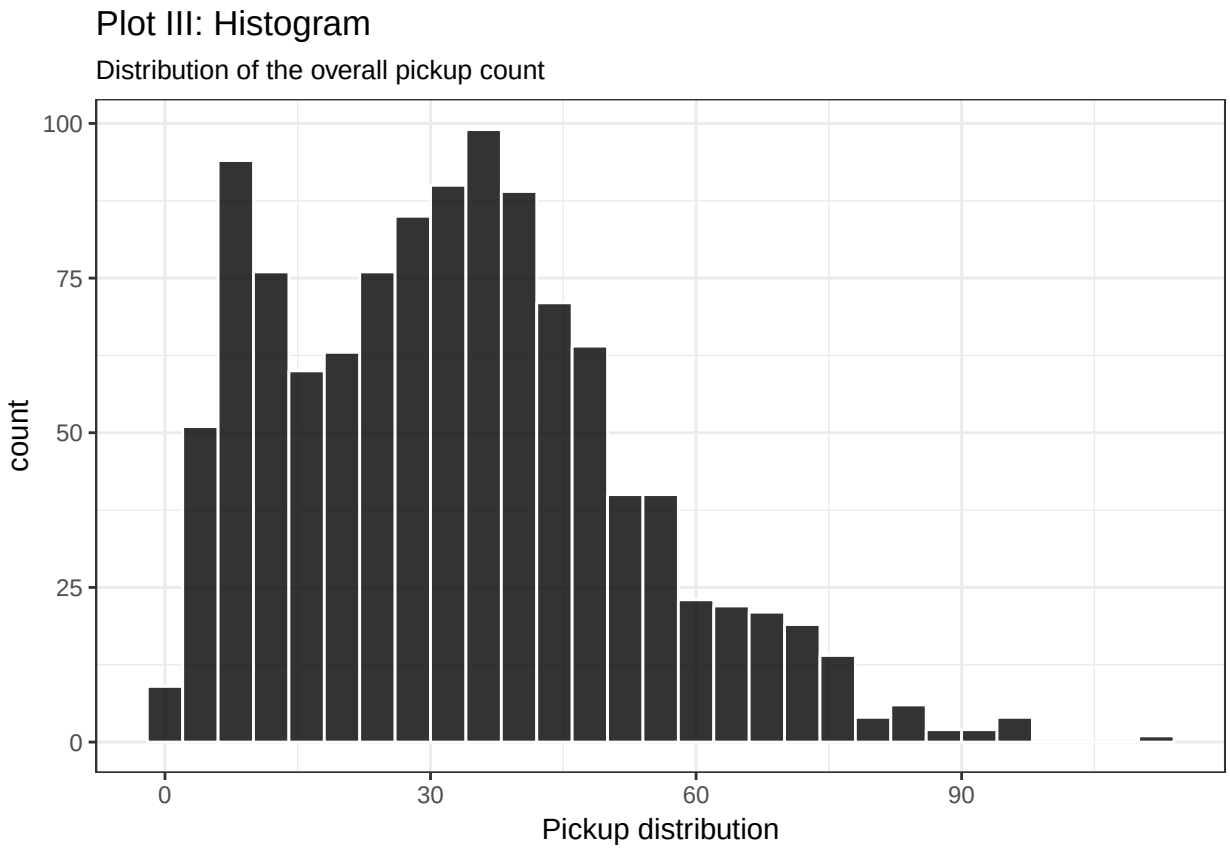
The above beeswarm plot shows (similar to a boxplot but visualizing all the individual data), that the hightest pick-up count is reached on Friday's. However this appears to be only the case for a few observation. While on Saturday's and Sunday's the pickup count is more evenly distributed reaching higher levels. Overall, this means that the weekends show more pick-ups than the week. The lowest pickup rates are reached on Tuesday. This pattern makes intiutivly sense. Next we're going to look at the distribution of the pickup rate during the day.

```
## Scatterplot
ggplot(df_train1, aes(TimeMin, PickupCount)) +
  geom_point(stroke=0, alpha=0.8) +
  theme_bw() +
  labs(title="Plot II: Scatterplot",
       subtitle="Pickup count vs. time of the day") +
  scale_x_continuous(breaks=c(0, 360, 720, 1080, 1440),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
  ylab(label="Pickup Count") +
  xlab("Time of the day")
```



The above plot show the distribution of the pick-ups over the time of the day. The pattern appears to be non-linear with different break points. The highest pickup rates are reached during the evening an early morning. The lowest rate is at around 06:00. This behaviour also makes sense.

```
## Histogram
ggplot(df_train1, aes(PickupCount)) +
  labs(title="Plot III: Histogram",
       subtitle="Distribution of the overall pickup count") +
  geom_histogram(binwidth = 4, fill="black", colour="white", alpha=0.8) +
  xlab("Pickup distribution") +
  theme_bw()
```



The above plot shows the overall pick-up distribution. There appear to be two centers at around 10 and 35. Furthermore, the distribution is skewed to the right, i.e., only a few observations reach a level above 80, which is to be expected.

Part 1a: Explore different regression models

You are required to fit a regression model that uses the time of the day (in minutes) as a predictor and predicts the average number of taxi pick ups at that time. For this part of the question, you can ignore the `DayOfWeek` predictor. Fit the following models on the training set and compare the R^2 of the fitted models on the test set. Include plots of the fitted models for each method.

1. Regression models with different basis functions:

- Simple polynomials with degrees 5, 10, and 25
- Cubic B-splines with the knots chosen by visual inspection of the data.
- Natural cubic splines with the degree of freedom chosen by cross-validation on the training set

Simple polynomials with degrees 5, 10, and 25

```
lm_poly5 <- lm(PickupCount ~ poly(TimeMin, degree=5, raw=TRUE), data=df_train1)
lm_poly10 <- lm(PickupCount ~ poly(TimeMin, degree=10, raw=TRUE), data=df_train1)
lm_poly25 <- lm(PickupCount ~ poly(TimeMin, degree=25, raw=TRUE), data=df_train1)
```

```
df_pred <- data.frame(df_train1, pred_lm5=predict(lm_poly5),
                      pred_lm10=predict(lm_poly10), pred_lm25=predict(lm_poly25))
```

Table1: Polynomials with degree 5

```
pander(summary(lm_poly5), covariate.labels=c("x", paste0("x^", 2:5), "intercept"),
       add.significance.stars=TRUE, style='rmarkdown', caption="",
       split.table=Inf, digits=2, justify='center')
```

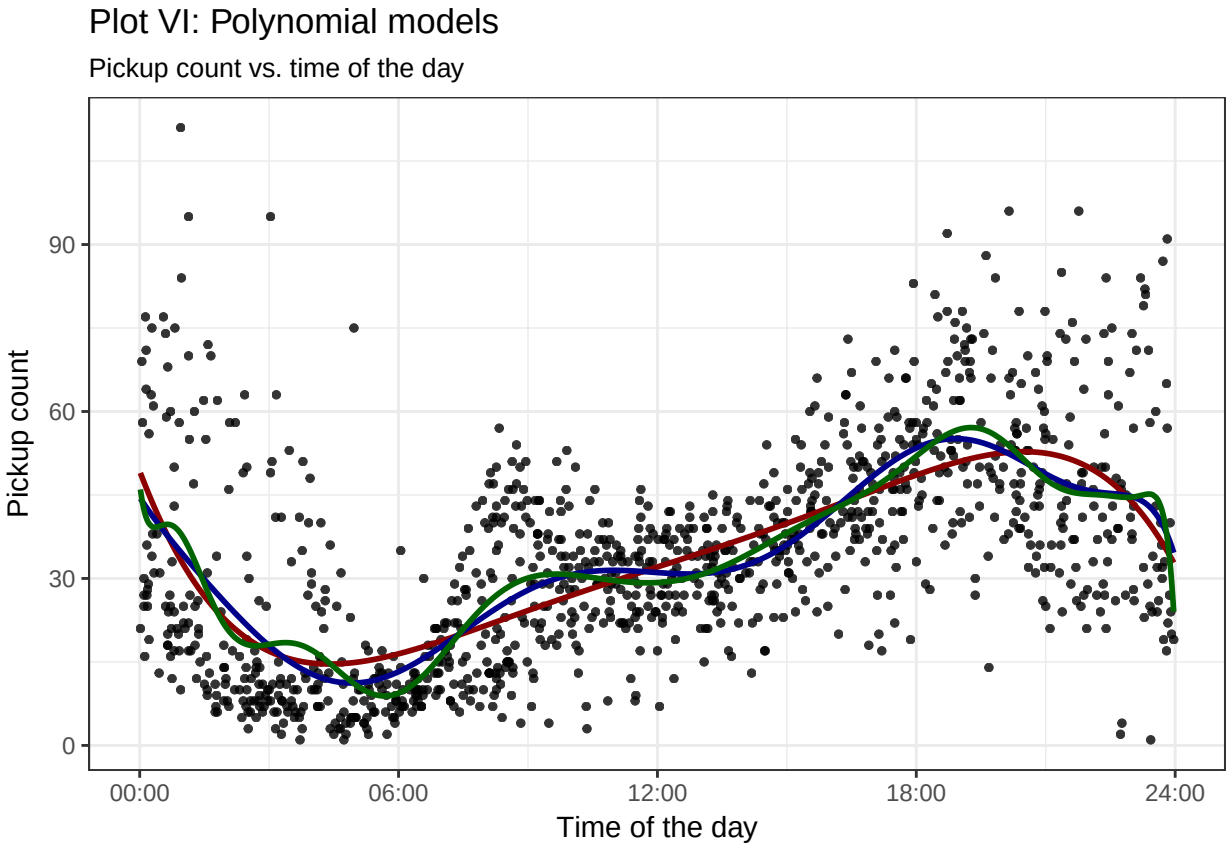
	Estimate	Std. Error	t value	Pr(>	t
x	-0.34	0.036	-9.6	4.6e-21	* * *
x^2	0.0012	0.00015	7.6	0.000000000000071	* * *
x^3	-0.0000017	0.00000027	-6.2	0.00000000066	* * *
x^4	0.0000000012	0.00000000021	5.7	0.000000019	* * *
x^5	-0.00000000000032	0.000000000000059	-5.5	0.000000048	* * *
intercept	49	2.5	20	5.9e-75	* * *

Observations	Residual Std. Error	R^2	Adjusted R^2
1125	14.73	0.424	0.4214

The above table shows, that the polynomial terms can be analysed on a statistical basis, i.e., looking at the p-values. In the above table all the values are significantly different from zero. We're not going to look at the other tables but instead are going to visualize the graphs instead.

Visualize

```
ggplot(df_pred, aes(x=TimeMin, y=PickupCount)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_lm5), colour="darkred", size=1) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_lm10), colour = "darkblue", size=1) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_lm25), colour = "darkgreen", size=1) +
  labs(title="Plot VI: Polynomial models",
  subtitle="Pickup count vs. time of the day") +
  scale_x_continuous(breaks=c(0, 360, 720, 1080, 1440),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
  ylab(label="Pickup count") +
  xlab("Time of the day") +
  theme_bw()
```



The three models (poly5 = red, poly10 = blue and poly 25 = green) appear to be able to capture some of the behaviour of the data. While the poly 5 model appears to be a bit too linear and the poly 25 model seems to overfit the data.

Cubic B-splines with the knots chosen by visual inspection of the data.

