Summary of Chapter 3: Linear Regression

Introduction to Linear Regression

Chapter 3 of the textbook introduces linear regression, a foundational statistical learning method used for predicting a quantitative response. Despite being a basic method compared to modern statistical learning techniques, linear regression remains a powerful and widely used tool. The chapter highlights its importance as a stepping stone for understanding more advanced machine learning and statistical methods.

The chapter begins by discussing a practical application: predicting product sales based on advertising budgets for TV, radio, and newspapers. It outlines key questions that linear regression can help answer, such as:

- Whether a relationship exists between advertising and sales.
- The strength of this relationship.
- The contribution of each advertising medium.
- The accuracy of future sales predictions.

Simple Linear Regression

Simple linear regression models a response variable Y as a linear function of a single predictor X:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

where:

- β_0 is the intercept.
- β_1 is the slope.
- ϵ is an error term.

Estimating Coefficients

The coefficients β_0 and β_1 are unknown and must be estimated using data. The most common approach is the **least squares method**, which minimizes the residual sum of squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = \beta_0 + \beta_1 x_i$ is the predicted value of Y.

Using calculus, the least squares estimates of the coefficients are:

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of X and Y.

Assessing the Accuracy of the Model

To assess the accuracy of estimated coefficients, standard errors (SE) are computed:

$$SE(\hat{\beta}_1) = \frac{\sigma}{\sqrt{\sum(x_i - \bar{x})^2}}$$

where σ is the standard deviation of the error term. Confidence intervals for the coefficients can be computed as:

$$\hat{\beta}_1 \pm 2 \times SE(\hat{\beta}_1)$$

Hypothesis Testing

A hypothesis test can determine whether a predictor is significantly related to the response. The null hypothesis is:

$$H_0 : \beta_1 = 0$$

The test statistic follows a t-distribution:

$$t = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)}$$

A small **p-value** indicates strong evidence against H_0, meaning X significantly affects Y.

Assessing Model Fit

Two important measures of model fit are:

- **Residual Standard Error (RSE)**: Measures the model's average prediction error.
- R^2 Statistic: Measures the proportion of variance in Y explained by X:

  $$R^2 = 1 - \frac{RSS}{TSS}$$

  where TSS (Total Sum of Squares) represents total variation in Y. Higher R^2 values indicate a better fit.

Multiple Linear Regression

Multiple linear regression extends simple regression to include multiple predictors:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

where $X_1, X_2, \ldots, X_p$ are multiple predictors.

Estimating Coefficients

Similar to simple regression, the coefficients are estimated using the least squares method, minimizing:

$$RSS = \sum(y_i - \hat{y}_i)^2$$

where $\hat{y}_i = \beta_0 + \sum \beta_j X_{ij}$.

Assessing Significance

The **F-test** is used to test whether at least one predictor is significantly related to Y:

$$F = \frac{(TSS - RSS)/p}{RSS/(n - p - 1)}$$

A high F-statistic with a low p-value suggests at least one predictor significantly contributes to the model.

Variable Selection

To determine which predictors to keep, variable selection methods include:

- **Forward selection**: Start with no predictors, add them one by one based on their significance.
- **Backward selection**: Start with all predictors, remove the least significant one iteratively.
- **Mixed selection**: Combines both forward and backward selection.

Collinearity

Collinearity occurs when predictors are highly correlated, making it difficult to isolate their effects. The **Variance Inflation Factor (VIF)** detects collinearity:

$$VIF(X_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}$$

High VIF values (above 5 or 10) indicate problematic collinearity.

Extensions to the Linear Model

Interaction Effects

Interaction terms capture synergistic effects between predictors:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 (X_1 X_2) + \epsilon$$

If β_3 is significant, the effect of X_1 on Y depends on X_2.

Non-linearity

The standard model assumes a linear relationship, but polynomial regression can capture non-linearity:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p + \epsilon$$

which introduces curvature to the model.

Qualitative Predictors

Categorical variables can be included using **dummy variables**. If a predictor has k levels, we create $k-1$ dummy variables.

For example, if "region" has three levels (East, West, South), we create:

$$X_1 = \begin{cases} 1, & \text{if South} \\ 0, & \text{otherwise} \end{cases}, \quad X_2 = \begin{cases} 1, & \text{if West} \\ 0, & \text{otherwise} \end{cases}$$

One category (East) is the **baseline**.

Common Problems in Regression

1. **Non-linearity**: Addressed using polynomial transformations or other modeling techniques.
2. **Correlation of error terms**: Often found in **time series data**.
3. **Non-constant variance (heteroscedasticity)**: Residual plots help detect this; transformations like $\log(Y)$ can help.
4. **Outliers**: Large residuals suggest unusual data points.
5. **High-leverage points**: Have extreme predictor values and can disproportionately affect the model.
6. **Collinearity**: Addressed by removing correlated predictors or using principal component analysis.

Comparison of K-Nearest Neighbors (KNN) with Linear Regression

The chapter provides a detailed comparison between **linear regression** (a parametric method) and **K-Nearest Neighbors (KNN)** (a non-parametric method).

Key Differences

1. **Assumption on Functional Form**:
   - **Linear regression** assumes a fixed functional form $f(X)$, which is beneficial when the true relationship is close to linear.
   - **KNN regression** does not assume any parametric form, making it more flexible in capturing complex relationships.
2. **Interpretability vs. Flexibility**:
   - **Linear regression** is highly interpretable, allowing for hypothesis testing and confidence intervals.
   - **KNN** is more flexible but lacks interpretability; it does not provide explicit coefficient estimates or statistical inference.
3. **Bias-Variance Tradeoff**:

- **Linear regression** has **low variance** but may have **high bias** if the true relationship is non-linear.
- **KNN** can have **low bias** but tends to have **high variance**, especially when K is small.

4. **Performance in Low vs. High Dimensions**:

- When p (number of predictors) is small, **KNN may outperform linear regression if the true relationship is highly non-linear**.
- However, as p increases, **KNN suffers from the "curse of dimensionality," leading to poor performance**, while **linear regression remains stable**.

Illustrative Findings from the Chapter

- **When the true relationship is linear**, **linear regression performs better** than KNN, as KNN introduces unnecessary variance.
- **When the true relationship is non-linear**, **KNN can outperform linear regression**, particularly when K is chosen optimally.
- **When there are many irrelevant predictors (high p)**, **KNN struggles** because neighbors are no longer close in high-dimensional space, making linear regression the better choice.

Final Takeaway

- **Linear regression** is the preferred choice when the relationship is approximately linear or when interpretability is important.
- **KNN** is more flexible and useful when the true relationship is complex and non-linear, but it requires careful tuning of K and suffers in high dimensions.

This comparison underscores the importance of understanding the nature of the data before selecting a modeling approach.

Conclusion

Chapter 3 provides a comprehensive guide to linear regression, covering:

- **Model formulation, estimation, and interpretation**.
- **Assessing model fit and hypothesis testing**.
- **Extensions such as interactions and polynomial regression**.
- **Practical issues such as collinearity and outliers**.
- **Comparison with KNN for different data scenarios**.

This foundation is essential for understanding more advanced regression techniques and machine learning models.

In [ ]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
```

In [ ]:
```
import statsmodels.api as sm
```

In [ ]:
```
from statsmodels.stats.outliers_influence \
  import variance_inflation_factor as VIF
from statsmodels.stats.anova import anova_lm
```

In [ ]:
```
#pip install ISLP
```

```
Collecting ISLP
  Downloading ISLP-0.4.0-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: numpy>=1.7.1 in /usr/local/lib/python3.11/dist-packa
ges (from ISLP) (1.26.4)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.11/dist-package
s (from ISLP) (1.13.1)
Requirement already satisfied: pandas>=0.20 in /usr/local/lib/python3.11/dist-packa
ges (from ISLP) (2.2.2)
Requirement already satisfied: lxml in /usr/local/lib/python3.11/dist-packages (fro
m ISLP) (5.3.1)
Requirement already satisfied: scikit-learn>=1.2 in /usr/local/lib/python3.11/dist-
packages (from ISLP) (1.6.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (f
rom ISLP) (1.4.2)
Requirement already satisfied: statsmodels>=0.13 in /usr/local/lib/python3.11/dist-
packages (from ISLP) (0.14.4)
Collecting lifelines (from ISLP)
  Downloading lifelines-0.30.0-py3-none-any.whl.metadata (3.2 kB)
Collecting pygam (from ISLP)
  Downloading pygam-0.9.1-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (fr
om ISLP) (2.5.1+cu124)
Collecting pytorch-lightning (from ISLP)
  Downloading pytorch_lightning-2.5.0.post0-py3-none-any.whl.metadata (21 kB)
Collecting torchmetrics (from ISLP)
  Downloading torchmetrics-1.6.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/
dist-packages (from pandas>=0.20->ISLP) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packa
ges (from pandas>=0.20->ISLP) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pac
kages (from pandas>=0.20->ISLP) (2025.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/di
st-packages (from scikit-learn>=1.2->ISLP) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packa
ges (from statsmodels>=0.13->ISLP) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-pa
ckages (from statsmodels>=0.13->ISLP) (24.2)
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.11/dist-pa
ckages (from lifelines->ISLP) (3.10.0)
Requirement already satisfied: autograd>=1.5 in /usr/local/lib/python3.11/dist-pack
ages (from lifelines->ISLP) (1.7.0)
Collecting autograd-gamma>=0.3 (from lifelines->ISLP)
  Downloading autograd-gamma-0.5.0.tar.gz (4.0 kB)
  Preparing metadata (setup.py) ... done
Collecting formulaic>=0.2.2 (from lifelines->ISLP)
  Downloading formulaic-1.1.1-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: progressbar2<5.0.0,>=4.2.0 in /usr/local/lib/python
3.11/dist-packages (from pygam->ISLP) (4.5.0)
Collecting scipy>=0.9 (from ISLP)
  Downloading scipy-1.11.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.w
hl.metadata (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.4/60.4 kB 2.4 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.11/dist-packa
ges (from pytorch-lightning->ISLP) (4.67.1)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.11/dist-packag
es (from pytorch-lightning->ISLP) (6.0.2)
Requirement already satisfied: fsspec>=2022.5.0 in /usr/local/lib/python3.11/dist-p
ackages (from fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (2024.10.0)
```

```
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.1
1/dist-packages (from pytorch-lightning->ISLP) (4.12.2)
Collecting lightning-utilities>=0.10.0 (from pytorch-lightning->ISLP)
  Downloading lightning_utilities-0.12.0-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages
(from torch->ISLP) (3.17.0)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages
(from torch->ISLP) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (f
rom torch->ISLP) (3.1.5)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch->ISLP)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.met
adata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch->ISLP)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.m
etadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch->ISLP)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.met
adata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch->ISLP)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata
(1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch->ISLP)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadat
a (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch->ISLP)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata
(1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch->ISLP)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metad
ata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch->ISLP)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metad
ata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch->ISLP)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.met
adata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.1
1/dist-packages (from torch->ISLP) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python
3.11/dist-packages (from torch->ISLP) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch->ISLP)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.meta
data (1.5 kB)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-pack
ages (from torch->ISLP) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-pack
ages (from torch->ISLP) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist
-packages (from sympy==1.13.1->torch->ISLP) (1.3.0)
Collecting interface-meta>=1.2.0 (from formulaic>=0.2.2->lifelines->ISLP)
  Downloading interface_meta-1.3.0-py3-none-any.whl.metadata (6.7 kB)
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.11/dist-package
s (from formulaic>=0.2.2->lifelines->ISLP) (1.17.2)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python
3.11/dist-packages (from fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (3.11.12)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-package
s (from lightning-utilities>=0.10.0->pytorch-lightning->ISLP) (75.1.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-p
```

```
ackages (from matplotlib>=3.0->lifelines->ISLP) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packa
ges (from matplotlib>=3.0->lifelines->ISLP) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-
packages (from matplotlib>=3.0->lifelines->ISLP) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-
packages (from matplotlib>=3.0->lifelines->ISLP) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages
(from matplotlib>=3.0->lifelines->ISLP) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-p
ackages (from matplotlib>=3.0->lifelines->ISLP) (3.2.1)
Requirement already satisfied: python-utils>=3.8.1 in /usr/local/lib/python3.11/dis
t-packages (from progressbar2<5.0.0,>=4.2.0->pygam->ISLP) (3.9.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
(from python-dateutil>=2.8.2->pandas>=0.20->ISLP) (1.17.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pa
ckages (from jinja2->torch->ISLP) (3.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.1
1/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-l
ightning->ISLP) (2.4.6)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-p
ackages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning
->ISLP) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-pack
ages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->I
SLP) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-
packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightnin
g->ISLP) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dis
t-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightn
ing->ISLP) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-p
ackages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning
->ISLP) (0.2.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightnin
g->ISLP) (1.18.3)
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.11/dist-packages
(from yarl<2.0,>=1.17.0->aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorc
h-lightning->ISLP) (3.10)
Downloading ISLP-0.4.0-py3-none-any.whl (3.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 33.9 MB/s eta 0:00:00
Downloading lifelines-0.30.0-py3-none-any.whl (349 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 349.3/349.3 kB 29.4 MB/s eta 0:00:00
Downloading pygam-0.9.1-py3-none-any.whl (522 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 522.0/522.0 kB 35.2 MB/s eta 0:00:00
Downloading scipy-1.11.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(36.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.4/36.4 MB 20.8 MB/s eta 0:00:00
Downloading pytorch_lightning-2.5.0.post0-py3-none-any.whl (819 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 819.3/819.3 kB 33.2 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 M
B)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.7 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 61.1 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6
```

```
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 34.5 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (88
3 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 53.3 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.0 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 6.0 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 M
B)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.9 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 8.7 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.
5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 6.3 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 81.4 MB/s eta 0:00:00
Downloading torchmetrics-1.6.1-py3-none-any.whl (927 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 927.3/927.3 kB 57.0 MB/s eta 0:00:00
Downloading formulaic-1.1.1-py3-none-any.whl (115 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.7/115.7 kB 14.2 MB/s eta 0:00:00
Downloading lightning_utilities-0.12.0-py3-none-any.whl (28 kB)
Downloading interface_meta-1.3.0-py3-none-any.whl (14 kB)
Building wheels for collected packages: autograd-gamma
  Building wheel for autograd-gamma (setup.py) ... done
  Created wheel for autograd-gamma: filename=autograd_gamma-0.5.0-py3-none-any.whl
size=4031 sha256=e372c8a9d9eb1f113ab6bbac1d73b5ed24de76cd9c8e3f5bb4547a37c3f7e225
  Stored in directory: /root/.cache/pip/wheels/8b/67/f4/2caaae2146198dcb824f31a3038
33b07b14a5ec863fb3acd7b
Successfully built autograd-gamma
Installing collected packages: scipy, nvidia-nvjitlink-cu12, nvidia-curand-cu12, nv
idia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupt
i-cu12, nvidia-cublas-cu12, lightning-utilities, interface-meta, nvidia-cusparse-cu
12, nvidia-cudnn-cu12, autograd-gamma, pygam, nvidia-cusolver-cu12, formulaic, life
lines, torchmetrics, pytorch-lightning, ISLP
  Attempting uninstall: scipy
    Found existing installation: scipy 1.13.1
    Uninstalling scipy-1.13.1:
      Successfully uninstalled scipy-1.13.1
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
```

