MARS

Fitting Multivariate Adaptive Regression Splines (MARS) Models

Introduction of the package

- Package Introduction

Our goal is to create an R package that implements the Multivariate Adaptive Regression Splines (MARS) algorithm described in: Jerome H. Friedman. "Multivariate Adaptive Regression Splines." Ann. Statist. 19 (1) 1 - 67, March, 1991. https://doi.org/10.1214/aos/1176347963.

- Features

The package includes the main function `mars()` to fit a mars model and four methods `print()`, `predict()`, `plot()`, and `summary()` compatible with mars object.

Getting Started

- Install the package from Github. First install and load the `devtools` package. Then install `mars` with the following codes

```
install_github(repo="https://github.com/SFUStatgen/SFUStat360Projects",
               subdir="Projects/Project_tauseef1/mars")
```

- Now load the package

```
library(mars)
```

The MARS Algorithm

- The algorithm will search for, and discover, nonlinearities in the data that help maximize predictive accuracy. Multivariate adaptive regression splines (MARS), an algorithm that essentially creates a piecewise linear model which provides an intuitive stepping block into nonlinearity after grasping the concept of linear regression and other intrinsically linear models.

Preparing Inputs

Two arguments required and one argument optional:

1. An R formula is required to specify the response and explanatory variables. The formula can be constructed using `formula()` function in R.
2. A data frame containing the data to analyze is required. For this example, we will use `Wage` data from `ISLR` package.

```r
library(ISLR)
data(Wage)
myform <- formula(wage ~ age+education)
```

3. The optional argument is a `mars.control` object. Users should use the constructor `mars.control()` to specify the three model fitting parameters.

- The parameter `Mmax` is maximum number of basis functions. Should be an even integer, the default value is 2.
- The parameter `d` is the coefficient in the penalty term of the generalized cross validation measure, the default value is 3.
- The parameter `trace` is should we print status information about the fitting? the default value is FALSE.

For example,

```r
mc <- mars.control(Mmax=6)
```

Calling mars()

We call the mars function using `mars()` and specify the formula and data arguments. In the following example, the formula is `y~.` and the data is `mars::marstestdata`. The mars model is stored in an object called `mm`.

```r
mm <- mars(y~x1+x2, data=mars::marstestdata)
```

Using mars methods

- Print some information of the model

Using `print()` on a `mars` object will print out the function call and the coefficients of the model. There is one required argument, `x`, for `print()` function. `x` should be a `mars` object.

```r
print(mm)
#> [1] "Call:"
#> mars(formula = y ~ x1 + x2, data = mars::marstestdata)
#> [1] "Coefficients:"
#>            B0           B2
#> 0.0001671199 3.3986822771
```