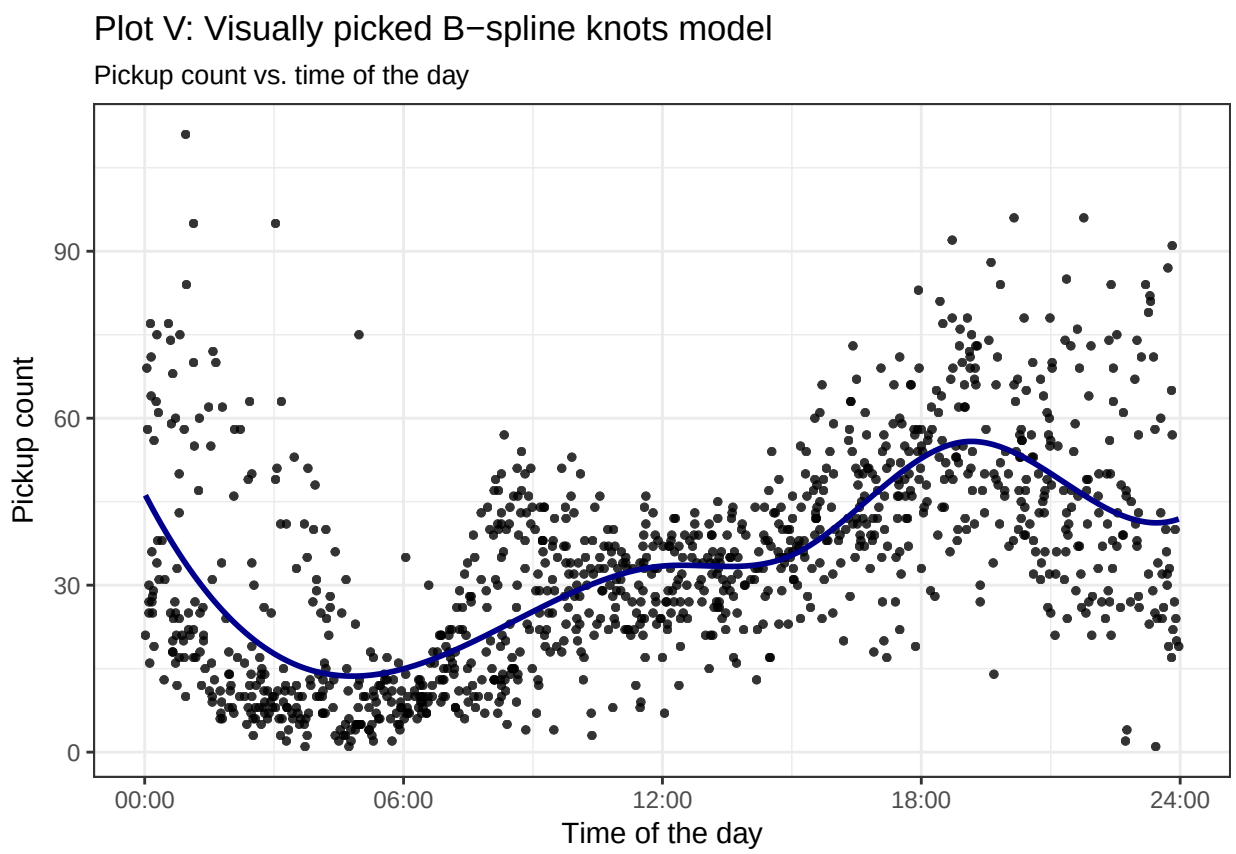
In the Plot II and VI, we can see that there appear to be shifts at around 04:00, 12:00, 15:00, 19:00. this can be used to fit B-splines to the data.

```
knots <- c(4*60, 12*60, 15*60, 19*60)
sp_cubic <- lm(PickupCount ~ bs(TimeMin, knots=knots), data=df_train1)

df_pred['pred_sp_cubic'] <- predict(sp_cubic)
rm(knots)
```

Visualize

```
ggplot(df_pred, aes(x=TimeMin, y=PickupCount)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_sp_cubic), colour="darkblue", size=1) +
  labs(title="Plot V: Visually picked B-spline knots model",
  subtitle="Pickup count vs. time of the day") +
  scale_x_continuous(breaks=c(0, 360, 720, 1080, 1440),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
  ylab(label="Pickup count") +
```

```
  xlab("Time of the day") +
  theme_bw()
```



The above model doesn't appear to be too bad. Nevertheless, it doesn't pick up on all the different non-linear behaviour in the data. That is why in the next step were going to fit the cubic splines according to cross-validation on the training set.

Natural cubic splines (df chosen by cv)

Experience shows, that a 5 or 10 fold cross-validation gives reasonable results. In this case the data is randomly split into 10 fold.

10 fold crossvalidation

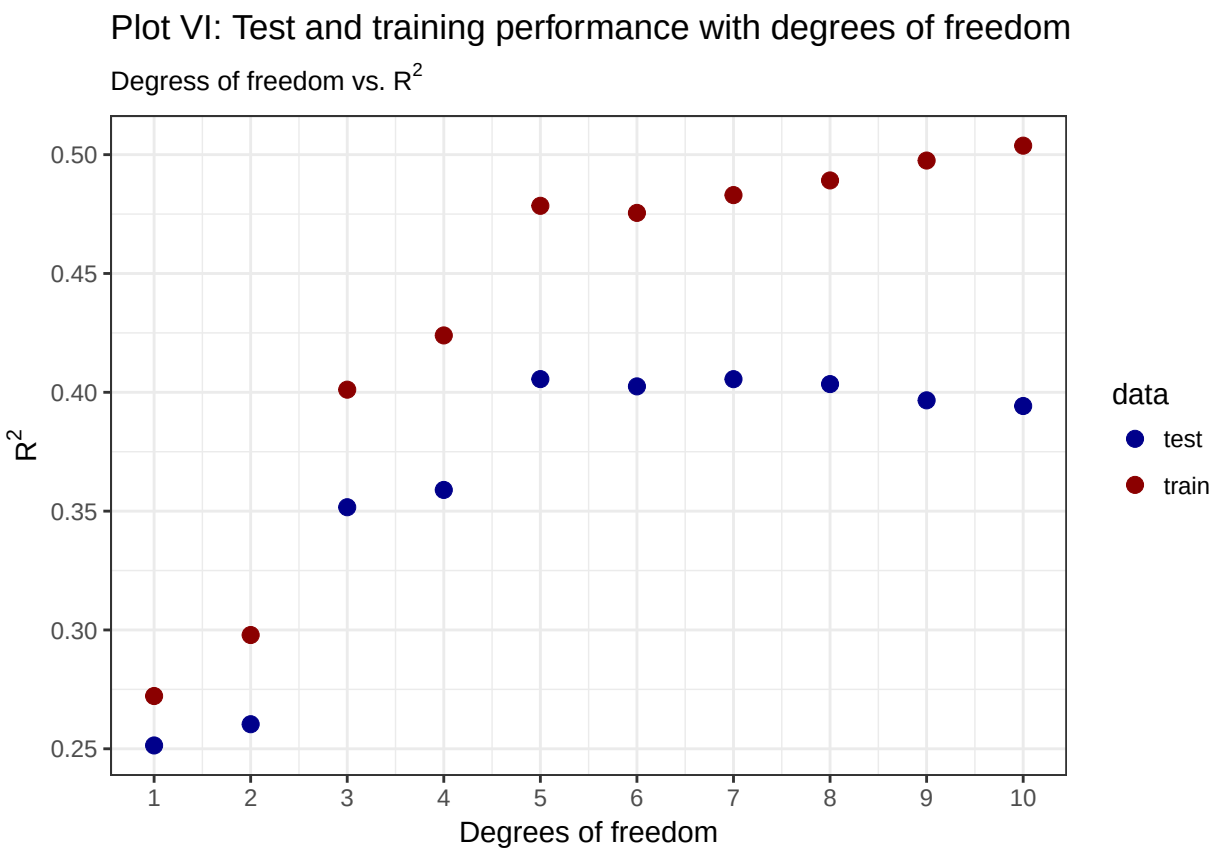
```
# Options
k <- 10          # Number of partitions
dfs <- 1:10      # degrees of freedom
set.seed(123)    # Set random seed

# Create k splits
df_train1$partitions <- cut(sample(1:nrow(df_train1),
                           nrow(df_train1)),
                           breaks=k,
                           labels=FALSE)
```

```
# Function for model performance
model_performance <- function(df, train, test){
    mod <- lm(PickupCount ~ ns(TimeMin, df=df), data=train)

    out <- c(train_r2 = calc_rsq(train$PickupCount, predict(mod, newdata=train)),
             test_r2 = calc_rsq(test$PickupCount, predict(mod, newdata=test)))

    out
}

# Looping trough the splits
perform_10fold <- lapply(unique(df_train1$partitions),
                         function(split){
                           train <- df_train1$partitions == split

                           mt_train <- df_train1[train, ]
                           mt_test <- df_train1[-train, ]

                           data.frame(t(sapply(dfs, model_performance,
                                               train=mt_train,
                                               test=mt_test)),
                                               df=dfs)
                           })

# Get k numbers of modeling statistics
perform_10fold <- do.call(rbind, perform_10fold)

# Aggregating the data
perform_10fold <- lapply(split(perform_10fold, perform_10fold$df),
                         function(x) {
                           data.frame(rsquare = c(mean(x$train_r2),
                                                  mean(x$test_r2)),
                                               data=c("train", "test"),
                                               dfs=unique(x$df))
                         })

# Bind the results
perform_10fold <- do.call(rbind, perform_10fold)
rm(k, dfs)
```

Visualize degress of freedom vs. R^2

Next we can visualize the R^2 performance of the train and test data according to the degrees of freedom.

```
ggplot(perform_10fold, aes(x=dfs, y=rsquare, color=data)) +
  geom_point(stroke=0, size=3) +
  labs(title="Plot VI: Test and training performance with degrees of freedom",
  subtitle=expression(paste("Degress of freedom vs. ", R^{2}))) +
  ylab(label=expression(R^{2})) +
  xlab("Degrees of freedom") +
  scale_x_continuous(breaks=1:10,
                     labels=1:10) +
  scale_color_manual(values=c("darkblue", "darkred")) +
  theme_bw()
```



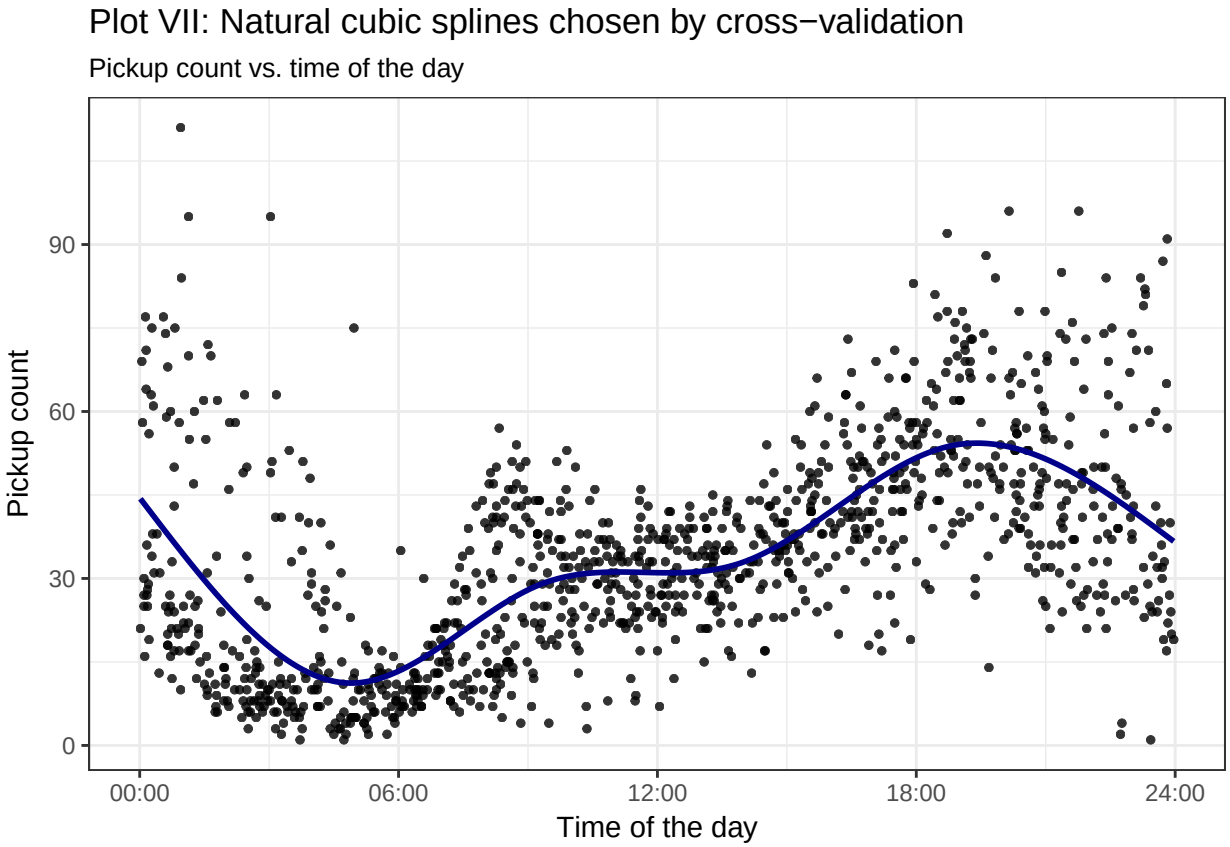
The 'best' model for the test set appears to be reached at 5 degrees of freedom. After 5 degrees of freedom the test perfomance starts to level of, i.e., start to overfit.