```
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed ISLP-0.4.0 autograd-gamma-0.5.0 formulaic-1.1.1 interface-me
ta-1.3.0 lifelines-0.30.0 lightning-utilities-0.12.0 nvidia-cublas-cu12-12.4.5.8 nv
idia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-c
u12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-curand-cu
12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-cusparse-cu12-12.3.1.170 nvidia-
nvjitlink-cu12-12.4.127 pygam-0.9.1 pytorch-lightning-2.5.0.post0 scipy-1.11.4 torc
hmetrics-1.6.1
```

In [ ]:
```python
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize,
                         poly)
```

In [ ]:
```python
#Question 8
Auto = load_data("Auto")
Auto.columns
```

Out[ ]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'year', 'origin'],
      dtype='object')
```

In [ ]:
```python
#8a
X = pd.DataFrame({'intercept': np.ones(Auto.shape[0]),
                  'horsepower': Auto['horsepower']})
X[:4]
```

Out[ ]:

name	intercept	horsepower
chevrolet chevelle malibu	1.0	130
buick skylark 320	1.0	165
plymouth satellite	1.0	150
amc rebel sst	1.0	150

In [ ]:
```
y = Auto['mpg']
model = sm.OLS(y, X)
results = model.fit()
```

In [ ]:
```
summarize(results)
```

Out[ ]:

	coef	std err	t	P>|t|
intercept	39.9359	0.717	55.660	0.0
horsepower	-0.1578	0.006	-24.489	0.0

In [ ]:
```
results.summary()
```

Out[ ]: OLS Regression Results

Dep. Variable:	mpg	**R-squared:**	0.606
Model:	OLS	**Adj. R-squared:**	0.605
Method:	Least Squares	**F-statistic:**	599.7
Date:	Tue, 18 Feb 2025	**Prob (F-statistic):**	7.03e-81
Time:	00:14:14	**Log-Likelihood:**	-1178.7
No. Observations:	392	**AIC:**	2361.
Df Residuals:	390	**BIC:**	2369.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	39.9359	0.717	55.660	0.000	38.525	41.347
horsepower	-0.1578	0.006	-24.489	0.000	-0.171	-0.145

Omnibus:	16.432	**Durbin-Watson:**	0.920
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	17.305
Skew:	0.492	**Prob(JB):**	0.000175
Kurtosis:	3.299	**Cond. No.**	322.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
np.sqrt(results.scale)/y.mean()
```

Out[ ]: 0.20923714066914834

In [ ]:
```
design = MS(['horsepower'])
X = design.fit_transform(Auto)
X[:4]
```

Out[ ]:

name	intercept	horsepower
chevrolet chevelle malibu	1.0	130
buick skylark 320	1.0	165
plymouth satellite	1.0	150
amc rebel sst	1.0	150

In [ ]:
```
new_df = pd.DataFrame({'horsepower':[98]})
newX = design.transform(new_df)
newX
```

Out[ ]:

	intercept	horsepower
0	1.0	98

In [ ]:
```
new_predictions = results.get_prediction(newX); new_predictions.predicted_mean
```

Out[ ]:
```
array([24.46707715])
```

In [ ]:
```
new_predictions.conf_int(alpha=0.05)
```

Out[ ]:
```
array([[23.97307896, 24.96107534]])
```

In [ ]:
```
new_predictions.conf_int(obs=True, alpha=0.05)
```

Out[ ]:
```
array([[14.80939607, 34.12475823]])
```

8a) i.There is a relation between the predictor and response since the t statistic is <0.05/2.

ii.There is a moderately strong relationship between the predictor and response since R^2=60.6% is high, and the percentage error of ~20% is low.

iii.The relation is **negative** between predictor and response since beta_1 estimate is less than 0.

iv.Preditcted mpg with 98 horsepower is 24.46707715.

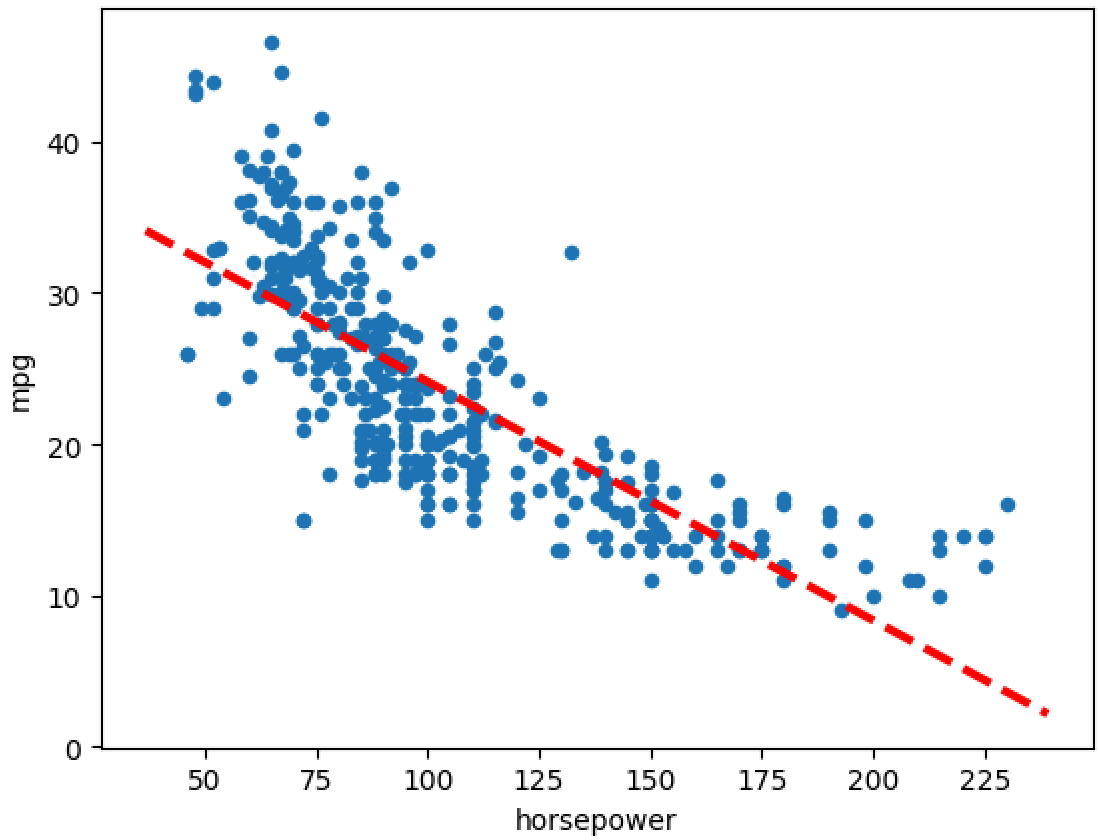
Confidence interval: (23.97307896, 24.96107534)

Prediction interval: (14.80939607, 34.12475823)

In [ ]:
```
#8b
def abline(ax, b, m, *args, **kwargs):
  "Add a line with slope m and intercept b to ax"
  xlim = ax.get_xlim()
  ylim = [m * xlim[0] + b, m * xlim[1] + b]
  ax.plot(xlim, ylim, *args, **kwargs)
```

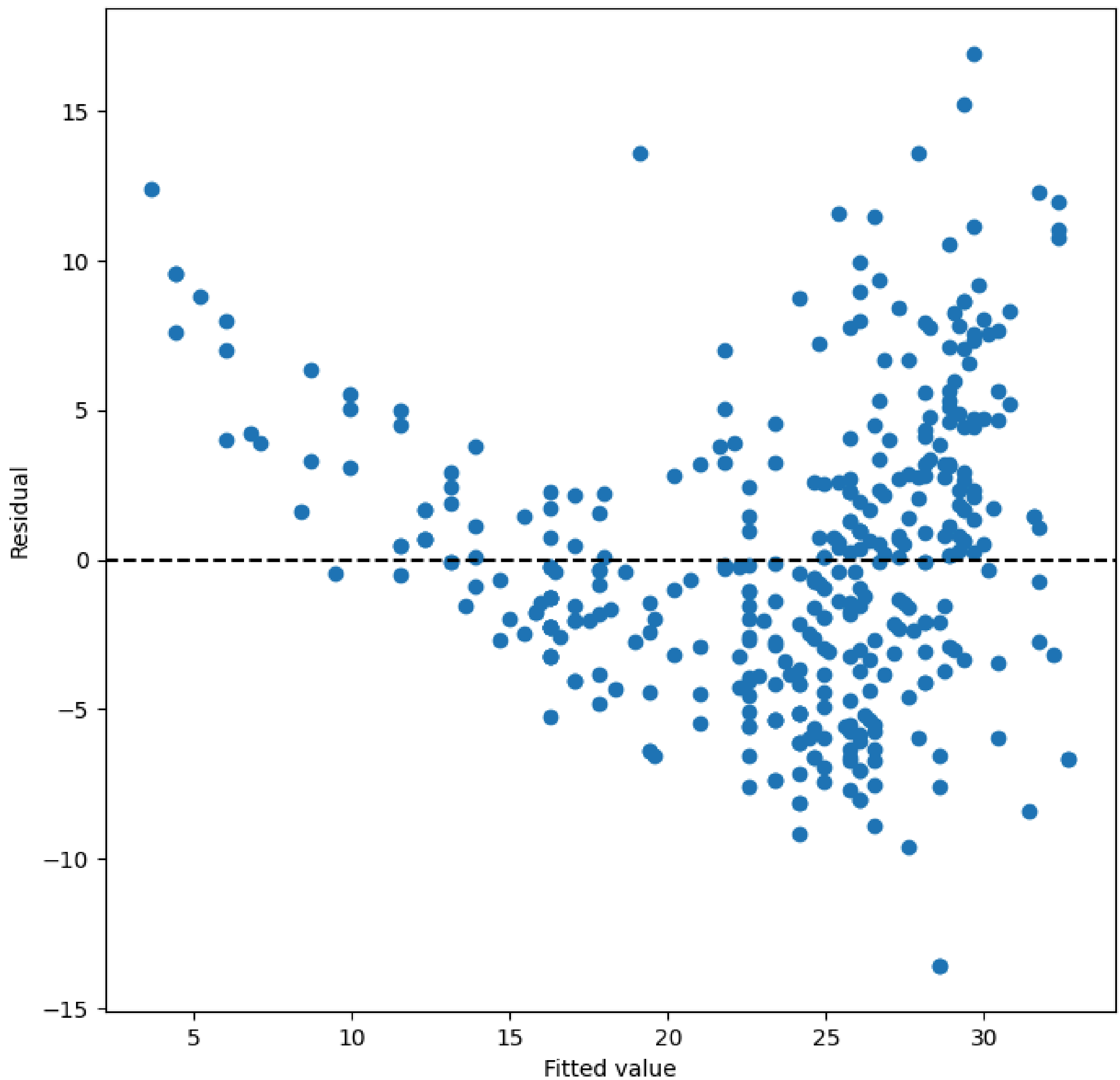
In [ ]:
```
ax = Auto.plot.scatter('horsepower', 'mpg')
abline(ax,
        results.params[0],
        results.params[1],
       'r--',
       linewidth=3)
```