- Get a summary

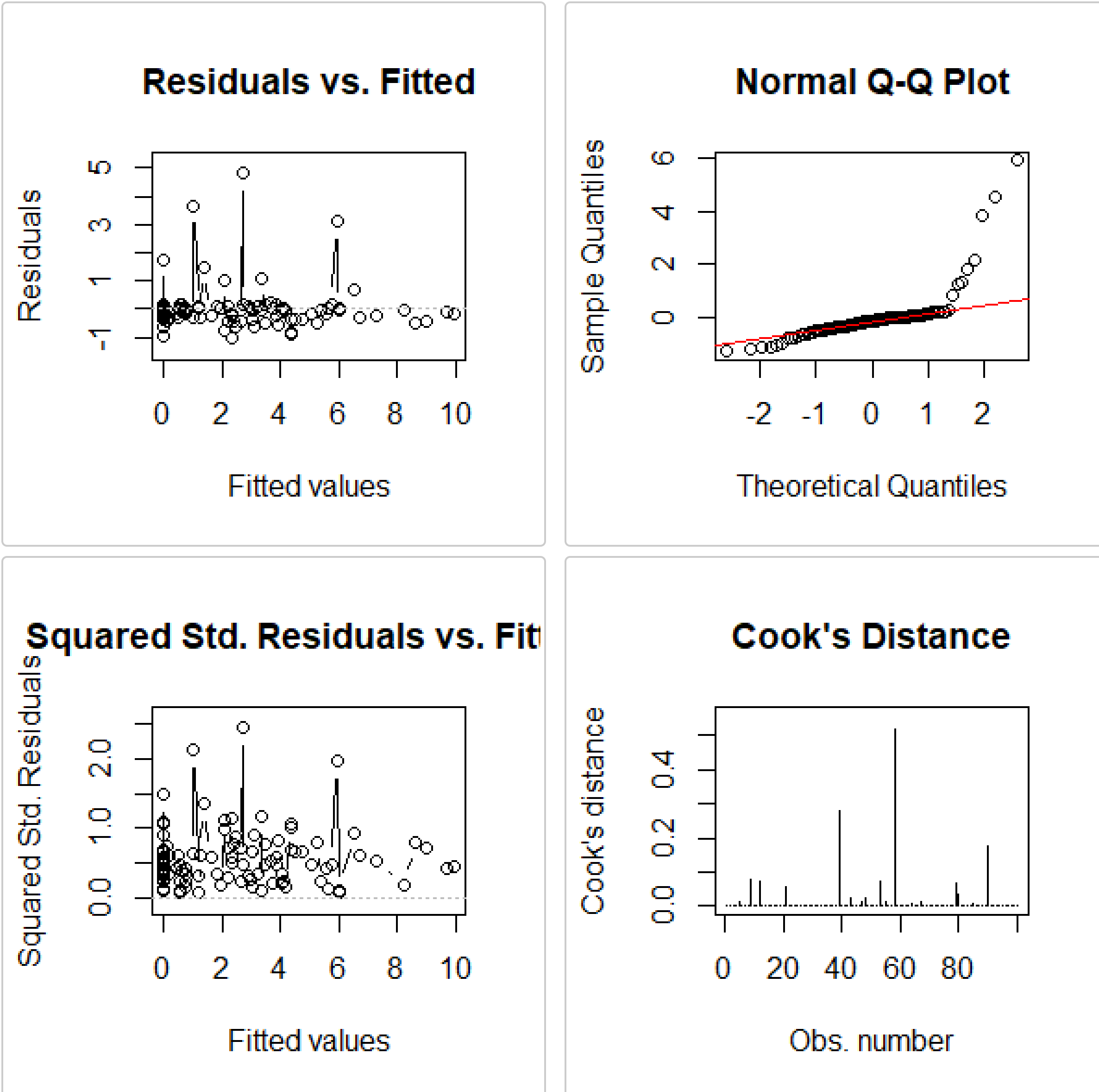
Using `summary()` on a `mars` object will output some summary statistics of the data and the fitted model. There is one required argument, `object`, for `summary()` function. `object` should be a `mars` object.

```
summary(mm)
#>
#> Call:
#> mars(formula = y ~ x1 + x2, data = mars::marstestdata)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.0465 -0.3135 -0.0978  0.0342  4.7984
#>
#> Coefficients:
#>     Estimate Std. Error t value Pr(>|t|)
#> B0 0.0001671  0.1166062   0.001    0.999
#> B2 3.3986823  0.1106875  30.705   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.8129 on 98 degrees of freedom
#> Multiple R-squared:  0.9519, Adjusted R-squared:  0.9509
#> F-statistic: 969.9 on 2 and 98 DF,  p-value: < 2.2e-16
```

- Get a plot

Using `plot()` on a `mars` object will give four plots: residuals vs fitted value, normal Q-Q plot, squared standardized residuals vs. fitted value, and the Cook’s distance. There is one required argument, `x`, for `plot()` function. `x` should be a `mars` object.

```
plot(mm)
```



- Make predictions

Using `predict()` on a `mars` object to predict response variable value based on new data or extracting fitted values in the existing mars model. If `newdata` is provided, the function returns the predicted values based on the new data. If `newdata` is missing, the function returns the fitted values of the mars model. The first argument `object` is required, and it should be a `mars` object. The second argument `newdata` is optional, which should be a data frame.

```
pred <- predict(mm, newdata=data.frame(x1=rnorm(10),x2=rnorm(10)))
```

Example

In this example, we analyze the relationship between the sales price and the year built of the properties.