Modeling with 5 degrees of freedom

```
ns_fit_5 <- lm(PickupCount ~ ns(TimeMin, df=5), data=df_train1)
df_pred['pred_ns_fit'] <- predict(ns_fit_5, newdata=df_train1)
```

Visualize

```
ggplot(df_pred, aes(x=TimeMin, y=PickupCount)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_ns_fit), colour="darkblue", size=1) +
  labs(title="Plot VII: Natural cubic splines chosen by cross-validation",
  subtitle="Pickup count vs. time of the day") +
  scale_x_continuous(breaks=c(0, 360, 720, 1080, 1440),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
  ylab(label="Pickup count") +
  xlab("Time of the day") +
  theme_bw()
```



The model doesn't appear to be too bad but it is not too distinguistable from the model in plot V.

2. Smoothing spline model (smoothness chosen by cv)

Next we can model a smooting spline model with the smoothness parameters chosen by cross-validation.

Smoothing Splines

Note: For smoothing splines, `R` provides an internal cross-validation feature: this can be used by leaving the `spar` attribute in `smooth.spline` unspecified; you may set the `cv` attribute to choose between leave-one-out cross-validation and generalized cross-validation.

Setting 'cv=TRUE' leads to the use of "ordinary" cross validation which uses a leave-one-out type strategy for validation. However, this isn't useful when one has duplicated x values. That is why CV is set to false which leads to "generalized" cross-validation.

```
set.seed(123)     # set random seed
spline_fit <- smooth.spline(df_train1$TimeMin, df_train1$PickupCount, cv=FALSE)
spline_fit['spar']
```

```
## $spar
## [1] 0.7632497
```

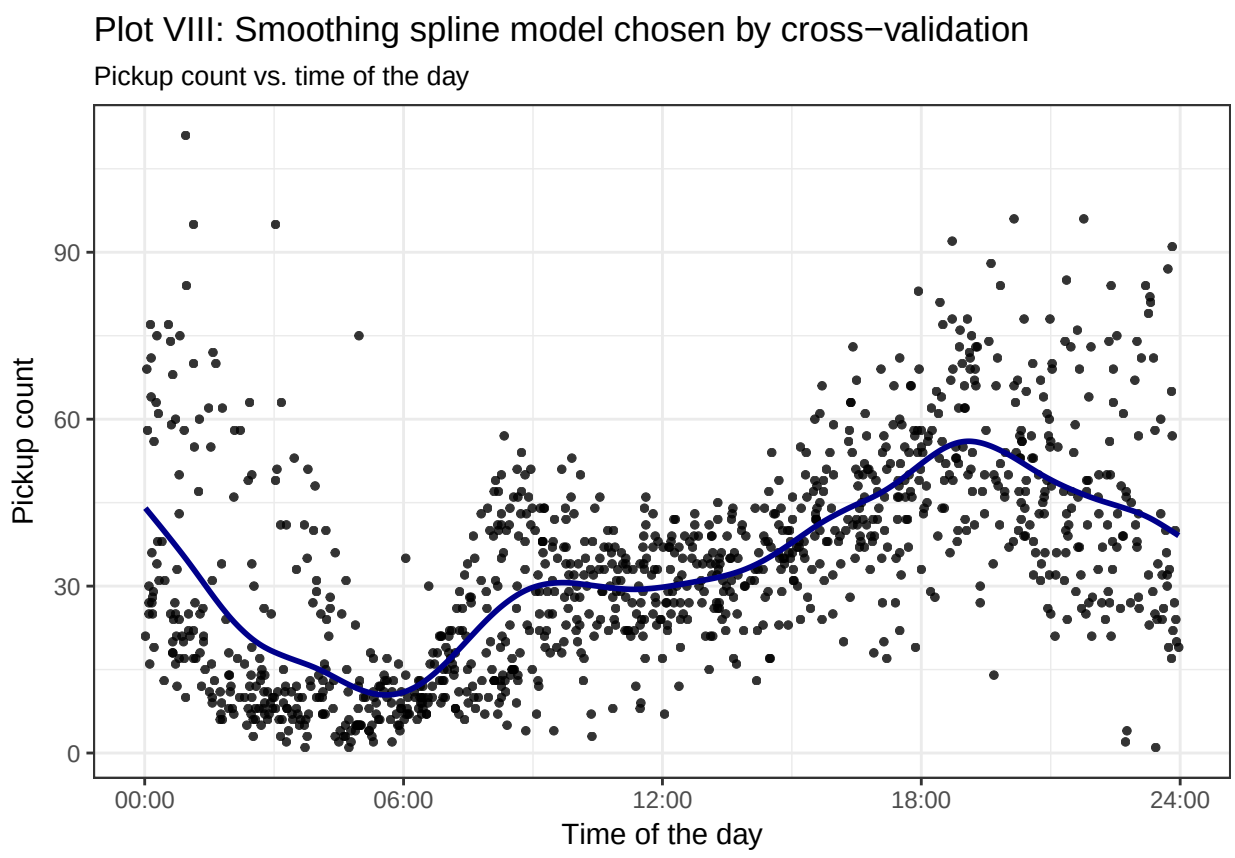
Modeling

Next we plug in the spar value (0.763) into the smooth.spline model.

```
sp_smooth = smooth.spline(df_train1$TimeMin, df_train1$PickupCount, spar = spline_fit['spar'])

# prediction
df_pred['pred_sp_smooth'] <- predict(sp_smooth, df_train1$TimeMin)$y
```

Visualize

```
ggplot(df_pred, aes(x=TimeMin, y=PickupCount)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_sp_smooth), colour="darkblue", size=1) +
  labs(title="Plot VIII: Smoothing spline model chosen by cross-validation",
  subtitle="Pickup count vs. time of the day") +
  scale_x_continuous(breaks=c(0, 360, 720, 1080, 1440),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
  ylab(label="Pickup count") +
  xlab("Time of the day") +
  theme_bw()
```



The above model appears to pick the nonlinear behavior better than the other models seen before.

3. Locally-weighted regression model (span chosen by cv)

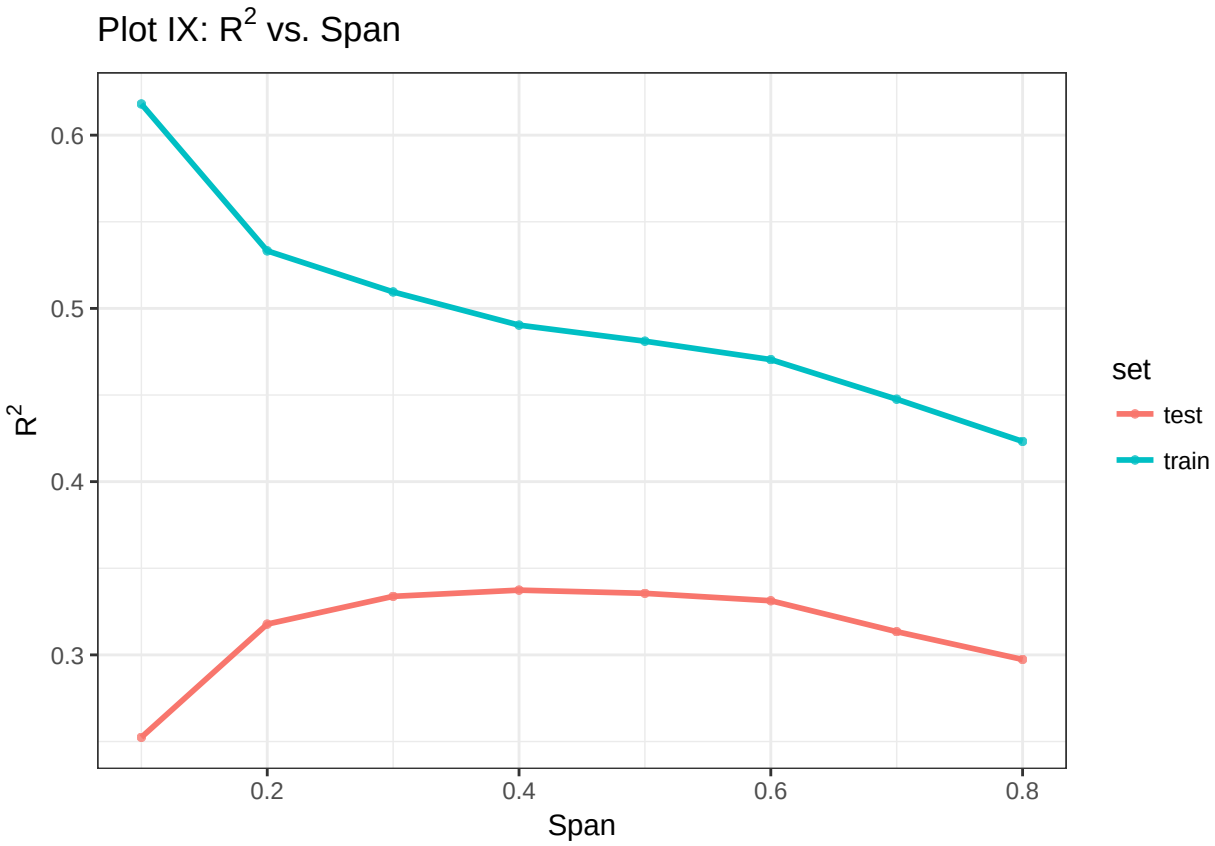
Next were modeling a localy weighted regression model.

Calculate 10 fold CV

```
set.seed(123)
perform_10fold <- loess_10foldcv(df_train1)
```

Visualize

```
ggplot(perform_10fold, aes(x=span, y=x, color=set)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(size=1) +
  labs(title=expression(paste("Plot IX: ", R^{2}, " vs. Span"))) +
  ylab(label=expression(R^{2})) +
  xlab("Span") +
  theme_bw()
```