```
<ipython-input-20-79ad2517a0a4>:3: FutureWarning: Series.__getitem__ treating keys
as positions is deprecated. In a future version, integer keys will always be treate
d as labels (consistent with DataFrame behavior). To access a value by position, us
e `ser.iloc[pos]`
  results.params[0],
<ipython-input-20-79ad2517a0a4>:4: FutureWarning: Series.__getitem__ treating keys
as positions is deprecated. In a future version, integer keys will always be treate
d as labels (consistent with DataFrame behavior). To access a value by position, us
e `ser.iloc[pos]`
  results.params[1],
```



In [ ]:

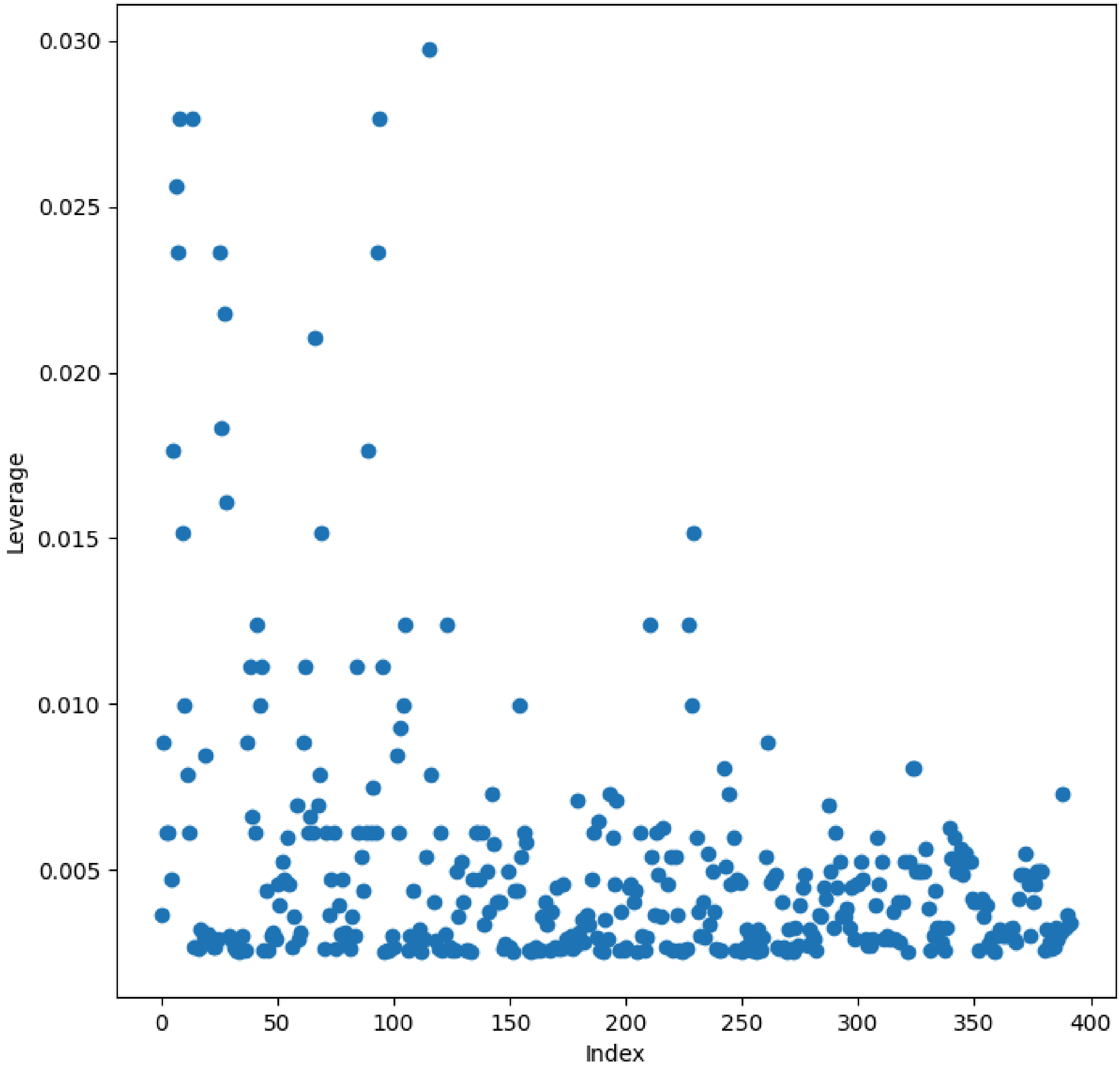
```
#8c
ax = subplots(figsize=(8,8))[1]
ax.scatter(results.fittedvalues , results.resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
```



In [ ]:

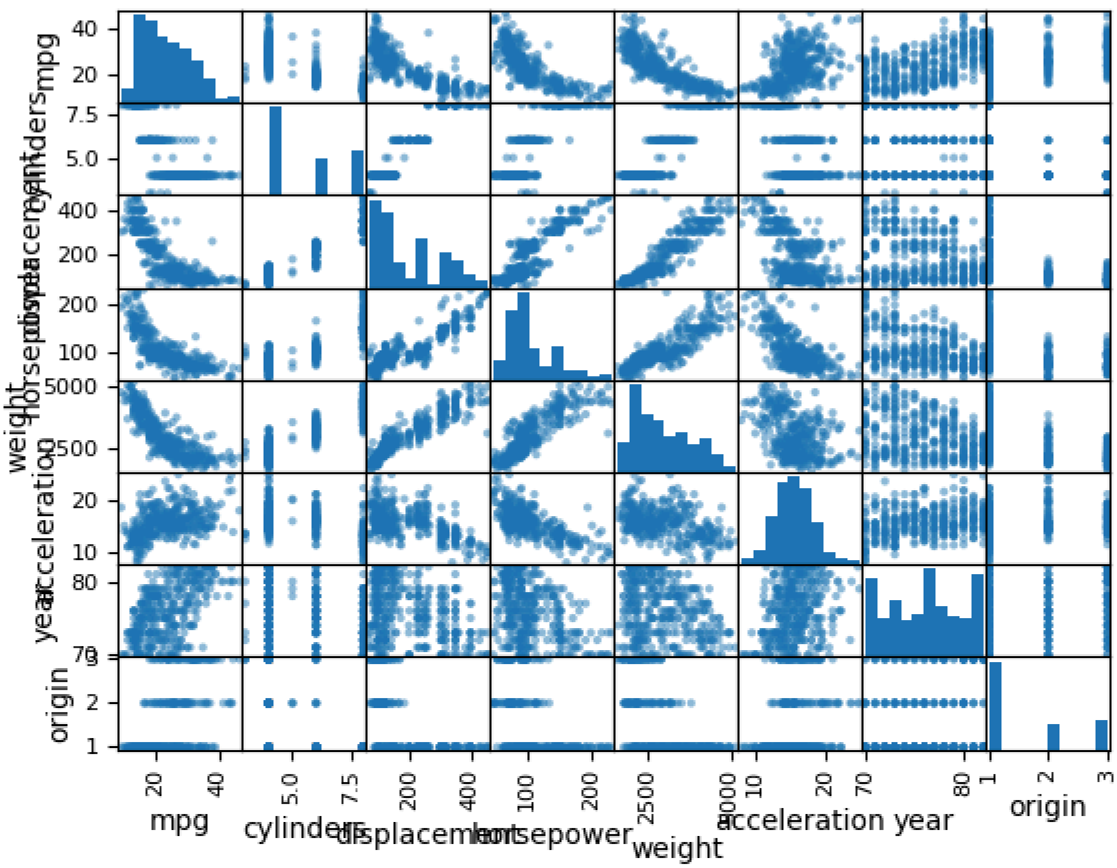
```
infl=results.get_influence()
ax = subplots(figsize=(8,8))[1]
ax.scatter(np.arange(X.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel('Index')
ax.set_ylabel('Leverage')
np.argmax(infl.hat_matrix_diag)
```

Out[ ]: 115



8c) I noticed that the residuals have heteroscedascity, and there are lots of leverage values higher than (p+1)/n=2/392=0.005.

In [ ]:
```
#9a
pd.plotting.scatter_matrix(Auto);
```



In [ ]:
```
#9b
print(Auto.corr())
```

```
                   mpg  cylinders  displacement  horsepower    weight  \
mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244
cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527
displacement -0.805127   0.950823      1.000000    0.897257  0.932994
horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538
weight       -0.832244   0.897527      0.932994    0.864538  1.000000
acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839
year          0.580541  -0.345647     -0.369855   -0.416361 -0.309120
origin        0.565209  -0.568932     -0.614535   -0.455171 -0.585005

              acceleration      year    origin
mpg               0.423329  0.580541  0.565209
cylinders        -0.504683 -0.345647 -0.568932
displacement     -0.543800 -0.369855 -0.614535
horsepower       -0.689196 -0.416361 -0.455171
weight           -0.416839 -0.309120 -0.585005
acceleration      1.000000  0.290316  0.212746
year              0.290316  1.000000  0.181528
origin            0.212746  0.181528  1.000000
```

In [ ]:
```
Auto.columns
```

Out[ ]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'year', 'origin'],
      dtype='object')
```

In [ ]:
```
#9c
allvars = list(Auto.columns.drop('mpg'))
y = Auto['mpg']
```

```
final = allvars
X = MS(final).fit_transform(Auto)
model = sm.OLS(y, X)
summarize(model.fit())
```

Out[ ]:

	coef	std err	t	P>|t|
intercept	-17.2184	4.644	-3.707	0.000
cylinders	-0.4934	0.323	-1.526	0.128
displacement	0.0199	0.008	2.647	0.008
horsepower	-0.0170	0.014	-1.230	0.220
weight	-0.0065	0.001	-9.929	0.000
acceleration	0.0806	0.099	0.815	0.415
year	0.7508	0.051	14.729	0.000
origin	1.4261	0.278	5.127	0.000

In [ ]:

```
anova_lm(results,model.fit())
```

Out[ ]:

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	390.0	9385.915872	0.0	NaN	NaN	NaN
1	384.0	4252.212530	6.0	5133.703341	77.267308	5.376746e-63

In [ ]:

```
model.fit().summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	mpg	**R-squared:**	0.821
Model:	OLS	**Adj. R-squared:**	0.818
Method:	Least Squares	**F-statistic:**	252.4
Date:	Tue, 18 Feb 2025	**Prob (F-statistic):**	2.04e-139
Time:	00:14:22	**Log-Likelihood:**	-1023.5
No. Observations:	392	**AIC:**	2063.
Df Residuals:	384	**BIC:**	2095.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-17.2184	4.644	-3.707	0.000	-26.350	-8.087
cylinders	-0.4934	0.323	-1.526	0.128	-1.129	0.142
displacement	0.0199	0.008	2.647	0.008	0.005	0.035
horsepower	-0.0170	0.014	-1.230	0.220	-0.044	0.010
weight	-0.0065	0.001	-9.929	0.000	-0.008	-0.005
acceleration	0.0806	0.099	0.815	0.415	-0.114	0.275
year	0.7508	0.051	14.729	0.000	0.651	0.851
origin	1.4261	0.278	5.127	0.000	0.879	1.973

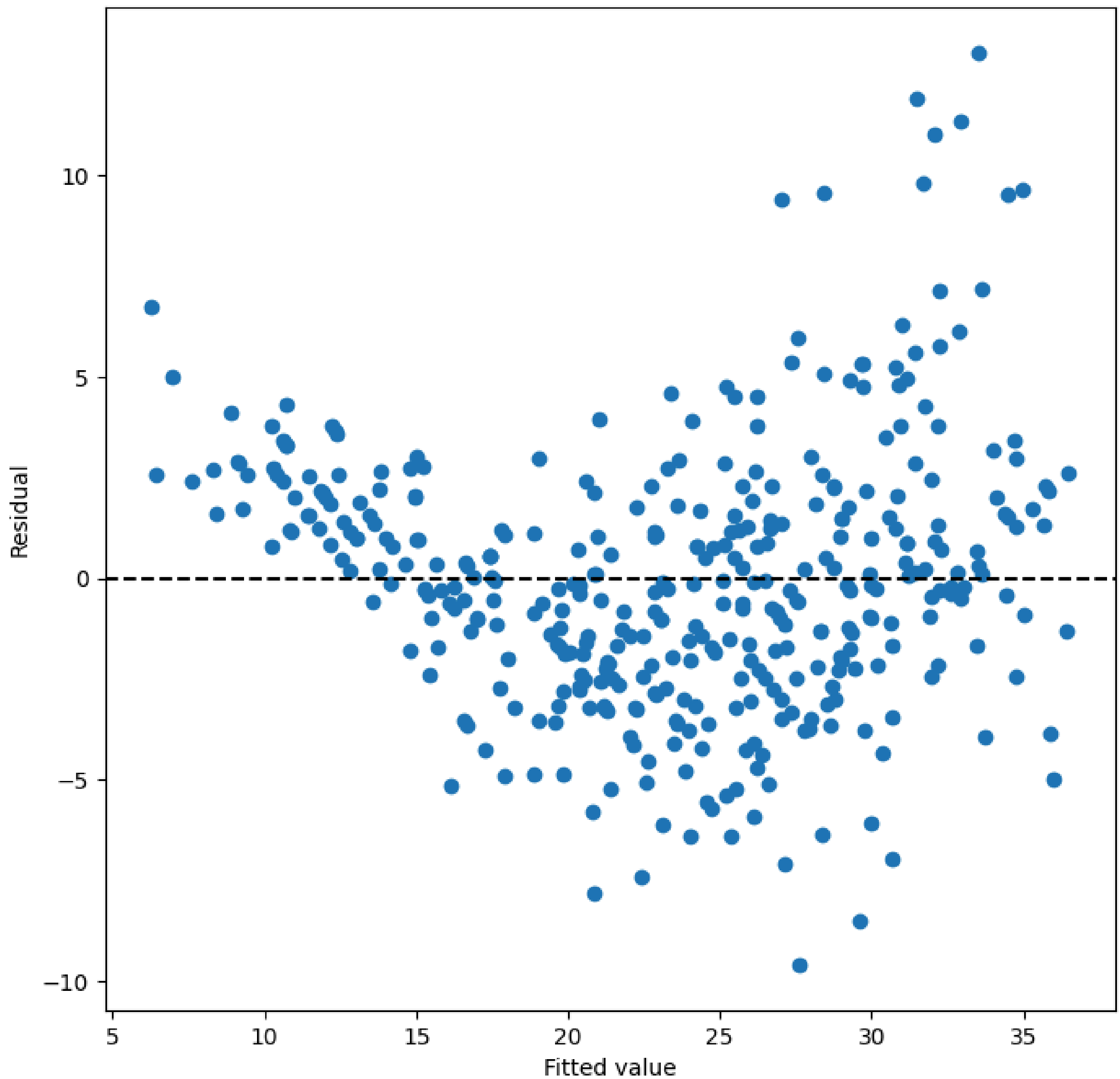
Omnibus:	31.906	**Durbin-Watson:**	1.309
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	53.100
Skew:	0.529	**Prob(JB):**	2.95e-12
Kurtosis:	4.460	**Cond. No.**	8.59e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.59e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

9c) Some of these predictors have a t-value greater than 0.025, showing insignificance.
i)Since the p-value in anova_lm is near 0, we can conclude the bigger model is superior and that there seems to be a relationship between the response and the predictors.
ii)Only displacement, weight,year, and origin are statistically significant.
iii)The coefficient for "year" suggests that each increase in (single) car model year, holding all other predictors constant, the mpg for a car increases by 0.7508.\

In [ ]:

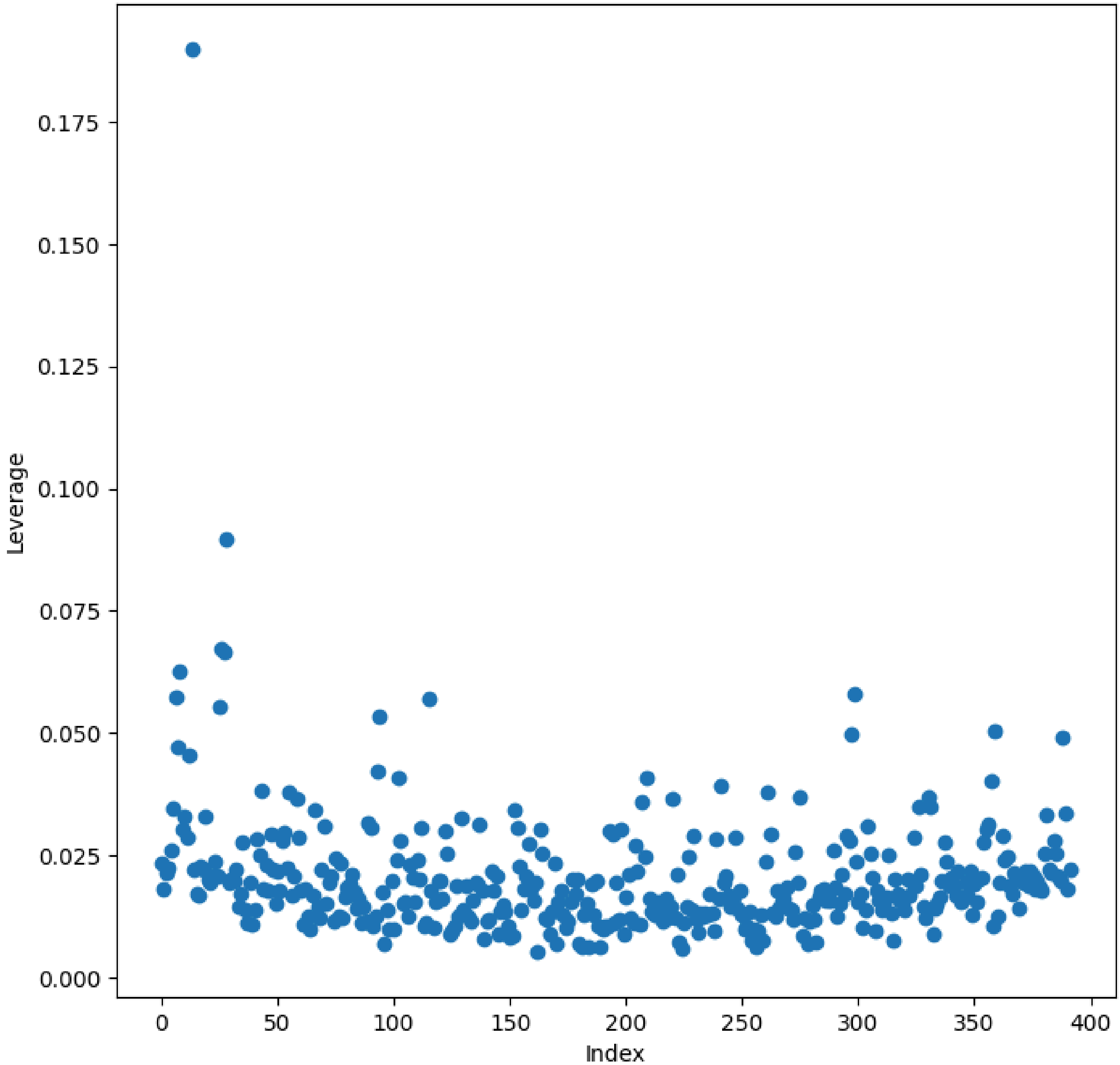
```
#9d
ax = subplots(figsize=(8,8))[1]
ax.scatter(model.fit().fittedvalues , model.fit().resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
```



In [ ]:

```
infl=model.fit().get_influence()
ax = subplots(figsize=(8,8))[1]
ax.scatter(np.arange(X.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel('Index')
ax.set_ylabel('Leverage')
np.argmax(infl.hat_matrix_diag)
```

Out[ ]: 13



9d)I notice that there are some residuals of -10 or 10 (showing high outliers), and that there is one extremely high leverage point.

In [ ]:
```
#9e
#After some changing of predictors (based on collinearity), I found this model whe
final1 = allvars + [('horsepower','acceleration'),('displacement','weight','cylind
X1 = MS(final).fit_transform(Auto)
model1 = sm.OLS(y, X1)
model1.fit().summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	mpg	**R-squared:**	0.821
Model:	OLS	**Adj. R-squared:**	0.818
Method:	Least Squares	**F-statistic:**	252.4
Date:	Tue, 18 Feb 2025	**Prob (F-statistic):**	2.04e-139
Time:	00:14:23	**Log-Likelihood:**	-1023.5
No. Observations:	392	**AIC:**	2063.
Df Residuals:	384	**BIC:**	2095.
Df Model:	7		
Covariance Type:	nonrobust		

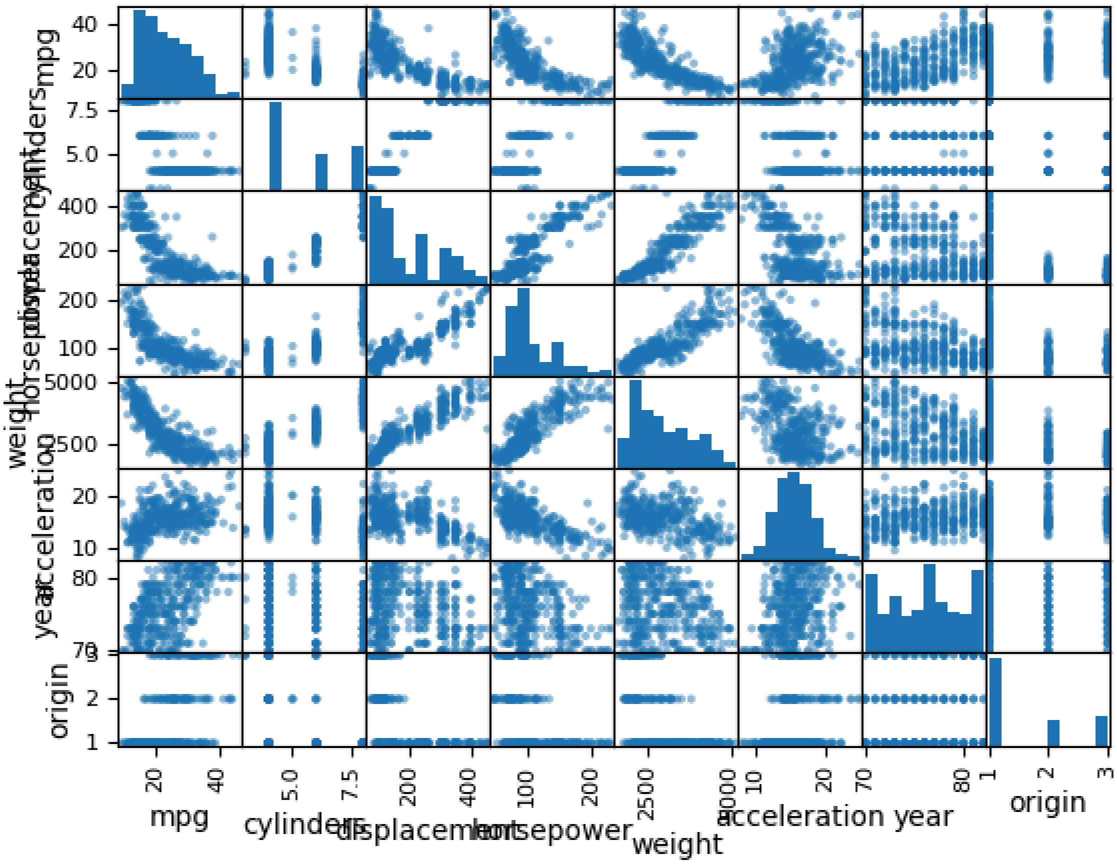
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	-17.2184	4.644	-3.707	0.000	-26.350	-8.087
cylinders	-0.4934	0.323	-1.526	0.128	-1.129	0.142
displacement	0.0199	0.008	2.647	0.008	0.005	0.035
horsepower	-0.0170	0.014	-1.230	0.220	-0.044	0.010
weight	-0.0065	0.001	-9.929	0.000	-0.008	-0.005
acceleration	0.0806	0.099	0.815	0.415	-0.114	0.275
year	0.7508	0.051	14.729	0.000	0.651	0.851
origin	1.4261	0.278	5.127	0.000	0.879	1.973

Omnibus:	31.906	**Durbin-Watson:**	1.309
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	53.100
Skew:	0.529	**Prob(JB):**	2.95e-12
Kurtosis:	4.460	**Cond. No.**	8.59e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.59e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

```
pd.plotting.scatter_matrix(Auto);
```

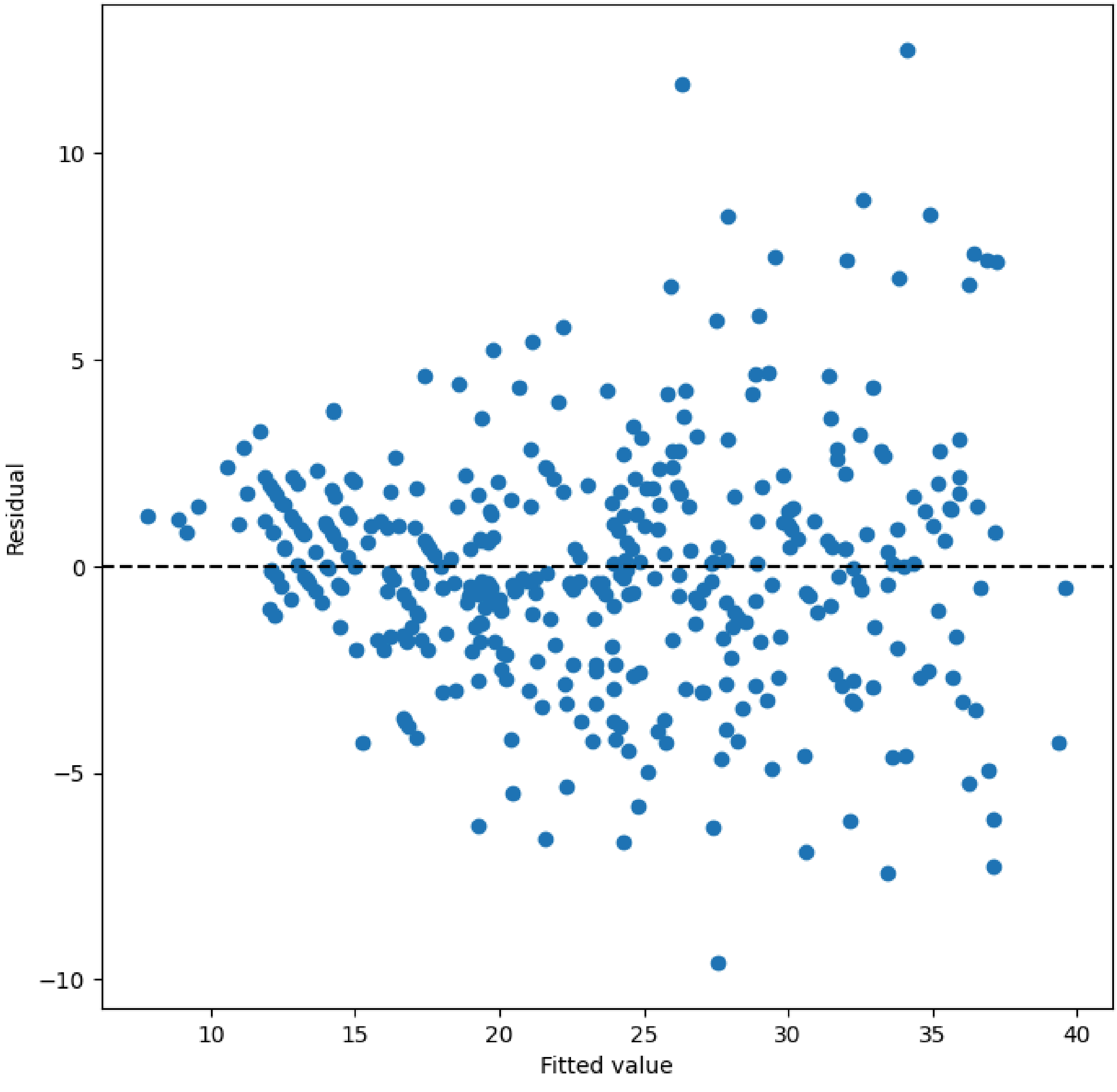


In [ ]:
```
print(Auto.corr())
```

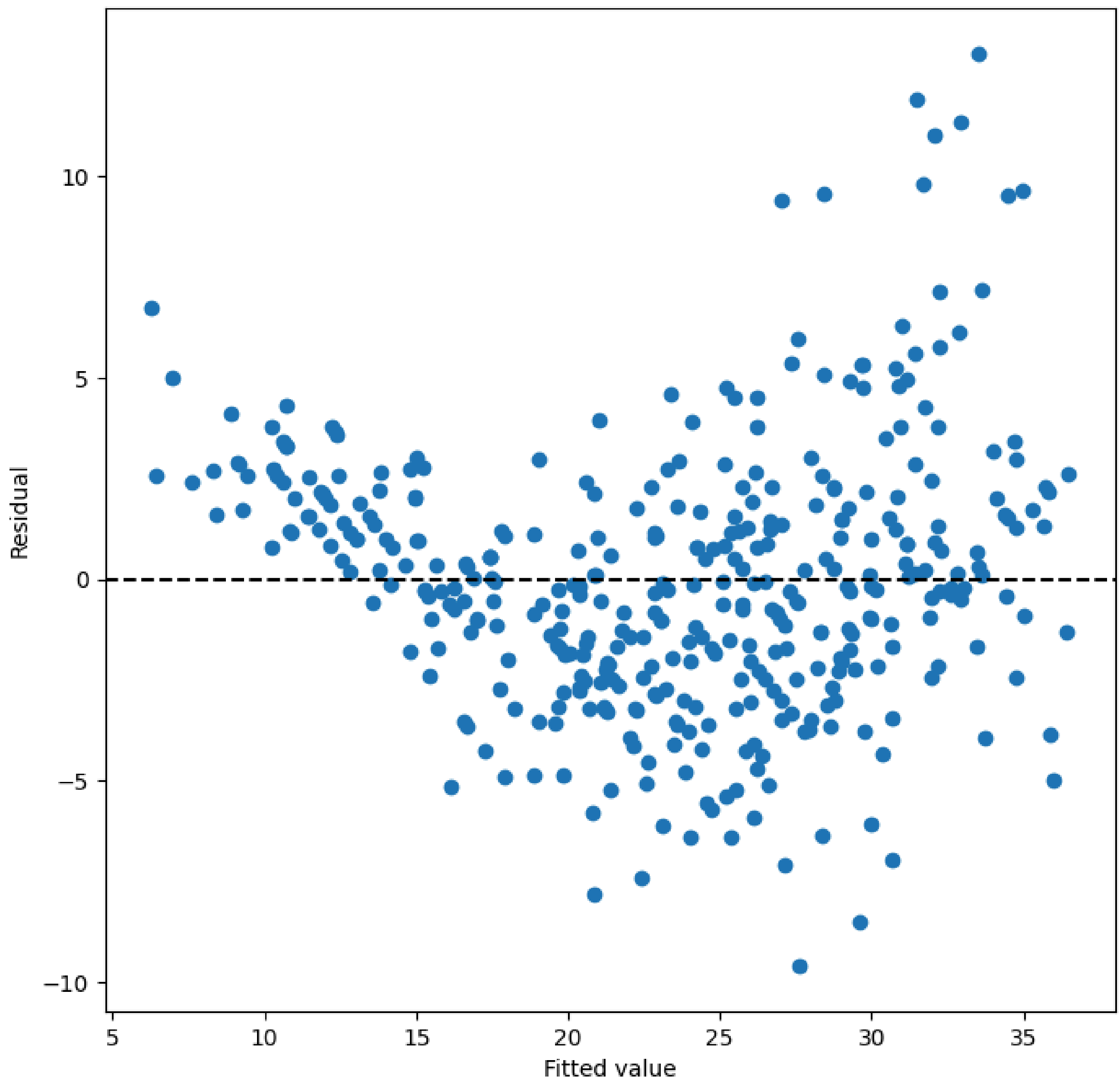
```
                   mpg  cylinders  displacement  horsepower    weight  \
mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244   
cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527   
displacement -0.805127   0.950823      1.000000    0.897257  0.932994   
horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538   
weight       -0.832244   0.897527      0.932994    0.864538  1.000000   
acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839   
year          0.580541  -0.345647     -0.369855   -0.416361 -0.309120   
origin        0.565209  -0.568932     -0.614535   -0.455171 -0.585005   

              acceleration      year    origin  
mpg               0.423329  0.580541  0.565209  
cylinders        -0.504683 -0.345647 -0.568932  
displacement     -0.543800 -0.369855 -0.614535  
horsepower       -0.689196 -0.416361 -0.455171  
weight           -0.416839 -0.309120 -0.585005  
acceleration      1.000000  0.290316  0.212746  
year              0.290316  1.000000  0.181528  
origin            0.212746  0.181528  1.000000  
```