We first extract the data from the `AmesHousing` package.

```
data <- AmesHousing::make_ames()
```

Then, we fit a `mars` model using formula `Sale_Price ~ Year_Built` on the `AmesHousing` data. We set the `Mmax` to be 6.

```
mars_fit <- mars(Sale_Price ~ Year_Built, data, control = mars.control(Mmax=6))
```

We print the function call, coefficients, and some summary statistics of the fitted model.

```
print(mars_fit)
#> [1] "Call:"
#> mars(formula = Sale_Price ~ Year_Built, data = data, control = mars.control(Mmax = 6))
#> [1] "Coefficients:"
#>          B0          B1          B3          B5          B6
#>   36630.805    3301.967  17108.619 -20723.702   18016.470
```

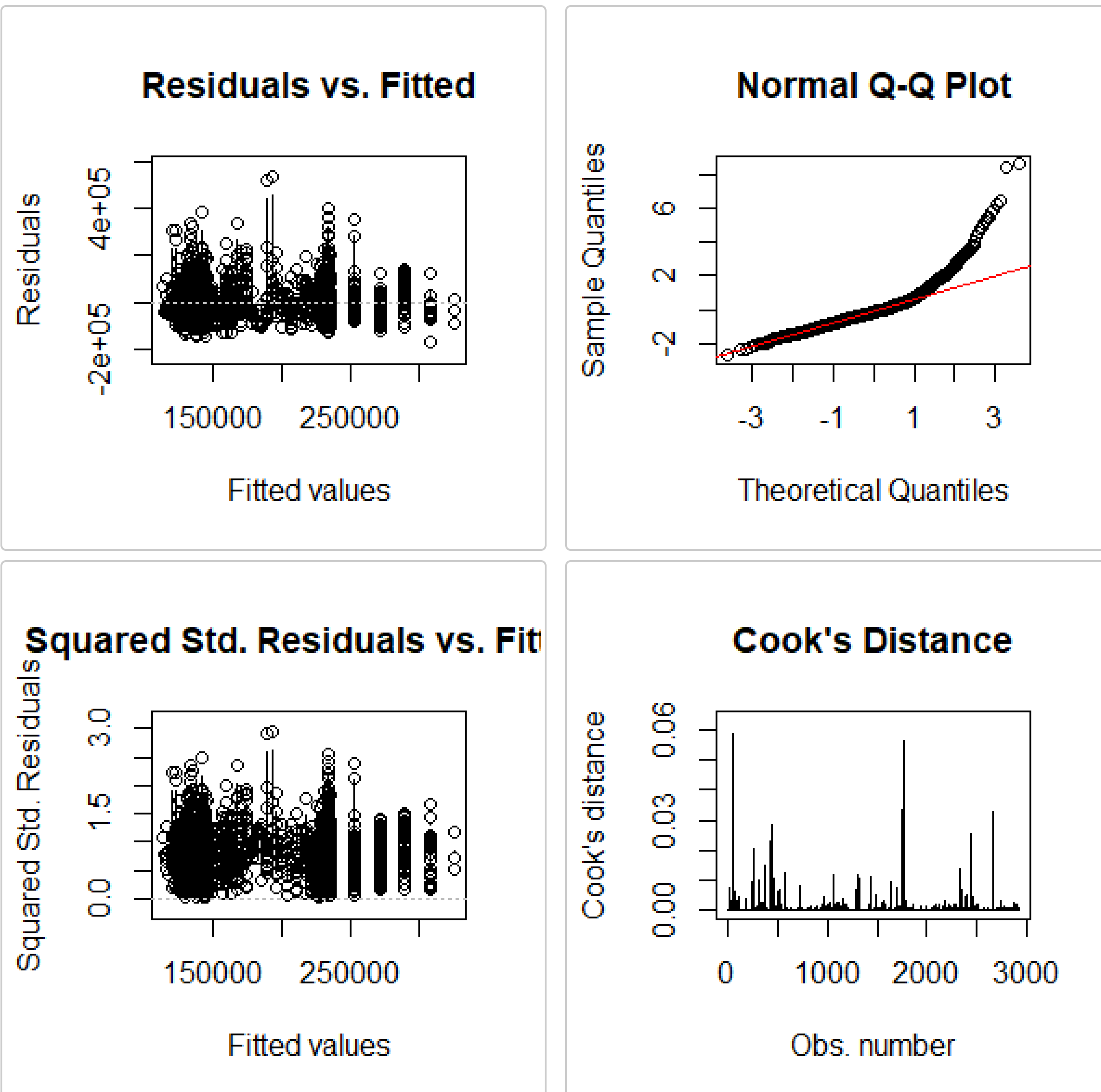
```
summary(mars_fit)
#>
#> Call:
#> mars(formula = Sale_Price ~ Year_Built, data = data, control = mars.control(Mmax = 6))
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -168328  -36467   -9042   21069  530174
#>
#> Coefficients:
#>    Estimate Std. Error t value Pr(>|t|)
#> B0  36630.8    24688.9   1.484    0.138
#> B1   3302.0      255.8  12.907  < 2e-16 ***
#> B3  17108.6     2331.8   7.337 2.82e-13 ***
#> B5 -20723.7     2401.8  -8.628  < 2e-16 ***
#> B6  18016.5     2056.7   8.760  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 61870 on 2925 degrees of freedom
#> Multiple R-squared:  0.9022, Adjusted R-squared:  0.902
#> F-statistic:  5395 on 5 and 2925 DF,  p-value: < 2.2e-16
```

```
mars_fit$Bfuncs
#> [[1]]
#> NULL
#>
#> [[2]]
#>       s v    t
#> [1,] -1 1 1972
#>
#> [[3]]
#>       s v    t
#> [1,] -1 1 2005
#>
```

```
#> [[4]]
#>        s v    t
#> [1,] -1 1 1994
#>
#> [[5]]
#>       s v    t
#> [1,] 1 1 1994
```