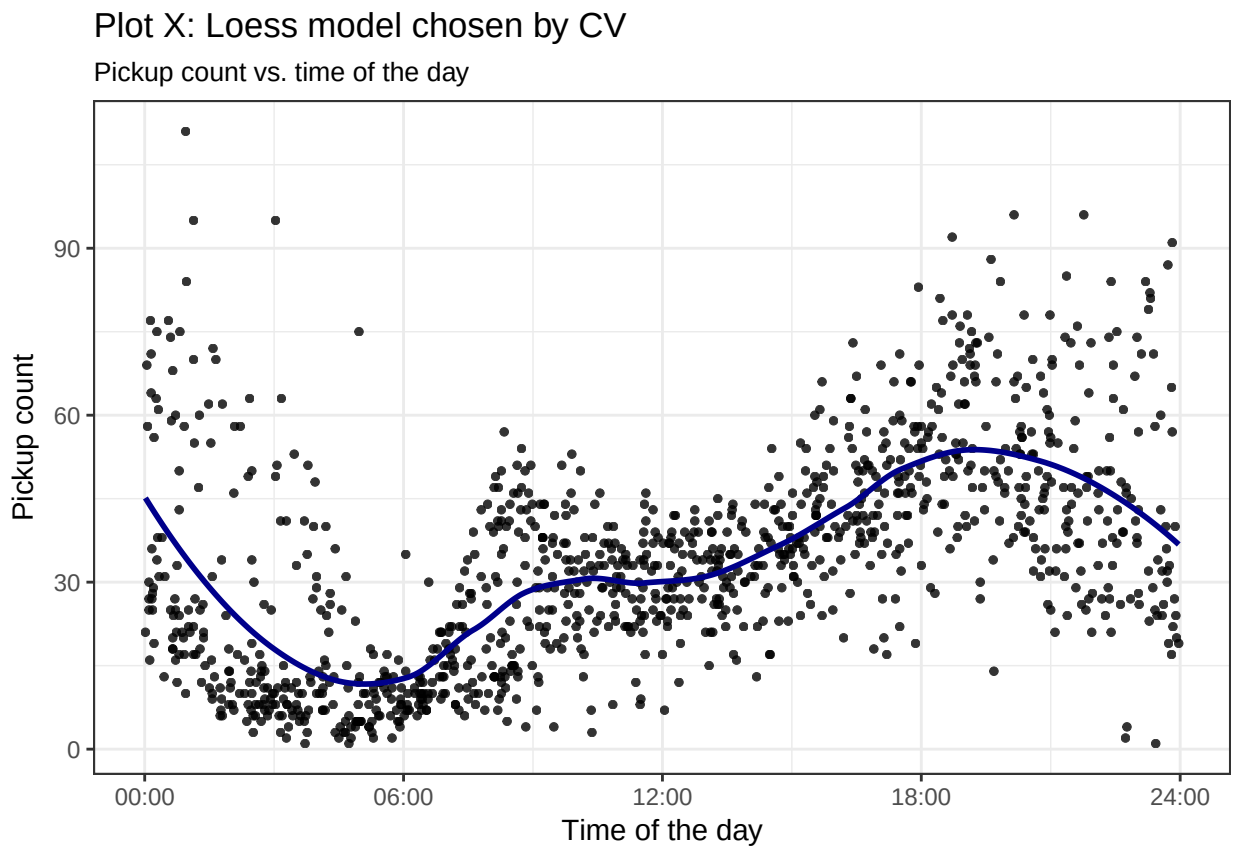
The above plot shows, that at a span of 0.4 the highest R^2 value is reached.

Calculate the loess model

```
loess_model <- loess(PickupCount ~ TimeMin, span=0.4, data=df_train1)

# Prediction
df_pred['pred_loess_model'] <- predict(loess_model, df_train1$TimeMin)
```

Visualize

```
ggplot(df_pred, aes(x=TimeMin, y=PickupCount)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(aes(x=df_pred$TimeMin, y=pred_loess_model), colour="darkblue", size=1) +
  labs(title="Plot X: Loess model chosen by CV",
  subtitle="Pickup count vs. time of the day") +
  scale_x_continuous(breaks=c(0, 360, 720, 1080, 1440),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
  ylab(label="Pickup count") +
  xlab("Time of the day") +
  theme_bw()
```



The above model appears to fit the data really well. However, it is very wighly and might be overfitting the data.

Calculate the accuracy

In each case, analyze the effect of the relevant tuning parameters on the training and test R^2, and give explanations for what you observe.

```
# Polynomial models
acc_lm5 <- data.frame('model'='poly (5th degree)',
                      'rsq_train'=calc_rsq(df_train1$PickupCount,
                                           predict(lm_poly5, newdata=df_train1)),
                      'rsq_test'=calc_rsq(df_test1$PickupCount,
                                          predict(lm_poly5, newdata=df_test1)))
acc_lm10 <- data.frame('model'='poly (10th degree)',
                      'rsq_train'=calc_rsq(df_train1$PickupCount,
                                           predict(lm_poly10, newdata=df_train1)),
                      'rsq_test'=calc_rsq(df_test1$PickupCount,
                                          predict(lm_poly10, newdata=df_test1)))
acc_lm25 <- data.frame('model'='poly (25th degree)',
                      'rsq_train'=calc_rsq(df_train1$PickupCount,
                                           predict(lm_poly25, newdata=df_train1)),
                      'rsq_test'=calc_rsq(df_test1$PickupCount,
                                          predict(lm_poly25, newdata=df_test1)))
# Cubic splines
acc_sp_cubic <- data.frame('model'='Cubic splines model ',
                      'rsq_train'=calc_rsq(df_train1$PickupCount,
                                           predict(sp_cubic, newdata=df_train1)),
                      'rsq_test'=calc_rsq(df_test1$PickupCount,
                                          predict(sp_cubic, newdata=df_test1)))
# Natural cubic splines models
acc_ns_8 <- data.frame('model'='Natural cubic splines model choosen by CV',
                      'rsq_train'=calc_rsq(df_train1$PickupCount,
                                           predict(ns_fit_5, newdata=df_train1)),
                      'rsq_test'=calc_rsq(df_test1$PickupCount,
                                          predict(ns_fit_5, newdata=df_test1)))

# Natural cubic splines models (!Carefull, the calc_rsq is different)
pred_train <- predict(sp_smooth, df_train1$TimeMin)$y
pred_test <- predict(sp_smooth, df_test1$TimeMin)$y
acc_sp_smooth <- data.frame('model'='Smoothing spline model chosen by CV',
                      'rsq_train'=calc_rsq_splines(pred_train, df_train1$PickupCount),
                      'rsq_test'=calc_rsq_splines(pred_test, df_test1$PickupCount))

# Loess model
pred_train <- predict(loess_model, newdata=df_train1$TimeMin)
pred_test <- predict(loess_model, newdata=df_test1$TimeMin)
acc_loess_model <- data.frame('model'='Loess model chosen by CV',
                      'rsq_train'=calc_rsq(df_train1$PickupCount, pred_train),
                      'rsq_test'=calc_rsq(df_test1$PickupCount, pred_test))

accuracy <- rbind(acc_lm5, acc_lm10, acc_lm25, acc_sp_cubic, acc_ns_8,
                  acc_sp_smooth, acc_loess_model)
rm(acc_lm5, acc_lm10, acc_lm25, acc_sp_cubic, acc_ns_8, acc_sp_smooth,
   acc_loess_model, pred_train, pred_test)
```

Look at the accuracy data

```
pander(accuracy, justify='center')
```

model	rsq_train	rsq_test
poly (5th degree)	0.424	0.3856
poly (10th degree)	0.4484	0.4132
poly (25th degree)	0.4598	0.4224
Cubic splines model	0.4385	0.4004
Natural cubic splines model choosen by CV	0.4465	0.4111
Smoothing spline model chosen by CV	0.457	0.4255
Loess model chosen by CV	0.4485	0.4156

Is there a reason you would prefer one of these methods over the other?

It appears, that the poly accuracy for the poly 25th degree has rank-deficient data. A lower degree polynomial or a different model might be a better choice as the results could otherwise be misleading. Taking that into concideration, Loess model as well as the smoothing spline model chosen by cross validation appear to be the best two model.

Part 1b: Adapting to weekends

Does the pattern of taxi pickups differ over the days of the week? Are the patterns on weekends different from those on weekdays? If so, we might benefit from using a different regression model for weekdays and weekends. Use the `DayOfWeek` predictor to split the training and test sets into two parts, one for weekdays and one for weekends, and fit a separate model for each training subset using locally-weighted regression. Do the models yield a higher R^2 on the corresponding test subsets compared to the (loess) model fitted previously? (You may use the loess model fitted in 1A (with the span parameter chosen by CV) to make predictions on both the weekday and weekend test sets, and compute its R^2 on each set separately, you may also use the same best_span calculated in 1A)

Answer: As already seen above in the beeswarm plot, the the weekend appears to indeed differ from the weekdays. In order to check for this hypothis further, let us look at the two different timeseries for the weekdays vs. the weekends.

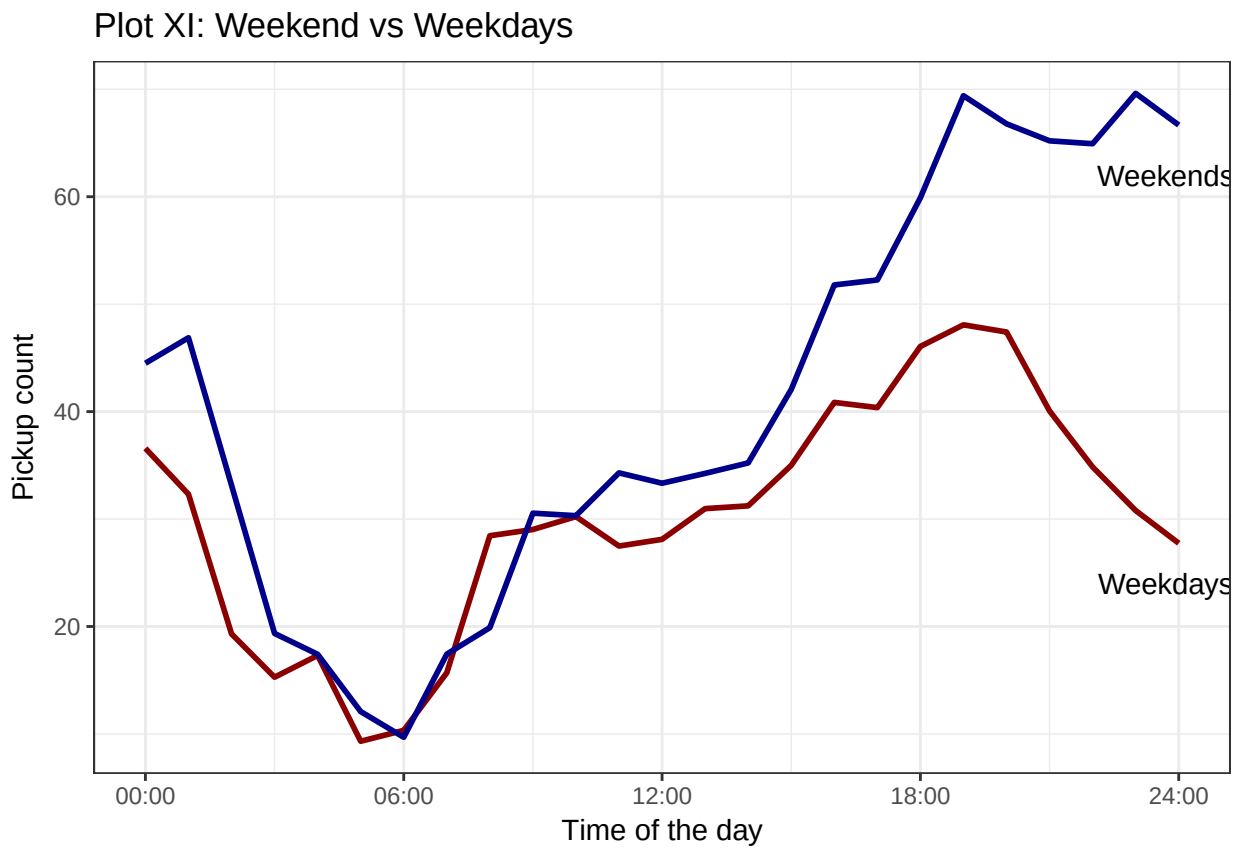
Split the data into weekday and weekend

```
# Select
df_weekday_train = df_train1[df_train1$DayOfWeek != "Saturday" &
                               df_train1$DayOfWeek != "Sunday" , ]
df_weekend_train = df_train1[df_train1$DayOfWeek == "Saturday" |
                               df_train1$DayOfWeek == "Sunday", ]
df_weekday_test = df_train1[df_train1$DayOfWeek != "Saturday" &
                              df_train1$DayOfWeek != "Sunday" , ]
df_weekend_test = df_train1[df_train1$DayOfWeek == "Saturday" |
                              df_train1$DayOfWeek == "Sunday", ]

# Aggregate
```

```
df_weekday_agg <- aggregate(df_weekday_train$PickupCount ~ df_weekday_train$TimeHours,
                            FUN = "mean")
names(df_weekday_agg) <- c("TimeHours", "PickupCount")
df_weekend_agg <- aggregate(df_weekend_train$PickupCount ~ df_weekend_train$TimeHours,
                            FUN = "mean")
names(df_weekend_agg) <- c("TimeHours", "PickupCount")
```

Visualize

```
ggplot(df_weekday_agg, aes(x=TimeHours, y=PickupCount)) +
  geom_line(aes(x=df_weekday_agg$TimeHours, y=df_weekday_agg$PickupCount),
            colour="darkred", size=1) +
  geom_line(aes(x=df_weekend_agg$TimeHours, y=df_weekend_agg$PickupCount),
            colour="darkblue", size=1) +
    labs(title="Plot XI: Weekend vs Weekdays") +
  scale_x_continuous(breaks=c(0, 6, 12, 18, 24),
                     labels=c("00:00", "06:00", "12:00", "18:00", "24:00")) +
    ylab(label="Pickup count") +
  xlab("Time of the day") +
  annotate("text", x=23.7, y=24, label="Weekdays") +
  annotate("text", x=23.7, y=62, label="Weekends") +
  theme_bw()
```



The above plot shows the different timeseries of weekend vs weekdays. As can be seen, the pickup times do indeed differ. This is especially the case during the evening.