In [37]:
```
#9f
Autonew=Auto
Autonew['transdisplacement']=1/(Auto['displacement'])
Autonew['transcylinders']=1/(Auto['cylinders'])
Autonew['transweight']=-1/(Auto['weight'])
Autonew['transacceleration']=1/(Auto['acceleration'])
Autonew['transhorsepower']=-1/(Auto['horsepower'])
Autonew['transorigin']=np.log(Auto['origin'])
Autonew['transyear']=np.log(Auto['year'])
Autonew=Autonew.drop(['displacement','cylinders','weight','acceleration','horsepow
```

```
y2 = Autonew['mpg']
allvars2 = list(Autonew.columns.drop(['mpg']))
final2 = allvars2
X2 = MS(final2).fit_transform(Autonew)
modelnew = sm.OLS(y2, X2)
ax = subplots(figsize=(8,8))[1]
ax.scatter(modelnew.fit().fittedvalues , modelnew.fit().resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
#This was the best I could do, but I was able to clump the data together.
```



In [ ]:
```
ax = subplots(figsize=(8,8))[1]
ax.scatter(model.fit().fittedvalues , model.fit().resid)
ax.set_xlabel('Fitted value')
ax.set_ylabel('Residual')
ax.axhline(0, c='k', ls='--');
```



Residual
10
5
0
−5
−10
5
10
15
20
25
30
35
Fitted value

Ch. 3 - Q10

In [1]:
```
import ISLP
import pandas as pd
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

In [2]:
```
# Load in data
Carseats = ISLP.load_data("Carseats")
Carseats.head()
```

Out[2]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Ur
0	9.50	138	73	11	276	120	Bad	42	17	
1	11.22	111	48	16	260	83	Good	65	10	
2	10.06	113	35	10	269	80	Medium	59	12	
3	7.40	117	100	4	466	97	Medium	55	14	
4	4.15	141	64	3	340	128	Bad	38	13	

In [3]:
```
# Assign design matrix and target vector
X = Carseats[['Price', 'Urban', 'US']].copy()
X[['Urban', 'US']] = X[['Urban', 'US']].apply(lambda col: col.map({'Yes': 1, 'No':
y = Carseats['Sales']
```

Part (a)

In [4]:
```
model = LinearRegression()
model.fit(X, y)

pd.DataFrame({'Variable': ['Intercept'] + list(X.columns),
              'Coefficient': [model.intercept_] + list(model.coef_)})
```

Out[4]:

	Variable	Coefficient
0	Intercept	13.043469
1	Price	-0.054459
2	Urban	-0.021916
3	US	1.200573

Part (b)

Keep in mind that the units of `Sales` *are in thousands.*

- The coefficient for "Price" means that, on average, increasing the price by $1 decreases sales by 54.46 units, assuming all other factors stay the same.

- The coefficient for "Urban" means that, on average, sales in urban locations are 21.92 units lower than in rural locations, keeping all other factors the same.
- The coefficient for "US" means that, on average, sales in US stores are 1,200.57 units higher than in non-US stores, assuming all other factors remain unchanged.

Part (c)

In [5]:
```
equation = f"Sales = {model.intercept_:.2f}"
for coef, col in zip(model.coef_, X.columns):
    if coef >= 0: equation += f" + {coef:.2f} * {col}"
    else: equation += f" - {-coef:.2f} * {col}"
print(equation)
```

```
Sales = 13.04 - 0.05 * Price - 0.02 * Urban + 1.20 * US
```

Part (d)

In [6]:
```
# Recreating linear model in statsmodels because apparently
#   sci-kit learn doesn't provide p-values :)
X_with_intercept = sm.add_constant(X)
full_model = sm.OLS(y, X_with_intercept).fit()
full_model.pvalues
```

Out[6]:
```
const     3.626602e-62
Price     1.609917e-22
Urban     9.357389e-01
US        4.860245e-06
dtype: float64
```

- We can reject the null hypothesis for the variables, `Price` and `US`.

Part (e)

In [7]:
```
reduced_X = Carseats[['Price', 'US']].copy()
reduced_X['US'] = reduced_X['US'].map({'Yes': 1, 'No': 0})

reduced_X_with_intercept = sm.add_constant(reduced_X)
reduced_model = sm.OLS(y, reduced_X_with_intercept).fit()

print(f"Full Model - R^2:\t{full_model.rsquared}")
print(f"Reduced Model - R^2:\t{reduced_model.rsquared}")
```

```
Full Model - R^2:       0.2392753921840549
Reduced Model - R^2:    0.23926288842678567
```

Part (f)

- The two models are nearly identical. Thus the reduced model is likely the preferred option due to it's simplicity

Part (g)

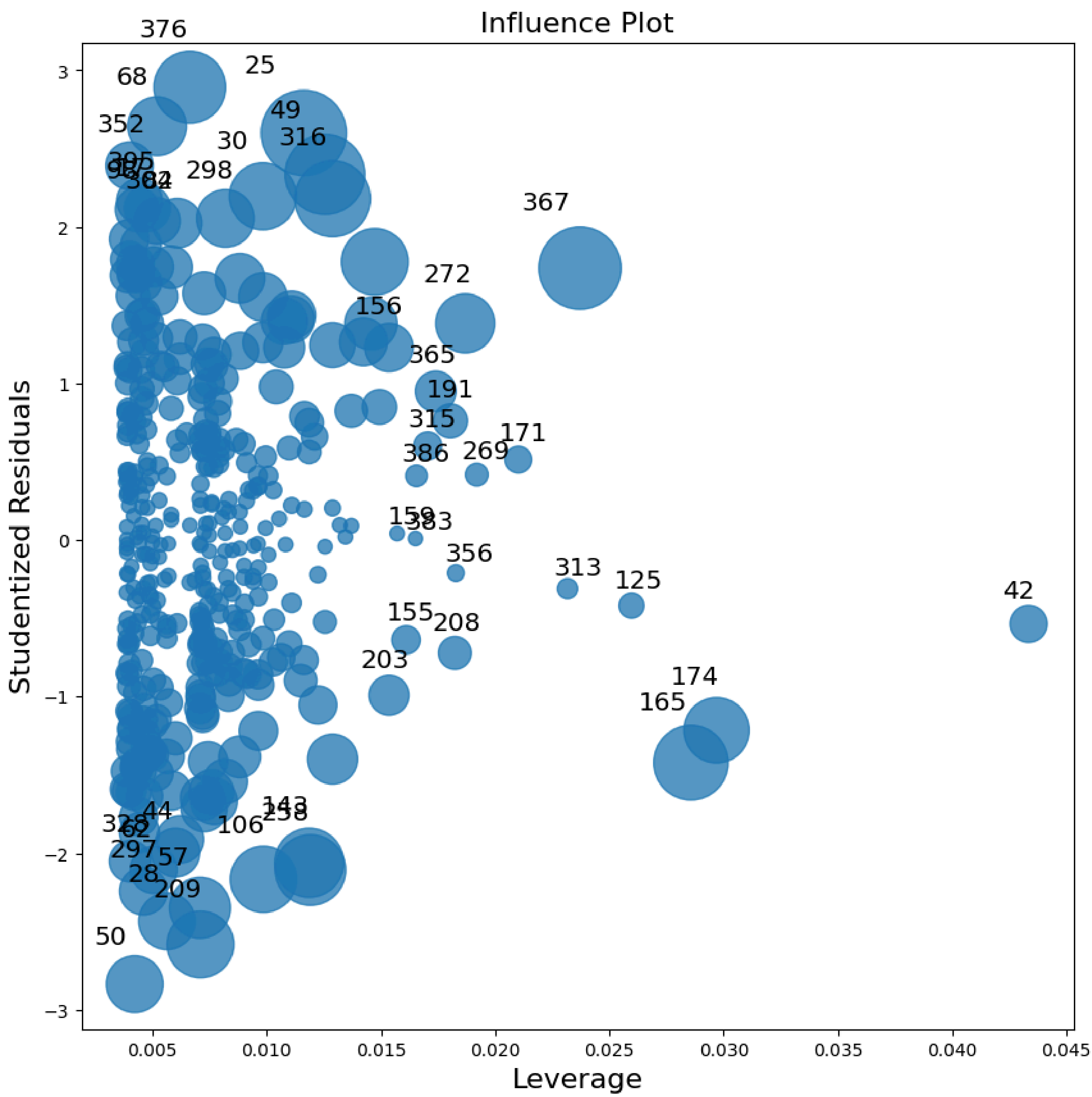
In [8]:
```python
confidence_intervals = reduced_model.conf_int(alpha=0.05)  # 95% CI
confidence_intervals.columns = ['Lower Bound', 'Upper Bound']
confidence_intervals
```

Out[8]:

	Lower Bound	Upper Bound
const	11.79032	14.271265
Price	-0.06476	-0.044195
US	0.69152	1.707766

Part (h)

In [9]:
```python
fig, ax = plt.subplots(figsize=(10, 10))
sm.graphics.influence_plot(reduced_model, ax=ax)
plt.show()
```



In [10]:
```
n, p = reduced_X_with_intercept.shape
print(f"Threshold: {(p) / n}")
```

```
Threshold: 0.0075
```

Outliers and High Leverage Points

The plot above shows that there are no outliers as all residuals are within ± 3 standard deviations, however some observations come close.

Additionally, many points have high leverage because their leverage values exceed the threshold 0.0075, which is calculated as:

$$\frac{p+1}{n} = \frac{3}{400} = 0.0075$$

where $p = 2$ is the number of predictors and $n = 400$ is the number of observations. However, likewise, these points are not outliers.

STAT 702 - Homework 2

Noah Javadi

2025-02-16

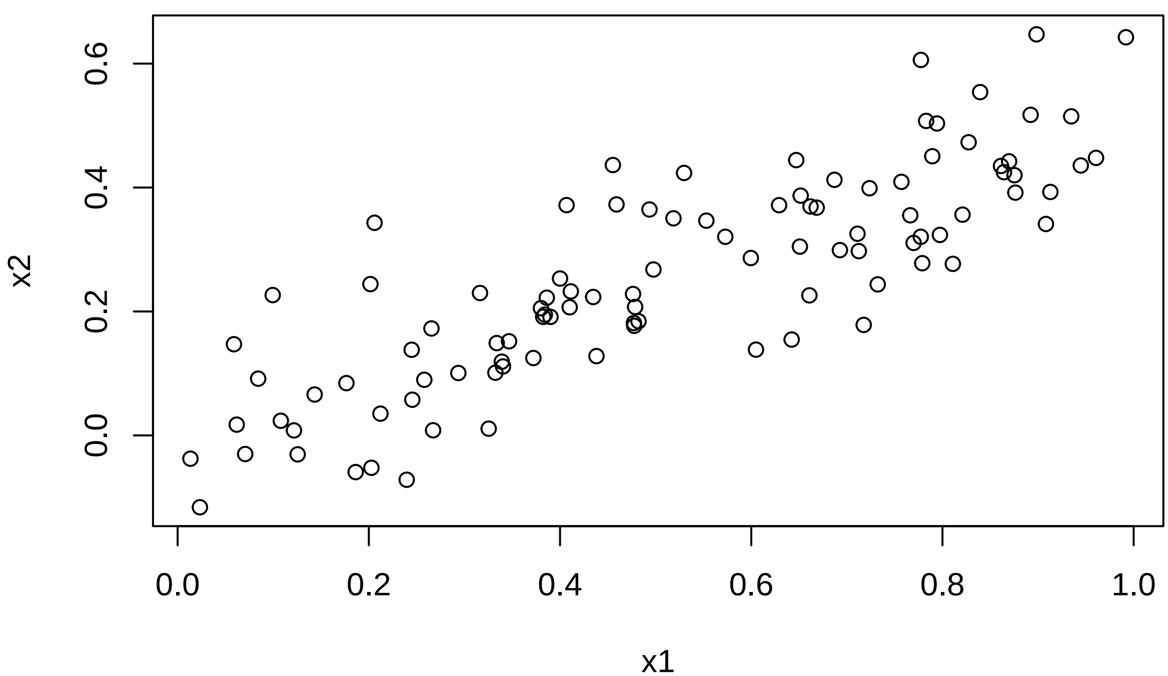
Setup

Problem 14

```
set.seed(1)
x1 <- runif(100)
x2 <- 0.5 * x1 + rnorm(100) / 10
y <- 2 + 2 * x1 + 0.3 * x2 + rnorm(100)

#Correlation and scatterplot
cor(x1,x2)
```

```
## [1] 0.8351212
```

```
plot(x1,x2)
```



```r
#Fitting full linear regression model
lm.p14 <- lm(y ~ x1 + x2)
summary(lm.p14)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8311 -0.7273 -0.0537  0.6338  2.3359
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1305     0.2319   9.188 7.61e-15 ***
## x1            1.4396     0.7212   1.996   0.0487 *
## x2            1.0097     1.1337   0.891   0.3754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.056 on 97 degrees of freedom
## Multiple R-squared:  0.2088, Adjusted R-squared:  0.1925
## F-statistic:  12.8 on 2 and 97 DF,  p-value: 1.164e-05
```