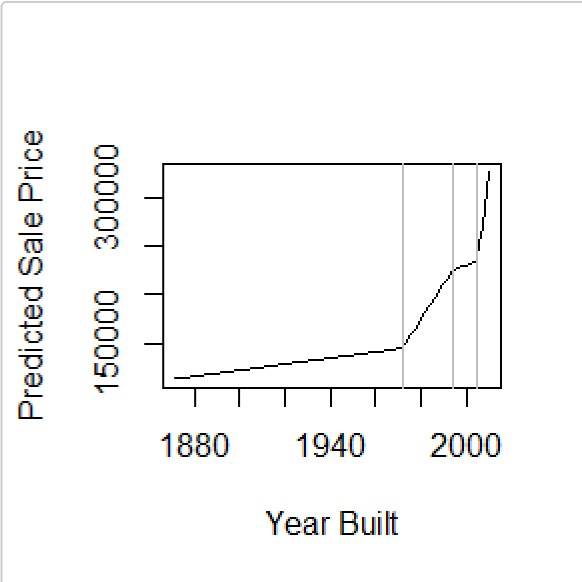
By extracting `Bfuncs`, which store the information of the split points, we can see that there are three significant turning points in the relationship between the property prices and the year built, i.e. 1972, 1994, and 2005.

```
plot(mars_fit)
```



The four plots above shows that the model performance is similar for smaller fitted values and for larger fitted values and the data doesn't seem to come from a normal distribution since the right tail of the qq plot is quite off.

```r
newdat <- data.frame(Year_Built = 1872:2010)
pred_price <- predict(mars_fit, newdata = newdat)
plot(unlist(newdat), pred_price, type="l", xlab="Year Built", ylab="Predicted Sale Price")
abline(v=c(1972, 1994, 2005), col="grey")
```



We show the trend of the property sale prices vs. their year built using the `predict()` function with the `newdata` equals `1872:2010`, the range of the `Year_Built` in the data set.

We can see that, the property prices increase all the time and the speed of the increase is mostly increasing as well, except for year built between 1994 and 2005, in which the speed of the rise of price slows down a bit.

References

- Jerome H. Friedman. “Multivariate Adaptive Regression Splines.” Ann. Statist. 19 (1) 1 - 67, March, 1991. https://doi.org/10.1214/aos/1176347963.