Fit Loess on weekday and weekend data

Modeling weekday data

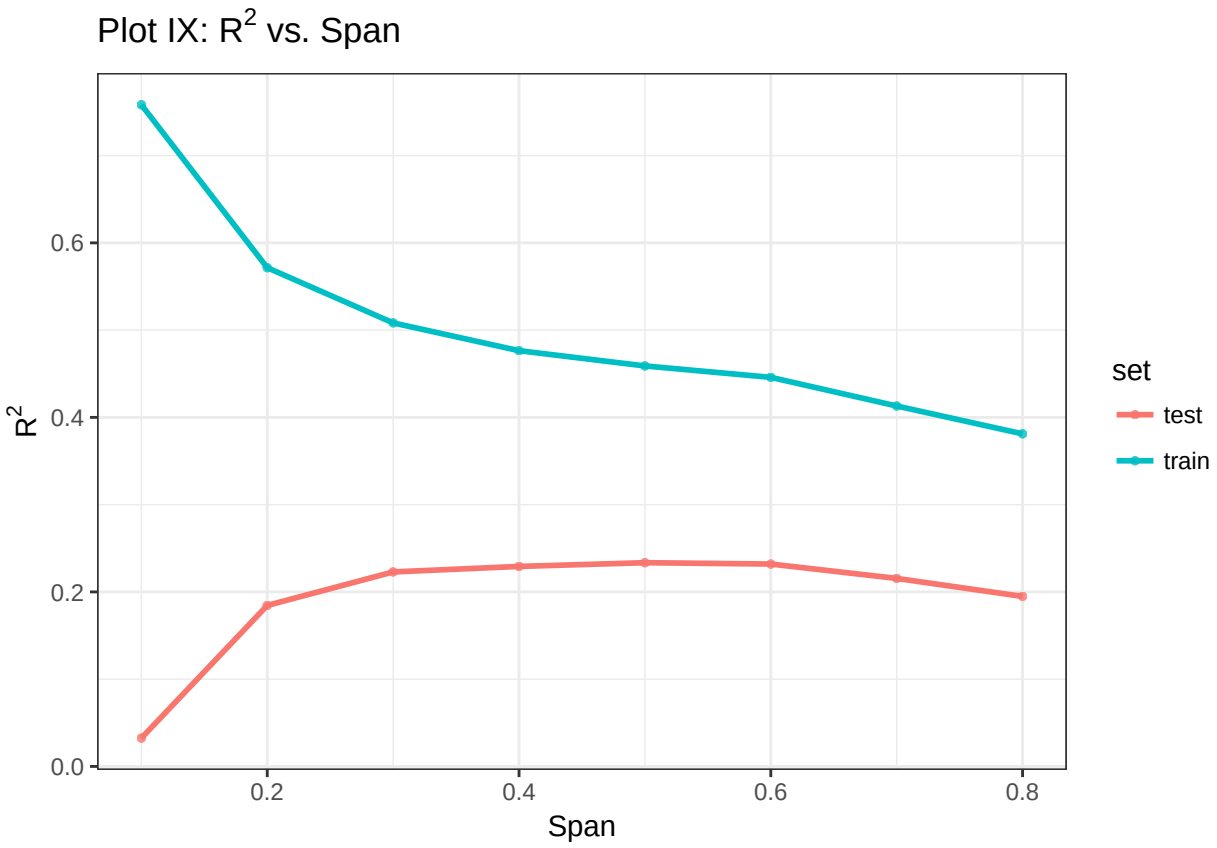
Calculate the loess model

Calculate 10 fold CV

```
set.seed(123)
perform_weekday_loess <- loess_10foldcv(df_weekday_train)
```

Visualize

```
ggplot(perform_weekday_loess, aes(x=span, y=x, color=set)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(size=1) +
  labs(title=expression(paste("Plot IX: ", R^{2}, " vs. Span"))) +
  ylab(label=expression(R^{2})) +
  xlab("Span") +
  theme_bw()
```



The best R^2 is reached with a span of 0.6

Modeling the data

```
set.seed(123)
weekend_model = loess(PickupCount ~ TimeMin, span=0.6, data=df_weekend_train)
weekday_model = loess(PickupCount ~ TimeMin, span=0.6, data=df_weekday_train)
```

Comparison of R^2 values

Comparison of R^2 values of the new models on the separated test sets compared to model fit on the entire test set

```
acc_all <- data.frame('model'='Overall Loess (Test)',
                      'test_rsquared'=calc_rsq(df_test1$PickupCount,
                                               predict(loess_model, newdata = df_test1)))
acc_weekday <- data.frame('model'='Weekday Loess (Test)',
                          'test_rsquared'=calc_rsq(df_weekday_test$PickupCount,
                                                   predict(weekday_model,
                                                           newdata = df_weekday_test)))
acc_weekend <- data.frame('model'='Weekend Loess (Test)',
                          'test_rsquared'=calc_rsq(df_weekend_test$PickupCount,
                                                   predict(weekend_model,
                                                           newdata = df_weekend_test)))

loess_accuracy <- rbind(acc_all, acc_weekday, acc_weekend)
rm(acc_all, acc_weekday, acc_weekend)
```

Get values

```
pander(loess_accuracy, ustify='center')
```

model	test_rsquared
Overall Loess (Test)	0.4156
Weekday Loess (Test)	0.3707
Weekend Loess (Test)	0.7137

The above table shows, that while the weekday model doesn't profit from split the weekend model performs much better.

Problem 2: Predicting Crime in the City

In this problem, the task is to build a model that can predict the per-capita crime rate in a given region of the US. The data set is provided in the files `dataset_2_train.txt` and `dataset_2_test.txt`. Each row corresponds to a region in the US: the first column contains the number of violent crimes per 100K population, and the remaining columns contain 8 attributes about the region. All numeric predictors are normalized into the range 0.00-1.00, and retain their distribution and skew (e.g. the population predictor has a mean value of 0.06 because most communities are small)

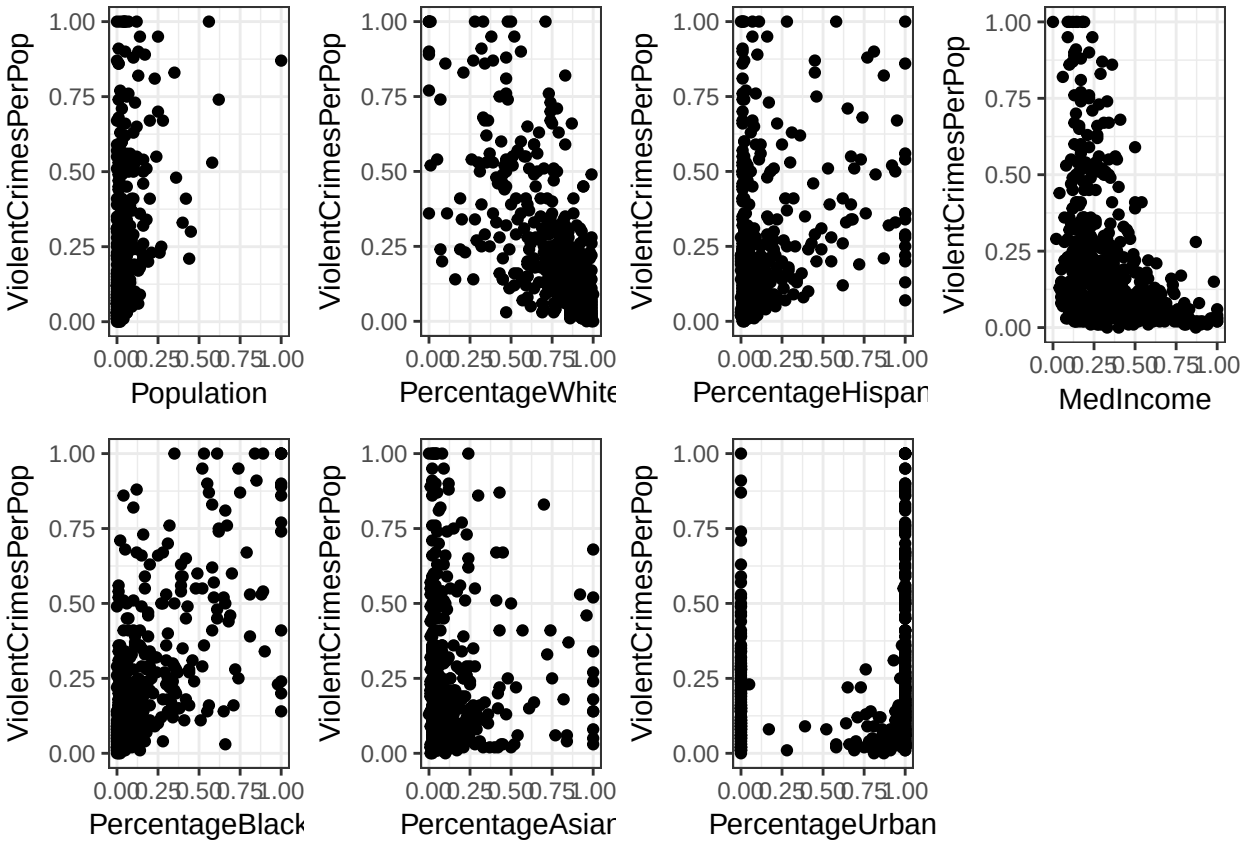
Examine the relationship between the crime rate and the individual predictors visually. Do some of the predictors have a non-linear relationship with the response variable, warranting the use of a non-linear regression model?

Load the data

```
# Clean workspace
rm(list = ls())

# Read data
df_train2 <- read_delim("data/dataset_2_train.txt", delim="\t")
df_test2 <- read_delim("data/dataset_2_test.txt", delim="\t")
```

Visualise the data:

```
# Beeswarm plot
g1 <- ggplot(df_train2, aes_string(x="Population", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g2 <- ggplot(df_train2, aes_string(x="PercentageBlack", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g3 <- ggplot(df_train2, aes_string(x="PercentageWhite", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g4 <- ggplot(df_train2, aes_string(x="PercentageAsian", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g5 <- ggplot(df_train2, aes_string(x="PercentageHispanic", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g6 <- ggplot(df_train2, aes_string(x="PercentageUrban", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g7 <- ggplot(df_train2, aes_string(x="MedIncome", y="ViolentCrimesPerPop")) +
              geom_point() +
              theme_bw()
g <- multiplot(g1, g2, g3, g4, g5, g6, g7, cols=4)
```




```
rm(g1, g2, g3, g4, g5, g6, g7)
```

Looking at the above plots shows, that the crime rate is not linearly related to the different predictors. That means they all have different effect on the y variable (response). Especially the Perentage Urban appears no be highly non-linear. Most variables are either 0% or 100%. Furthermore, the distribution of violent crimes is also heavily spread around the different ethnic communities. All of this leads to the conclusion that a non-linear model approach should be chosen.