```r
#Fitting simple linear regression model with x1
lm.x1 <- lm(y ~ x1)
summary(lm.x1)
```

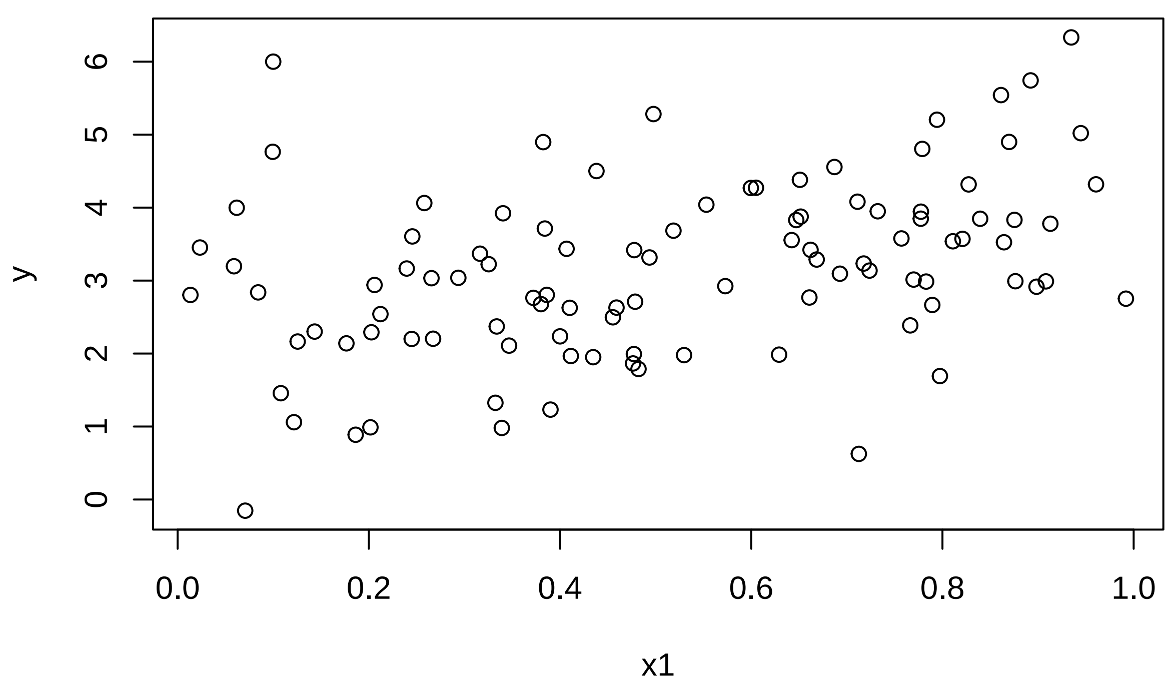
```
##
## Call:
## lm(formula = y ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89495 -0.66874 -0.07785  0.59221  2.45560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1124     0.2307   9.155 8.27e-15 ***
## x1            1.9759     0.3963   4.986 2.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.055 on 98 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.1942
## F-statistic: 24.86 on 1 and 98 DF,  p-value: 2.661e-06
```

```r
#Fitting simple linear regression model with x2
lm.x2 <- lm(y ~ x2)
summary(lm.x2)
```

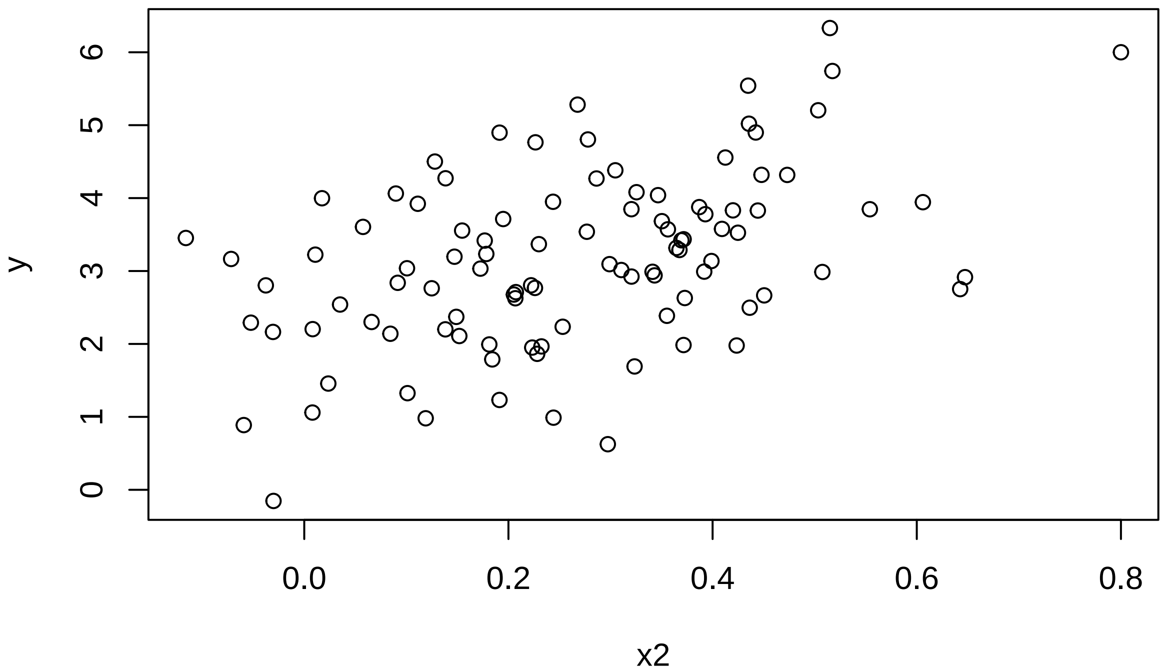
```
##
```

```
## Call:
## lm(formula = y ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## x2            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

```r
#Adding new values
x1 <- c(x1,0.1)
x2 <- c(x2,0.8)
y <- c(y,6)

#Plot new variables to determine relationship
plot(x1,y)
```



```r
plot(x2,y)
```




```r
#Refitting full model
lm.p14 <- lm(y ~ x1 + x2)
summary(lm.p14)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73348 -0.69318 -0.05263  0.66385  2.30619
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.2267     0.2314   9.624 7.91e-16 ***
## x1            0.5394     0.5922   0.911  0.36458
## x2            2.5146     0.8977   2.801  0.00614 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.075 on 98 degrees of freedom
## Multiple R-squared:  0.2188, Adjusted R-squared:  0.2029
## F-statistic: 13.72 on 2 and 98 DF,  p-value: 5.564e-06
```

```r
#Refitting simple linear regression with x1
lm.x1 <- lm(y ~ x1)
summary(lm.x1)
```

```
##
## Call:
## lm(formula = y ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8897 -0.6556 -0.0909  0.5682  3.5665
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.2569     0.2390   9.445 1.78e-15 ***
## x1            1.7657     0.4124   4.282 4.29e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.111 on 99 degrees of freedom
## Multiple R-squared:  0.1562, Adjusted R-squared:  0.1477
## F-statistic: 18.33 on 1 and 99 DF,  p-value: 4.295e-05
```

```r
#Refitting simple linear regression with x2
lm.x2 <- lm(y ~ x2)
summary(lm.x2)
```

```
##
## Call:
## lm(formula = y ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.64729 -0.71021 -0.06899  0.72699  2.38074
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3451     0.1912  12.264  < 2e-16 ***
## x2            3.1190     0.6040   5.164 1.25e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.074 on 99 degrees of freedom
## Multiple R-squared:  0.2122, Adjusted R-squared:  0.2042
## F-statistic: 26.66 on 1 and 99 DF,  p-value: 1.253e-06
```

a) The regression coefficients are 2 for beta_0, 2 for beta_1, and 0.3 for beta_2.
b) 0.8424
c) The model with both x_1 and x_2 show a weak linear relationship to y. beta_hat_0 is 1.91, beta_hat_1 is 1.96, and beta_hat_2 is 0.549. The estimated coefficients should approach the true beta_0, beta_1, and beta_2. For beta_1, we can reject the null hypothesis where beta_1 = 0. For beta_2, we cannot reject the null hypothesis where beta_2 = 0.

d) This is a similar model to the full model previously analyzed. There is enough evidence to reject the null hypothesis for beta_1 = 0.
e) This is a weaker model to the full model previously analyzed. There is not enough evidence to reject the null hypothesis for beta_1 = 0.
f) The results obtained in c-e do not contradict each other. The correlation between x1 and x2 show that x1 is the main predictor and adding x2 does not add much to describing the variability of y.
g) We have introduced highly influential points to x1 and x2 which has changed the impact of x2 and the interpretation of each model. The point introduced to x1 is an outlier but not a high leverage point because it has a high residual, but not an extreme x value. Whereas the point introduced to x2 is not an outlier, but a high leverage point because it is in line with expected values, but has a large x value.

Problem 15

```
#?Boston

#Correlation matrix for all variables against crim
cor(Boston[-1],Boston$crim)
```

```
##                [,1]
## zn      -0.20046922
## indus    0.40658341
## chas    -0.05589158
## nox      0.42097171
## rm      -0.21924670
## age      0.35273425
## dis     -0.37967009
## rad      0.62550515
## tax      0.58276431
## ptratio  0.28994558
## lstat    0.45562148
## medv    -0.38830461
```

```
#Fitting simple linear regressions for each variable against crim
zn.lm <- lm(crim ~ zn,data = Boston)
indus.lm <- lm(crim ~ indus,data = Boston)
chas.lm <- lm(crim ~ chas,data = Boston)
nox.lm <- lm(crim ~ nox,data = Boston)
rm.lm <- lm(crim ~ rm,data = Boston)
age.lm <- lm(crim ~ age,data = Boston)
dis.lm <- lm(crim ~ dis,data = Boston)
rad.lm <- lm(crim ~ rad,data = Boston)
tax.lm <- lm(crim ~ tax,data = Boston)
ptratio.lm <- lm(crim ~ ptratio,data = Boston)
lstat.lm <- lm(crim ~ lstat,data = Boston)
medv.lm <- lm(crim ~ medv,data = Boston)

summary(zn.lm)
```

```
##
## Call:
## lm(formula = crim ~ zn, data = Boston)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max 
## -4.429 -4.222 -2.620  1.250 84.523 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  4.45369    0.41722  10.675  < 2e-16 ***
## zn          -0.07393    0.01609  -4.594 5.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 8.435 on 504 degrees of freedom
## Multiple R-squared:  0.04019,    Adjusted R-squared:  0.03828 
## F-statistic:  21.1 on 1 and 504 DF,  p-value: 5.506e-06
```

```r
summary(indus.lm)
```

```
## 
## Call:
## lm(formula = crim ~ indus, data = Boston)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -11.972  -2.698  -0.736   0.712  81.813 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.06374    0.66723  -3.093  0.00209 ** 
## indus        0.50978    0.05102   9.991  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 7.866 on 504 degrees of freedom
## Multiple R-squared:  0.1653, Adjusted R-squared:  0.1637 
## F-statistic: 99.82 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(chas.lm)
```

```
## 
## Call:
## lm(formula = crim ~ chas, data = Boston)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -3.738 -3.661 -3.435  0.018 85.232 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   3.7444     0.3961   9.453   <2e-16 ***
## chas         -1.8928     1.5061  -1.257    0.209    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 8.597 on 504 degrees of freedom
```

```
## Multiple R-squared:  0.003124,   Adjusted R-squared:  0.001146
## F-statistic: 1.579 on 1 and 504 DF,  p-value: 0.2094
```

```r
summary(nox.lm)
```

```
##
## Call:
## lm(formula = crim ~ nox, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.371  -2.738  -0.974   0.559  81.728
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -13.720      1.699  -8.073 5.08e-15 ***
## nox           31.249      2.999  10.419  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.81 on 504 degrees of freedom
## Multiple R-squared:  0.1772, Adjusted R-squared:  0.1756
## F-statistic: 108.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(rm.lm)
```

```
##
## Call:
## lm(formula = crim ~ rm, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.604 -3.952 -2.654  0.989 87.197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.482      3.365   6.088 2.27e-09 ***
## rm            -2.684      0.532  -5.045 6.35e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.401 on 504 degrees of freedom
## Multiple R-squared:  0.04807,    Adjusted R-squared:  0.04618
## F-statistic: 25.45 on 1 and 504 DF,  p-value: 6.347e-07
```

```r
summary(age.lm)
```

```
##
## Call:
## lm(formula = crim ~ age, data = Boston)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -6.789 -4.257 -1.230  1.527 82.849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.77791    0.94398  -4.002 7.22e-05 ***
## age          0.10779    0.01274   8.463 2.85e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.057 on 504 degrees of freedom
## Multiple R-squared:  0.1244, Adjusted R-squared:  0.1227
## F-statistic: 71.62 on 1 and 504 DF,  p-value: 2.855e-16
```

```
summary(dis.lm)
```

```
##
## Call:
## lm(formula = crim ~ dis, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.708 -4.134 -1.527  1.516 81.674
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.4993     0.7304  13.006   <2e-16 ***
## dis          -1.5509     0.1683  -9.213   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.965 on 504 degrees of freedom
## Multiple R-squared:  0.1441, Adjusted R-squared:  0.1425
## F-statistic: 84.89 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
summary(rad.lm)
```

```
##
## Call:
## lm(formula = crim ~ rad, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.164  -1.381  -0.141   0.660  76.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.28716    0.44348  -5.157 3.61e-07 ***
## rad          0.61791    0.03433  17.998  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.718 on 504 degrees of freedom
```

```
## Multiple R-squared:  0.3913, Adjusted R-squared:   0.39
## F-statistic: 323.9 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(tax.lm)
```

```
##
## Call:
## lm(formula = crim ~ tax, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.513  -2.738  -0.194   1.065  77.696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.528369   0.815809  -10.45   <2e-16 ***
## tax          0.029742   0.001847   16.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.997 on 504 degrees of freedom
## Multiple R-squared:  0.3396, Adjusted R-squared:  0.3383
## F-statistic: 259.2 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(ptratio.lm)
```

```
##
## Call:
## lm(formula = crim ~ ptratio, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.654 -3.985 -1.912  1.825 83.353
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.6469     3.1473  -5.607 3.40e-08 ***
## ptratio       1.1520     0.1694   6.801 2.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.24 on 504 degrees of freedom
## Multiple R-squared:  0.08407,    Adjusted R-squared:  0.08225
## F-statistic: 46.26 on 1 and 504 DF,  p-value: 2.943e-11
```

```r
summary(lstat.lm)
```