Part 2a: Polynomial regression

Fit the following models on the training set and compare the R^2 score of the fitted models on the test set:

Modeling lm, poly2, poly3 and B-splines

- Linear regression
- Regression with polynomial basis functions of degree 2 (i.e. basis functions x, x^2 for each predictor x)
- Regression with polynomial basis functions of degree 3 (i.e. basis functions x, x^2, x^3 for each predictor x)
- Regression with B-splines basis function on each predictor with three degrees of freedom

```
# Modelling
fit_lm <- lm(ViolentCrimesPerPop ~ ., data=df_train2)

fit_poly2 <- 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=df_train2)

fit_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, raw=TRUE) +
                   poly(PercentageUrban, degree=3, raw=TRUE) +
                   poly(MedIncome, degree=3, raw=TRUE), data=df_train2)

fit_bs <- 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=df_train2)


# Bind together
crime_accuracy <- data.frame(t(sapply(list(fit_lm, fit_poly2, fit_poly_3,
                                           fit_bs), model_performance)))
crime_accuracy <- cbind(c("Linear model", "Poly 2 model", "Poly 3 model", "B-Spline model"),
                        crime_accuracy)
names(crime_accuracy) <- c("Models", "R2_Train", "R2_Test")
```

Accuracy table

```
pander(crime_accuracy, justify='center')
```

Models	R2_Train	R2_Test
Linear model	0.6185	0.5554
Poly 2 model	0.6328	0.5753
Poly 3 model	0.6437	0.5734
B-Spline model	0.6437	0.5734

The Poly 2 model has the best performance on the test set. This is followed by the poly 3 and B-Spline model. This is a bit surprising as one would expect that the poly3 and B-Spline model should be performing better.

Part 2b: Generalized Additive Model (GAM)

Do you see any advantage in fitting an additive regression model to this data compared to the above models?

Answer: Yes, due to the non-linear nature of the data a GAM model might be performing better. Furthermore, next to the inclusion of nonlinear effect the GAM also has the advantage that it offers an interpretable solution.

1. Fitting a GAM model

Fit a GAM model (Generalized additive model) to the training set, and compare the test R^2 of the fitted model to the above models. You may use a smoothing spline basis function on each predictor, with the same smoothing parameter for each basis function, tuned using cross-validation on the training set.

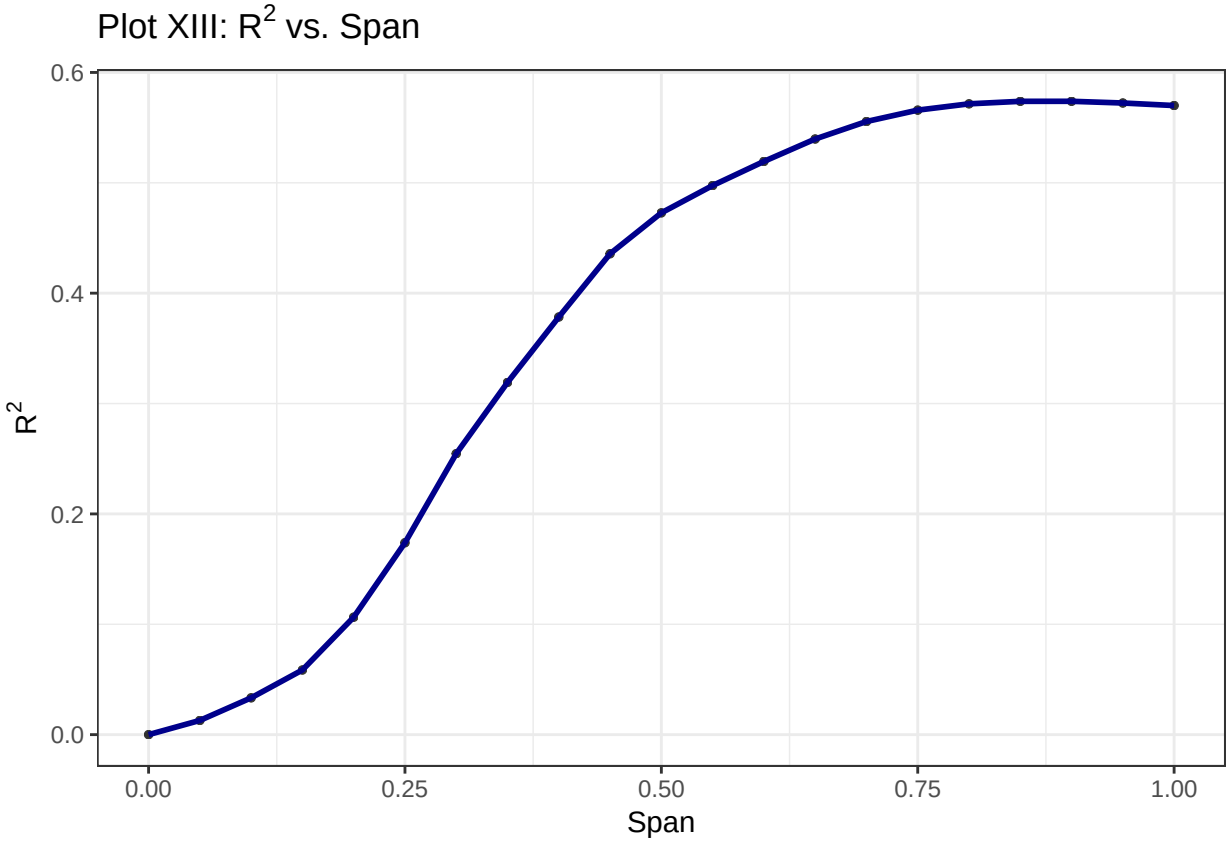
Crossvalidation

```
# Set parameter
set.seed(123)
train = df_train2
span_val  <- seq(0, 1, by = 0.05)
k <- 10
n <- length(span_val)

# Divide training set into k folds by sampling uniformly at random
folds <- sample(1:k, nrow(train), replace=TRUE)

cv_rsq <- rep(0., n) # Store cross-validated R^2 for different parameter values

# Loper over the parameter values
for(i in 1:n){

  # Loop over the folds
  for(j in 1:k){

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

    # Prediction
    pred = predict(model, train[folds == j, ])

    # Compute R^2
    cv_rsq[i] = cv_rsq[i] + calc_rsq(train[folds == j, ]$ViolentCrimesPerPop, pred)
  }

  # Average R^2 across k folds
  cv_rsq[i] = cv_rsq[i] / k
}
```

Visualize

```
ggplot(data.frame(span_val, cv_rsq), aes(x = span_val, y = cv_rsq)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(colour="darkblue", size=1) +
  labs(title=expression(paste("Plot XIII: ", R^{2}, " vs. Span"))) +
```

```
  ylab(label=expression(R^{2})) +
  xlab("Span") +
  theme_bw()
```



The above plot shows that the hightest R^2 value is reached with a span of 0.9. This value is used in the following step to build the GAM model.

GAM modelling

```
# Null model
gam_null = gam(ViolentCrimesPerPop ~ 1, data=df_train2)

# GAM model
fit_gam = gam(ViolentCrimesPerPop ~ s(Population, spar=0.9) + s(PercentageBlack, spar=0.9) +
                  s(PercentageWhite, spar=0.9) + s(PercentageAsian, spar=0.9) +
                  s(PercentageHispanic, spar=0.9) + s(PercentageUrban, spar=0.9) +
                  s(MedIncome, spar=0.9),
                data=df_train2)

# Bind together
crime_accuracy <- rbind(crime_accuracy, data.frame('Models'='GAM model',
                                                   'R2_Train' = calc_rsq(df_train2$ViolentCrimesPerPop,
                                                                         predict(fit_gam, newdata=df_tra
                                                   'R2_Test' = calc_rsq(df_test2$ViolentCrimesPerPop,
                                                                        predict(fit_gam, newdata=df_tes
```

```
# Show data
pander(crime_accuracy, justify='center')
```

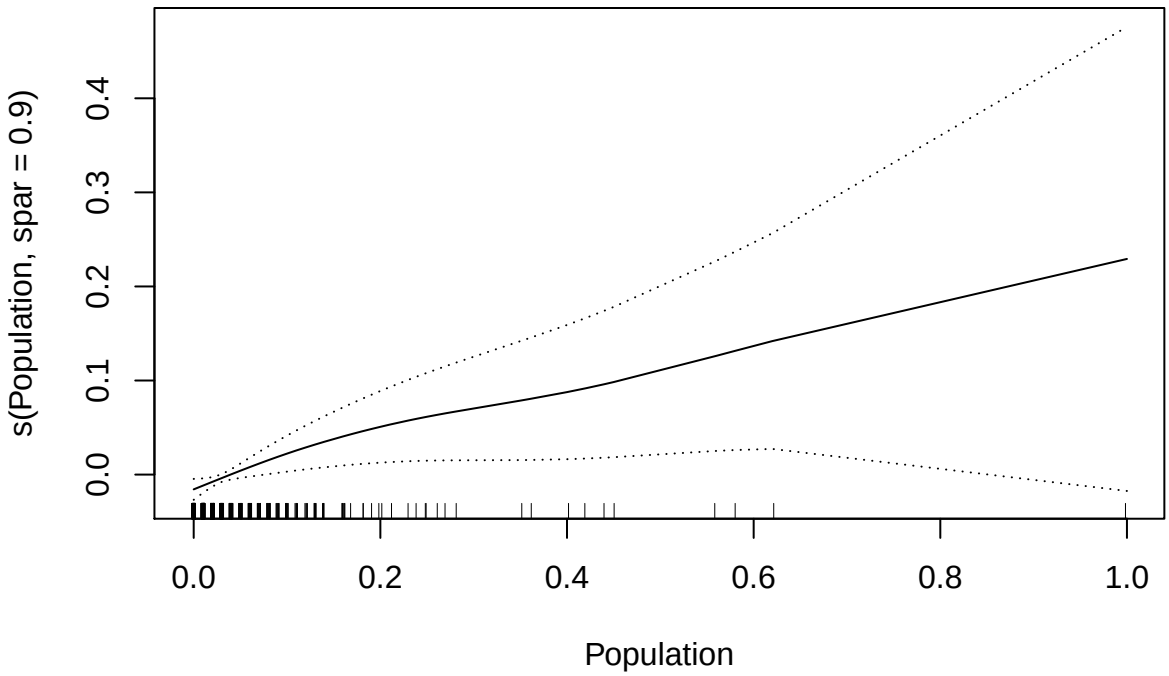
Models	R2_Train	R2_Test
Linear model	0.6185	0.5554
Poly 2 model	0.6328	0.5753
Poly 3 model	0.6437	0.5734
B-Spline model	0.6437	0.5734
GAM model	0.6421	0.5772

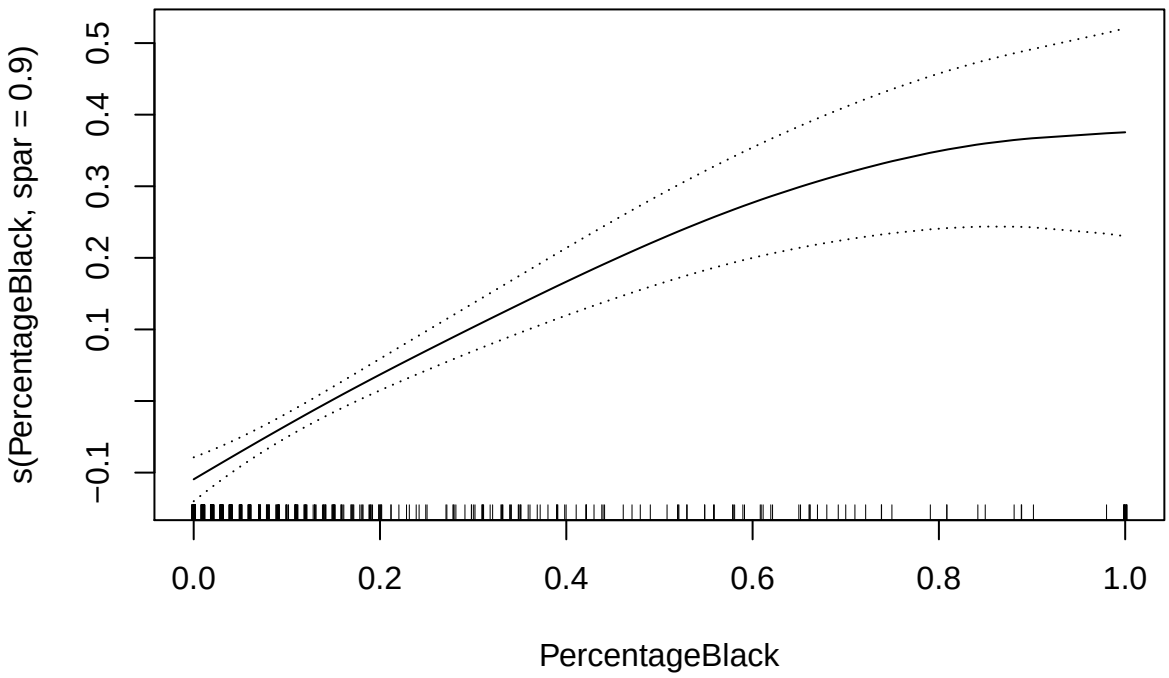
The above table shows, that the GAM model has the highest R^2 ov all the models.

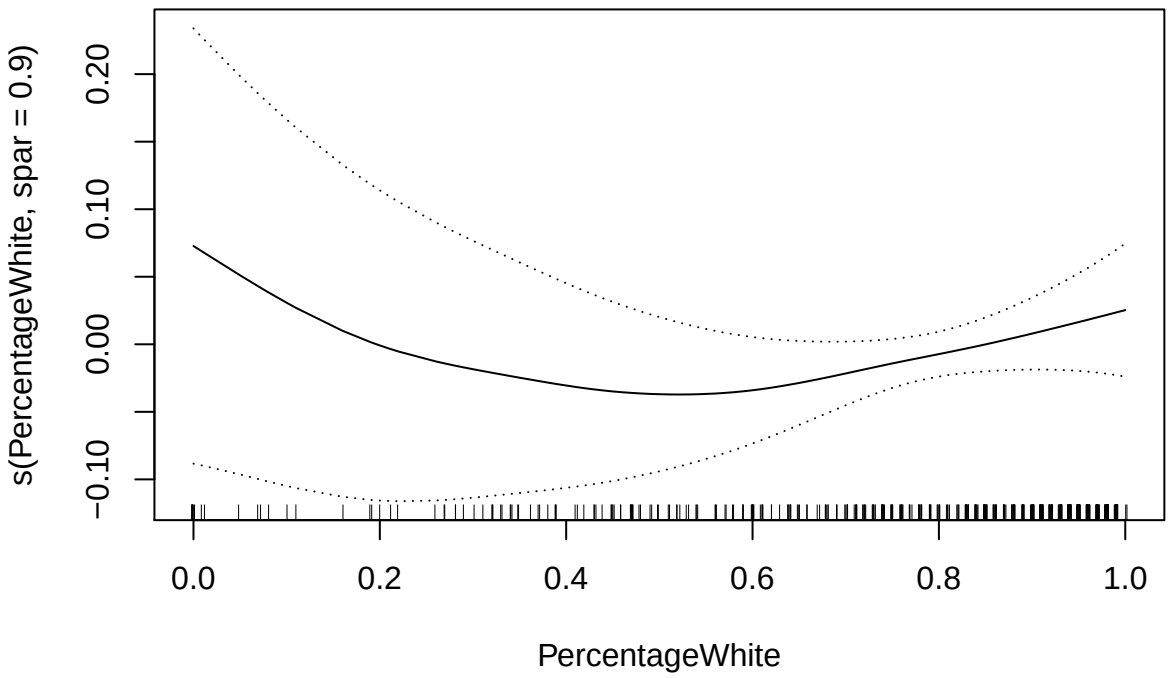
2. Plot the GAM models

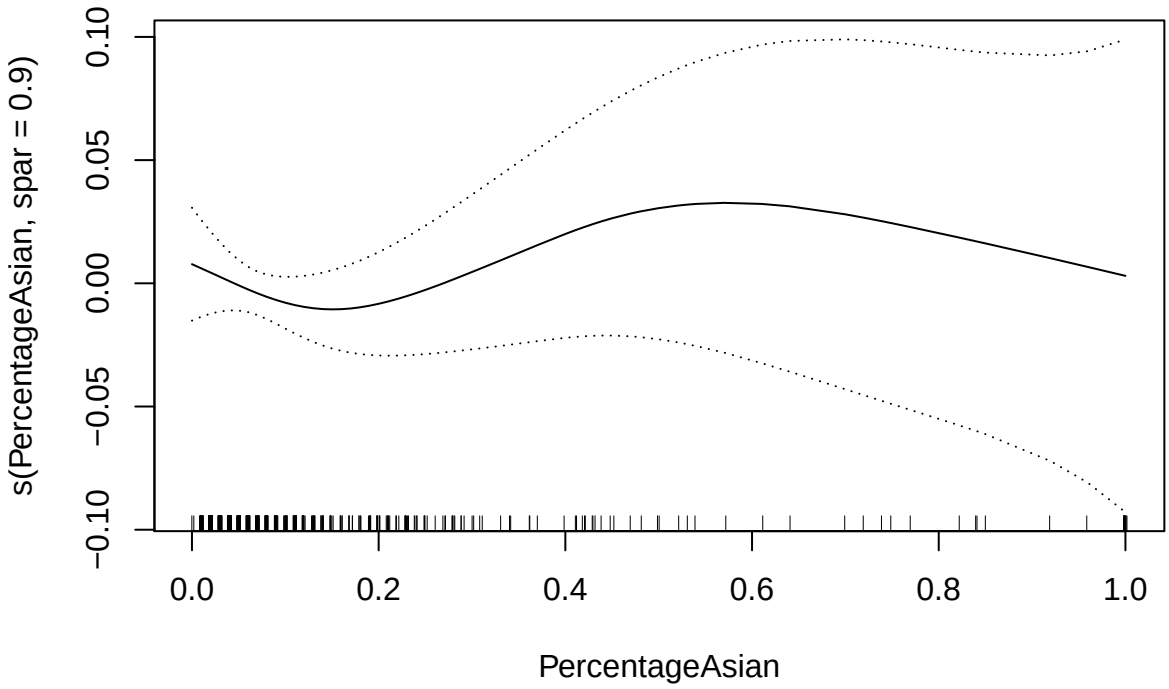
Plotting and examanation of the smooth of each predictor for the fitted GAM, along with plots of upper and lower standard errors on the predictions. What are some useful insights conveyed by these plots, and by the coefficients assigned to each local model?