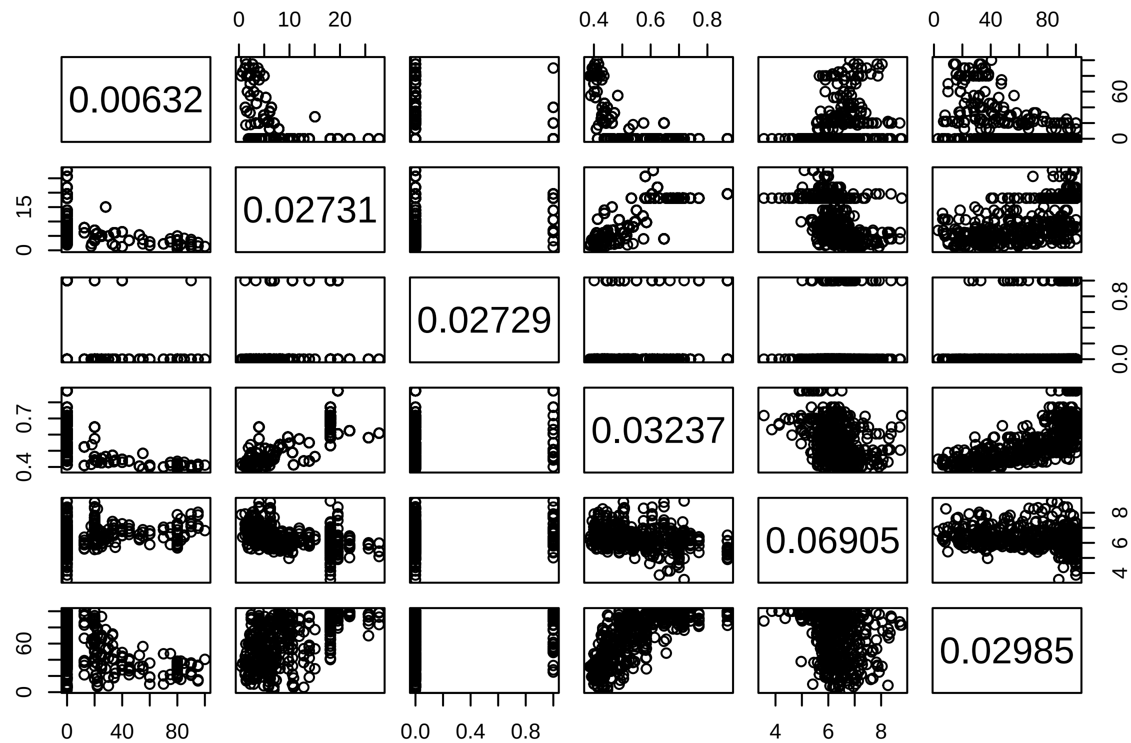
```
##
## Call:
## lm(formula = crim ~ lstat, data = Boston)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max 
## -13.925  -2.822  -0.664   1.079  82.862 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -3.33054    0.69376  -4.801 2.09e-06 ***
## lstat        0.54880    0.04776  11.491  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 7.664 on 504 degrees of freedom
## Multiple R-squared:  0.2076, Adjusted R-squared:  0.206 
## F-statistic:   132 on 1 and 504 DF,  p-value: < 2.2e-16
```

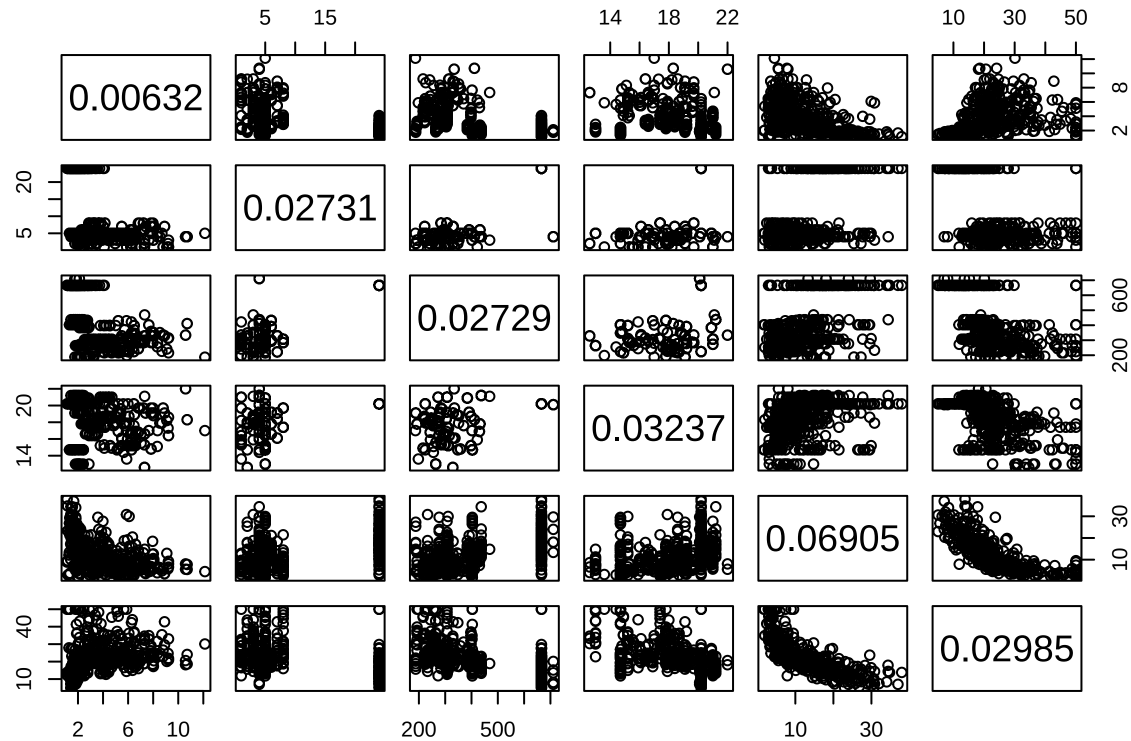
```r
summary(medv.lm)
```

```
## 
## Call:
## lm(formula = crim ~ medv, data = Boston)
## 
## Residuals:
##    Min     1Q Median     3Q    Max 
## -9.071 -4.022 -2.343  1.298 80.957 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 11.79654    0.93419   12.63   <2e-16 ***
## medv        -0.36316    0.03839   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 7.934 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491 
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
#Plots to confirm observations and linear relationships for each model against crim
plot(Boston[c(2:7)],Boston$crim)
```




```r
plot(Boston[c(8:13)],Boston$crim)
```

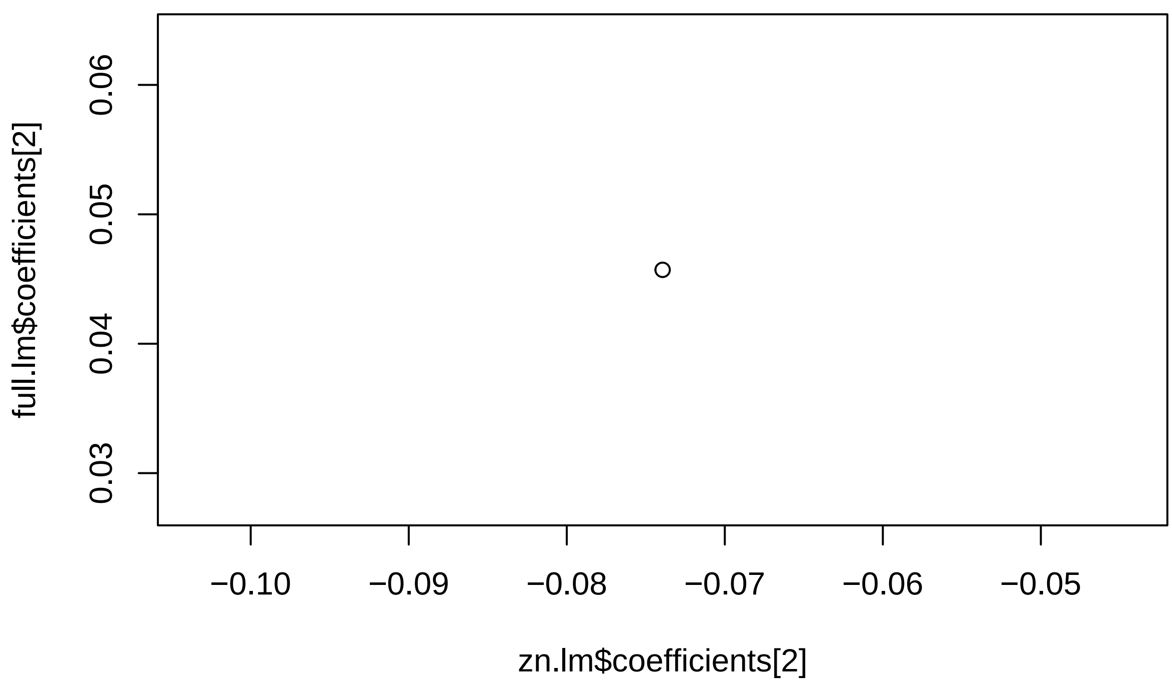



```r
#Fitting linear model for crim against all variables in data set
full.lm <- lm(crim ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + lstat + medv,data

#Summary of the full model
summary(full.lm)
```

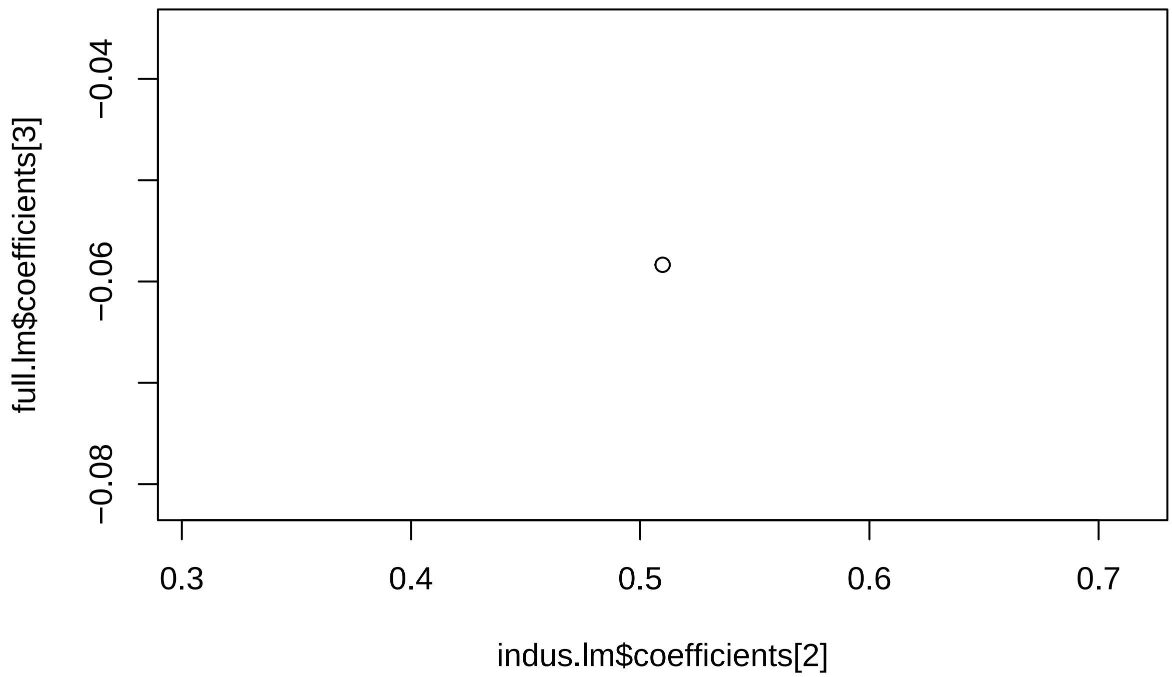
```
##
## Call:
## lm(formula = crim ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + lstat + medv, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.534 -2.248 -0.348  1.087 73.923
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.7783938  7.0818258   1.946 0.052271 .
## zn           0.0457100  0.0187903   2.433 0.015344 *
## indus       -0.0583501  0.0836351  -0.698 0.485709
## chas        -0.8253776  1.1833963  -0.697 0.485841
## nox         -9.9575865  5.2898242  -1.882 0.060370 .
## rm           0.6289107  0.6070924   1.036 0.300738
## age         -0.0008483  0.0179482  -0.047 0.962323
## dis         -1.0122467  0.2824676  -3.584 0.000373 ***
## rad          0.6124653  0.0875358   6.997 8.59e-12 ***
```

```
## tax         -0.0037756  0.0051723  -0.730 0.465757
## ptratio     -0.3040728  0.1863598  -1.632 0.103393
## lstat        0.1388006  0.0757213   1.833 0.067398 .
## medv        -0.2200564  0.0598240  -3.678 0.000261 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.46 on 493 degrees of freedom
## Multiple R-squared:  0.4493, Adjusted R-squared:  0.4359
## F-statistic: 33.52 on 12 and 493 DF,  p-value: < 2.2e-16
```

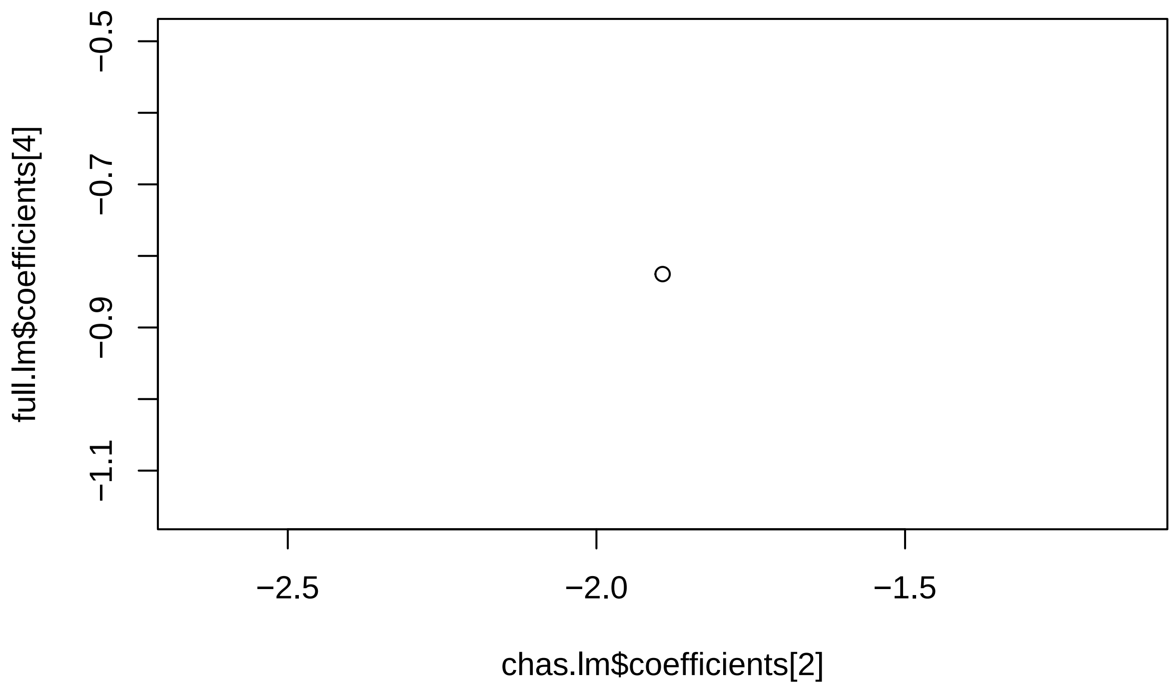
```r
#Plots comparing univariate model coefficients against the full model coefficients
plot(zn.lm$coefficients[2],full.lm$coefficients[2])
```