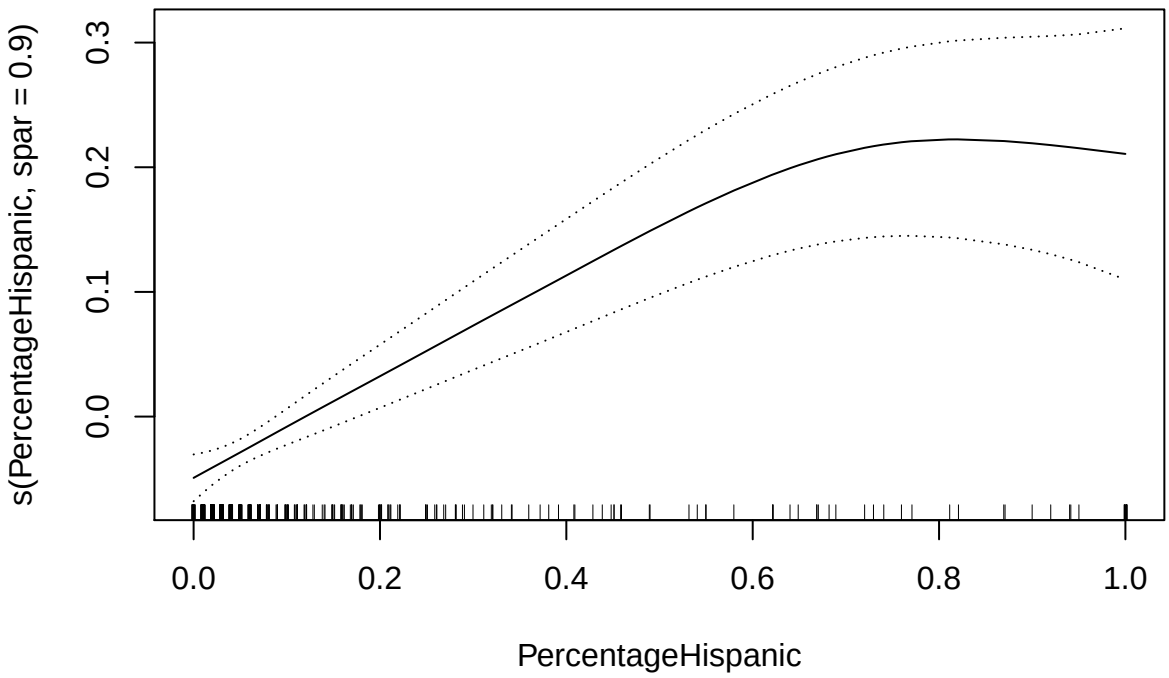
```
plot(fit_gam, se=TRUE)
```

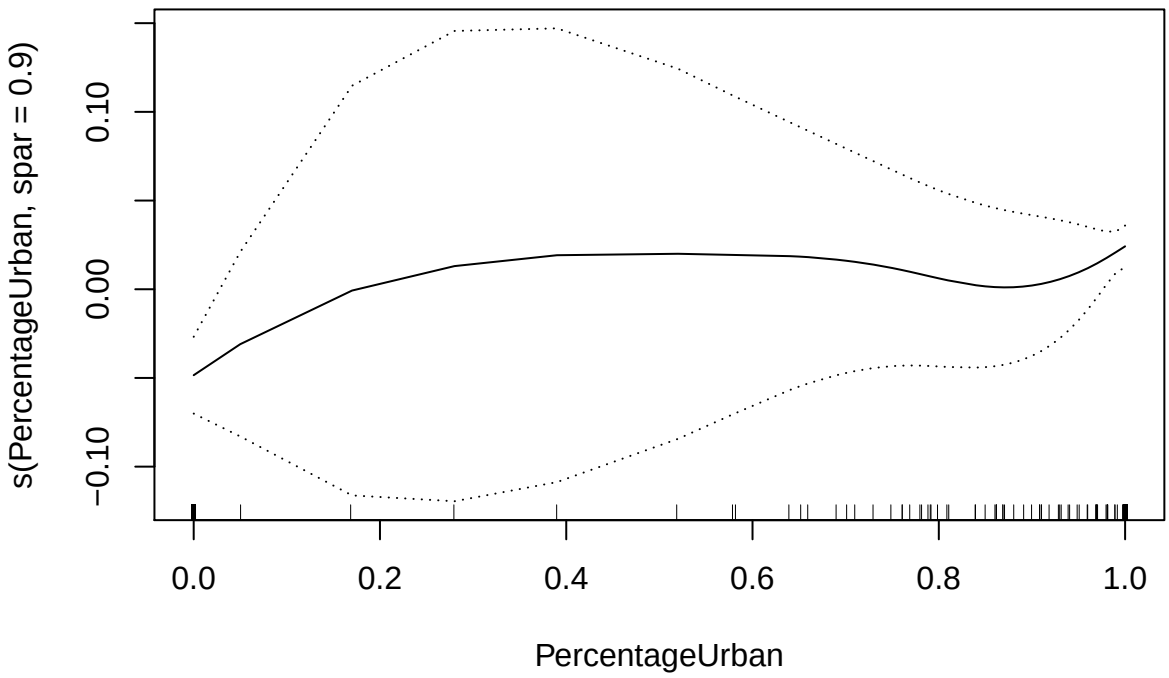


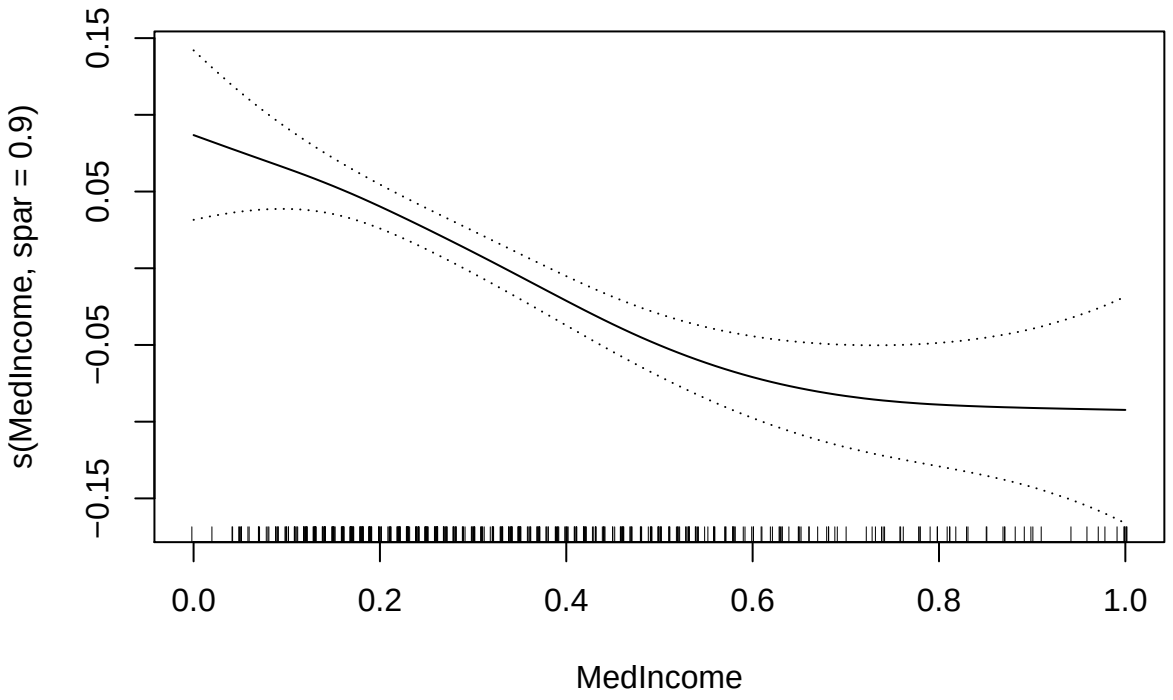

s(PercentageBlack, spar = 0.9)
−0.1
0.1
0.2
0.3
0.4
0.5
0.0
0.2
0.4
0.6
0.8
1.0
PercentageBlack


s(PercentageWhite, spar = 0.9)
−0.10
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
PercentageWhite


s(PercentageAsian, spar = 0.9)
−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 = 0.9)
0.0
0.1
0.2
0.3
0.0
0.2
0.4
0.6
0.8
1.0
PercentageHispanic


s(PercentageUrban, spar = 0.9)
−0.10
0.00
0.10
0.0
0.2
0.4
0.6
0.8
1.0
PercentageUrban



The above plot show, that for most of the x-variables (predictors) the intervall increases (i.e., diverges) as the predictor value increases. This is however not the case for PercentageUrban and PerentageWhite. Furthermore, the smallest error bars are seen for the PercentageHispanic and PercentageBlack.

```
pander(fit_gam$coefficients, justify='center')
```

Table 7: Table continues below

(Intercept)	s(Population, spar = 0.9)	s(PercentageBlack, spar = 0.9)
0.08128	0.2727	0.5609

Table 8: Table continues below

s(PercentageWhite, spar = 0.9)	s(PercentageAsian, spar = 0.9)
0.03312	0.01757

Table 9: Table continues below

s(PercentageHispanic, spar = 0.9)	s(PercentageUrban, spar = 0.9)
0.3104	0.07099

s(MedIncome, spar = 0.9)
-0.2258

Looking at the table above the coefficients show, that the predictor for PercentageBlack predictor has the highest predictive power followed by PercentageHispanic and Population. This is in accordance the already discussed plots where the error is associated with each coefficient.

3. Likelihood ratio test

Use a likelihood ratio test to compare GAM with the linear regression model fitted previously. Re-fit a GAM leaving out the predictors 'PrecentageAsian' and 'PercentageUrban'. Using a likelihood ratio test, comment if the new model is preferred to a GAM with all predictors.

ANOVA test

```
pander(anova(fit_lm, fit_gam, test='Chi'), justify='center')
```

Table 11: Analysis of Variance Table

Res.Df	RSS	Df	Sum of Sq	Pr(>Chi)
490	10.15	NA	NA	NA
476.6	9.517	13.41	0.6293	0.003495

Looking at the ANOVA test above conducted between the linear model and the GAM model, it can be seen that the GAM model is better than the linear model with a significance of 0.0034.

Refining the model

```
fit_gam_new <- gam(ViolentCrimesPerPop ~ s(Population, spar=0.9) +
                     s(PercentageBlack, spar=0.9) + s(PercentageWhite, spar=0.9) +
                     s(PercentageHispanic, spar=0.9) + s(MedIncome, spar=0.9),
                   data=df_train2)

pander(anova(fit_gam_new, fit_gam, test='Chi'), justify='center')
```

Table 12: Analysis of Deviance Table

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
482.9	9.981	NA	NA	NA
476.6	9.517	6.26	0.465	0.0008693

Considering the likelihood of the new GAM model compared to the original GAM model, the more complex model is seen to be significantly different with a value of 0.0008.

Part 2c: Including interaction terms

Re-fit the GAM with the following interaction terms included:

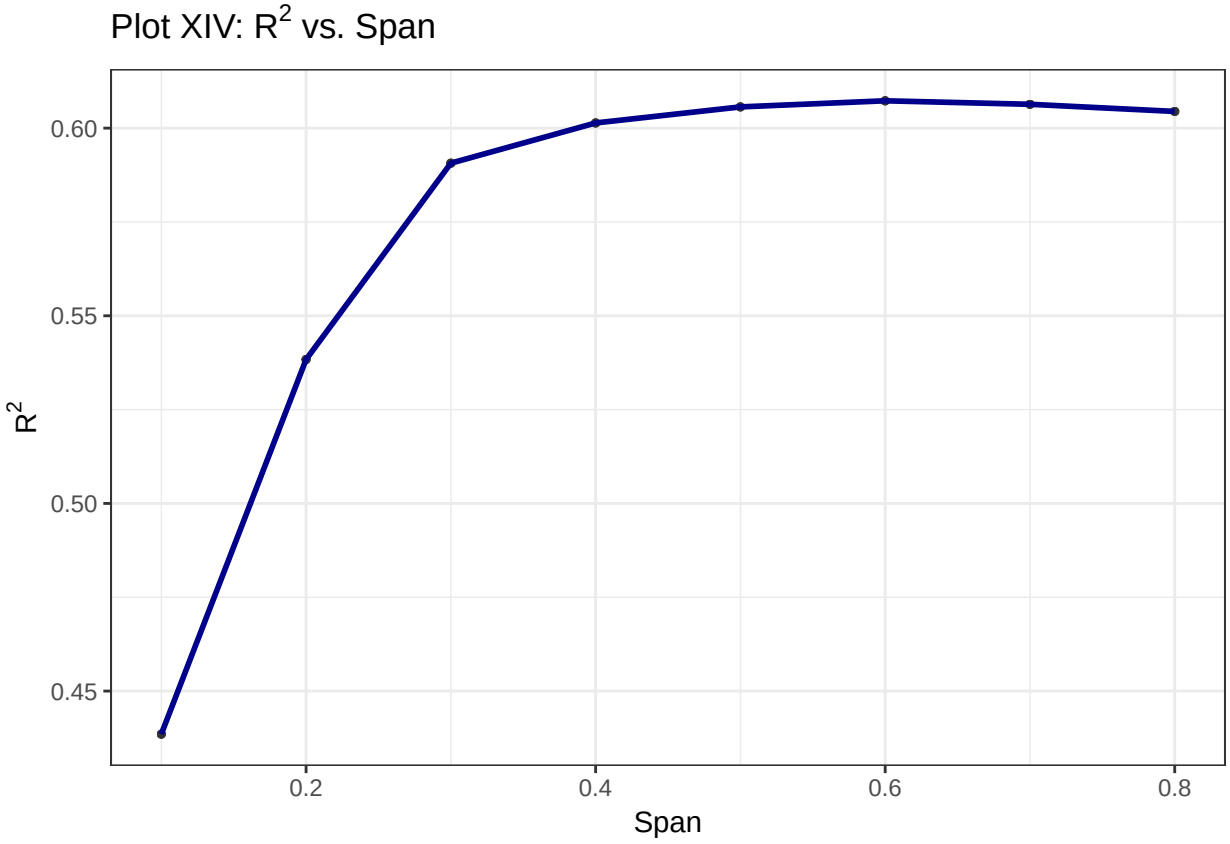
- A local regression basis function involving attributes 'Population', 'PercentageUrban' and 'MedIncome'
- A local regression basis function involving a race-related attribute and 'MedIncome'

Note: Because of performance problems the warning for knitr has been deactivet for some chunks. For more detail see: https://piazza.com/class/ivlbdd3nigy3um?cid=98

```
# Set parameter
set.seed(123)
train = df_train2
span_val  <- seq(0.1, 0.8, by = 0.1)
k <- 5
n = length(span_val)


# Divide training set into k folds by sampling uniformly at random
folds <- sample(1:k, nrow(train), replace=TRUE)

cv_rsq <- rep(0., n) # Store cross-validated R^2 for different parameter values

# Loper over the parameter values
for(i in 1:n){

  # Loop over the folds
  for(j in 1:k){

    model <- gam(ViolentCrimesPerPop ~ s(Population, spar=0.9) +
                   s(PercentageBlack, spar=0.9) +
                   s(PercentageWhite, spar=0.9) +
                   s(PercentageAsian, spar=0.9) +
                   s(PercentageHispanic, spar=0.9) +
                   s(PercentageUrban, spar=0.9) +
                   s(MedIncome, spar=0.9) +
                 lo(Population, PercentageUrban, MedIncome, span=span_val[i]) +
                 lo(MedIncome, PercentageBlack, span=span_val[i]),
                data=train[folds!=j, ], control = gam.control(maxit = 1000, bf.maxit = 1000))

    # Prediction
    pred = predict(model, train[folds == j, ])

    # Compute R^2
    cv_rsq[i] = cv_rsq[i] + calc_rsq(train[folds == j, ]$ViolentCrimesPerPop, pred)
  }

  # Average R^2 across k folds
  cv_rsq[i] = cv_rsq[i] / k
}
```

Visualize

```
ggplot(data.frame(span_val, cv_rsq), aes(x=span_val, y=cv_rsq)) +
  geom_point(stroke=0, alpha=0.8) +
  geom_line(colour="darkblue", size=1) +
  labs(title=expression(paste("Plot XIV: ", R^{2}, " vs. Span"))) +
  ylab(label=expression(R^{2})) +
  xlab("Span") +
  theme_bw()
```



The above plot shows that the hightest R^2 value is reached with a span of 0.6. This value is used in the following step to build the GAM model.

```
fit_gam_new2 <- gam(ViolentCrimesPerPop ~ s(Population, spar=0.9) +
                      s(PercentageBlack, spar=0.9) + s(PercentageWhite, spar=0.9) +
                      s(PercentageHispanic, spar=0.9) + s(MedIncome, spar=0.9) +
                       lo(Population, PercentageUrban, MedIncome, span=0.6) +
                       lo(MedIncome, PercentageBlack, span=0.6),
                    data=df_train2)

pander(anova(fit_gam_new2, fit_gam_new, test='Chi'), justify='center')
```

Table 13: Analysis of Deviance Table

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
471.4	9.018	NA	NA	NA

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
482.9	9.981	-11.5	-0.9635	0.0000008091

The model with interaction term is significantly different from the other model with a value of 0.0000008.

Comparing the GAM models

```
R2_measure <- data.frame(rbind(c('Original GAM model',
                                 calc_rsq(df_test2$ViolentCrimesPerPop,
                                          predict(fit_gam, newdata=df_test2))),
                               c('GAM model w/o interaction',
                                 calc_rsq(df_test2$ViolentCrimesPerPop,
                                          predict(fit_gam_new,

                               c('GAM model w interaction',  calc_rsq(df_test2$ViolentCrimesPerPop, pre
names(R2_measure) <- c("Model", "Test R2")
pander(R2_measure, justify='center')
```

Model	Test R2
Original GAM model	0.577206767855598
GAM model w/o interaction	0.573277994998558
GAM model w interaction	0.594665864321491

The GAM model that includes interaction terms is better in predicting the crime rate compared to the other two GAM models. Compared to the linear and polynomial models, the GAM model with interaction offers a slight improvement in the predictive accuracy on the test set.