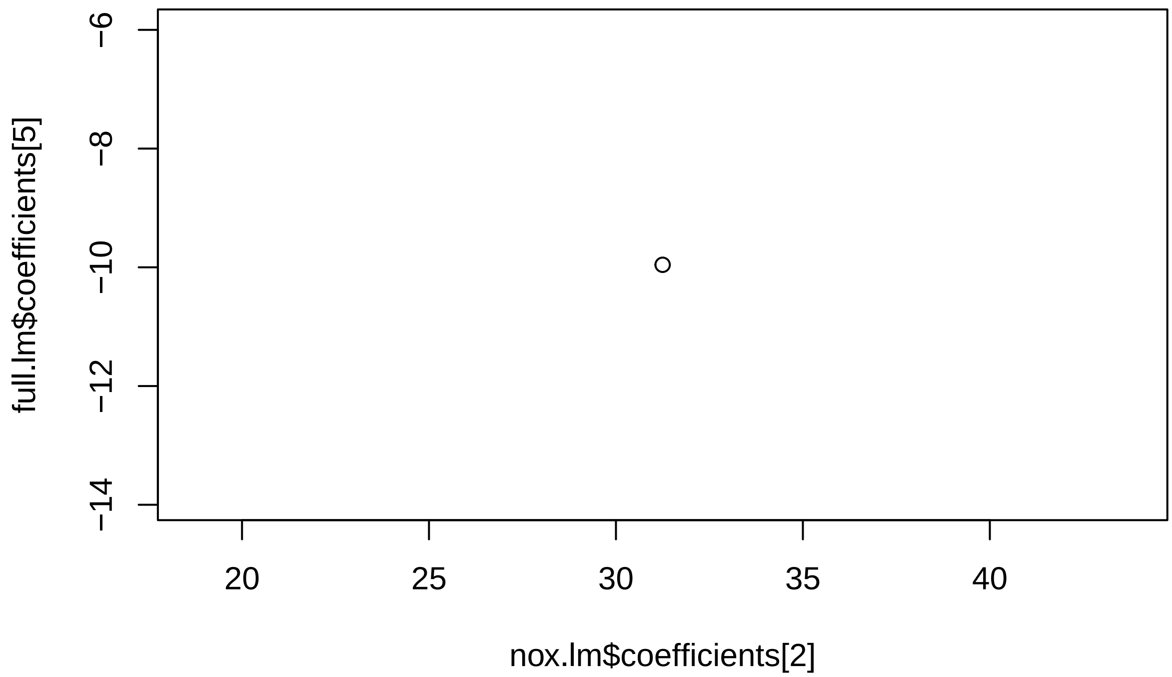

```r
plot(indus.lm$coefficients[2],full.lm$coefficients[3])
```



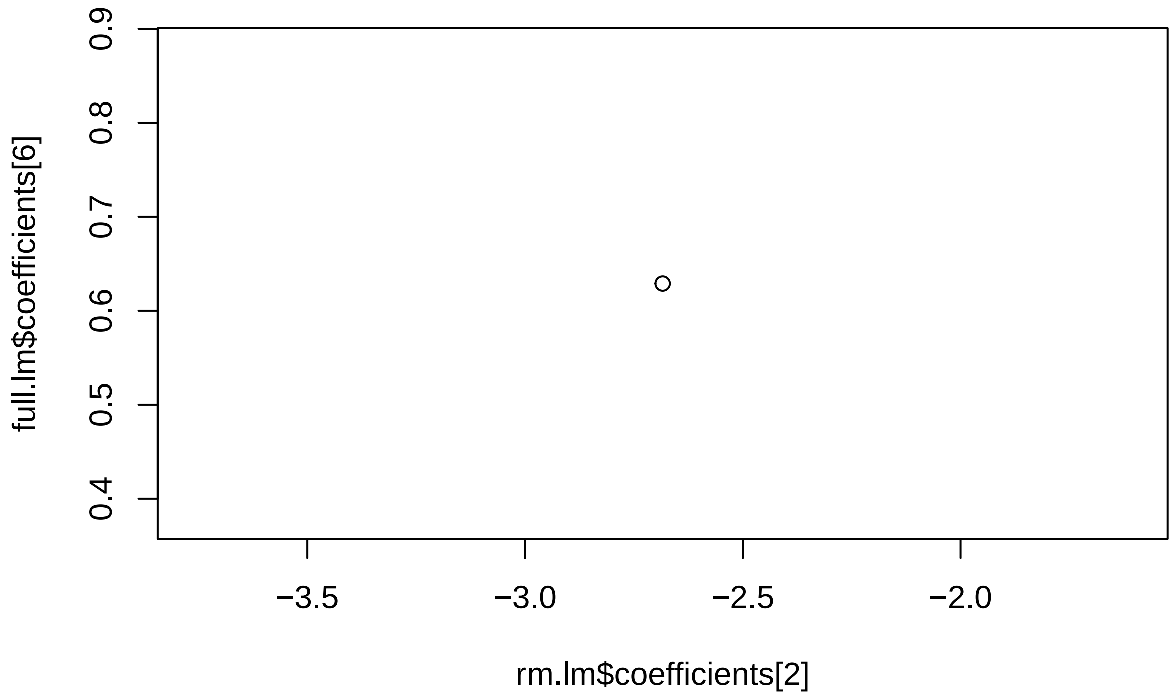

```
plot(chas.lm$coefficients[2],full.lm$coefficients[4])
```



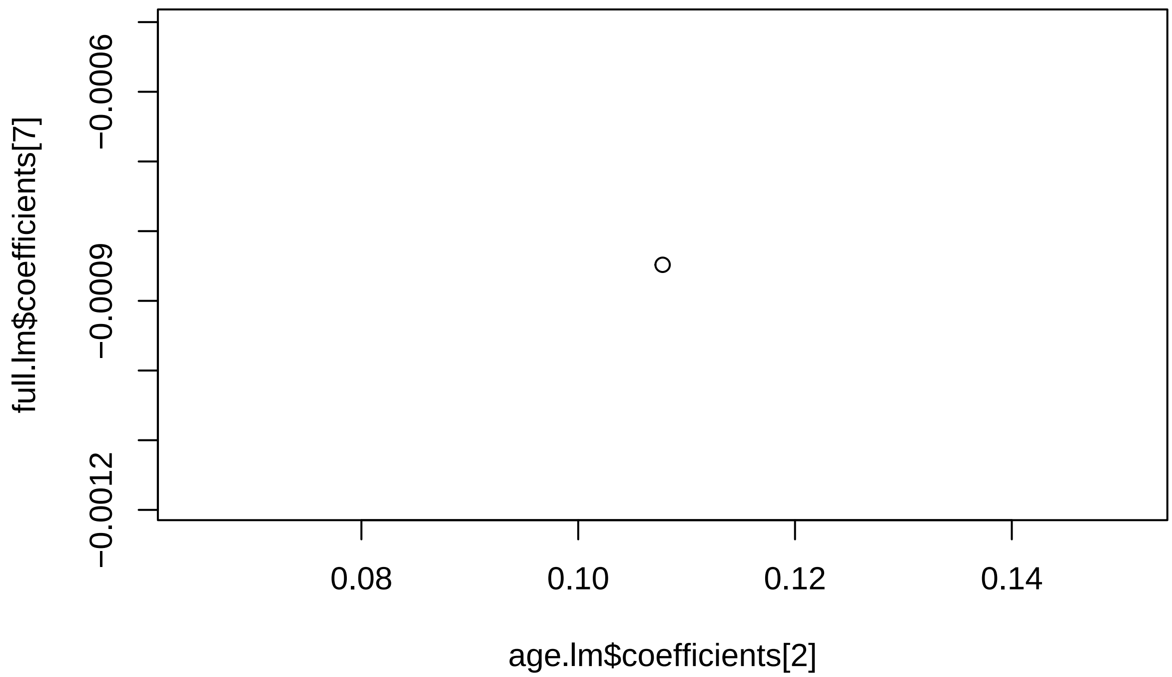

```
plot(nox.lm$coefficients[2],full.lm$coefficients[5])
```



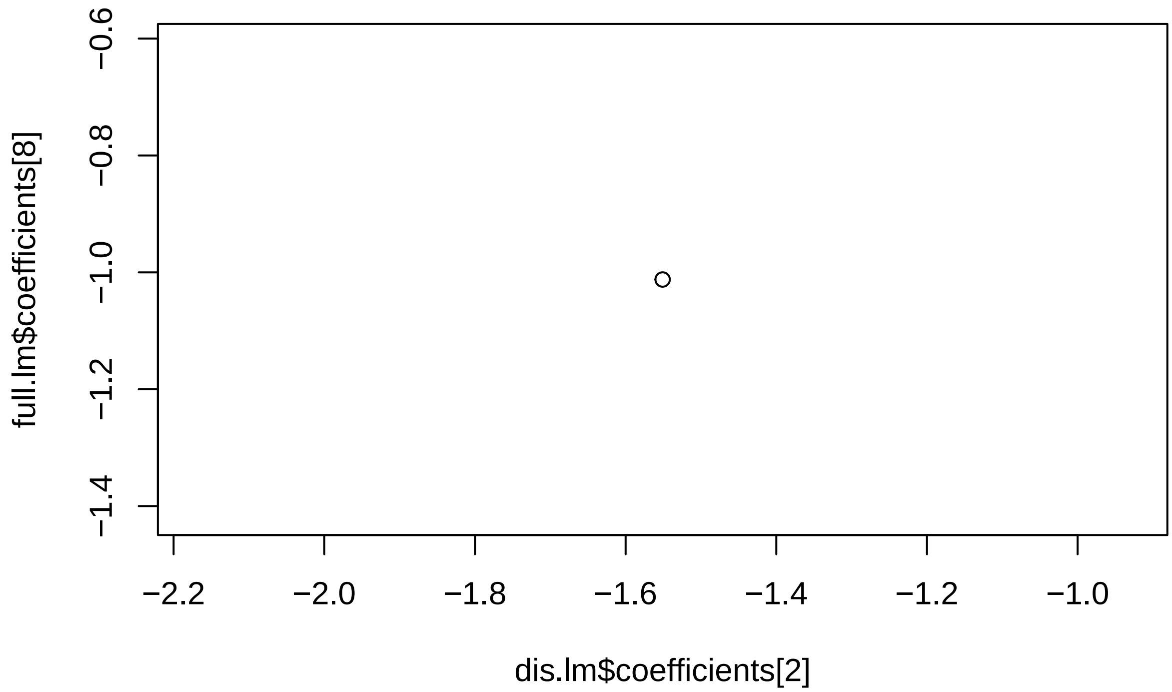

```
plot(rm.lm$coefficients[2],full.lm$coefficients[6])
```



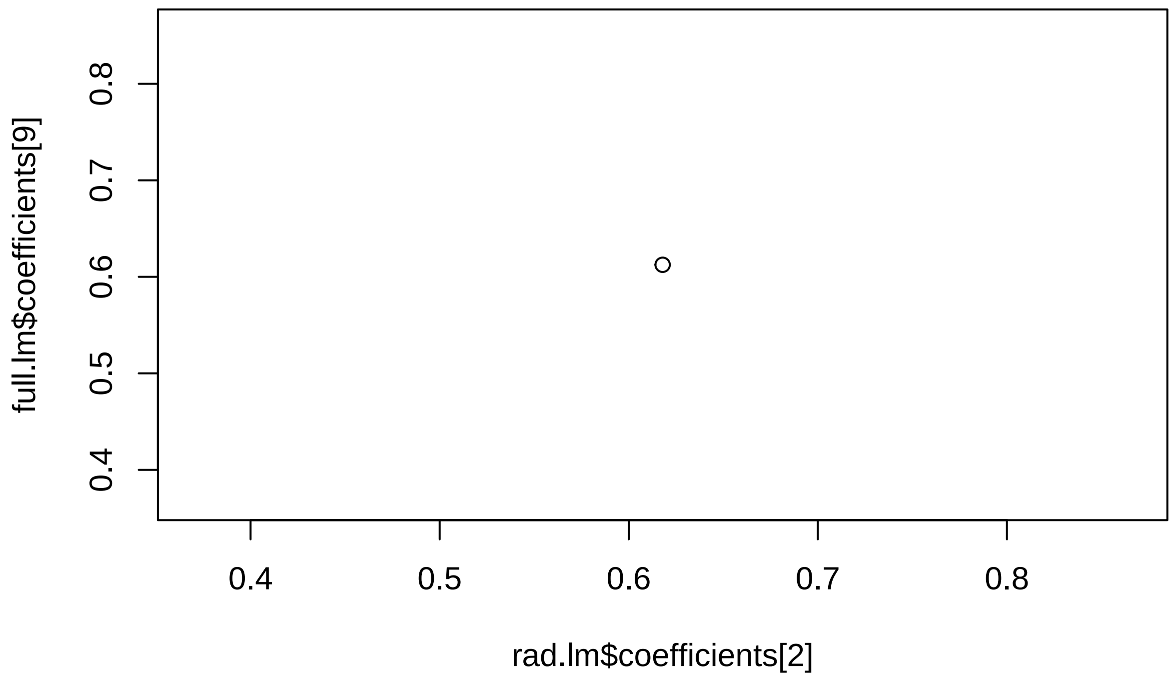

```
plot(age.lm$coefficients[2],full.lm$coefficients[7])
```



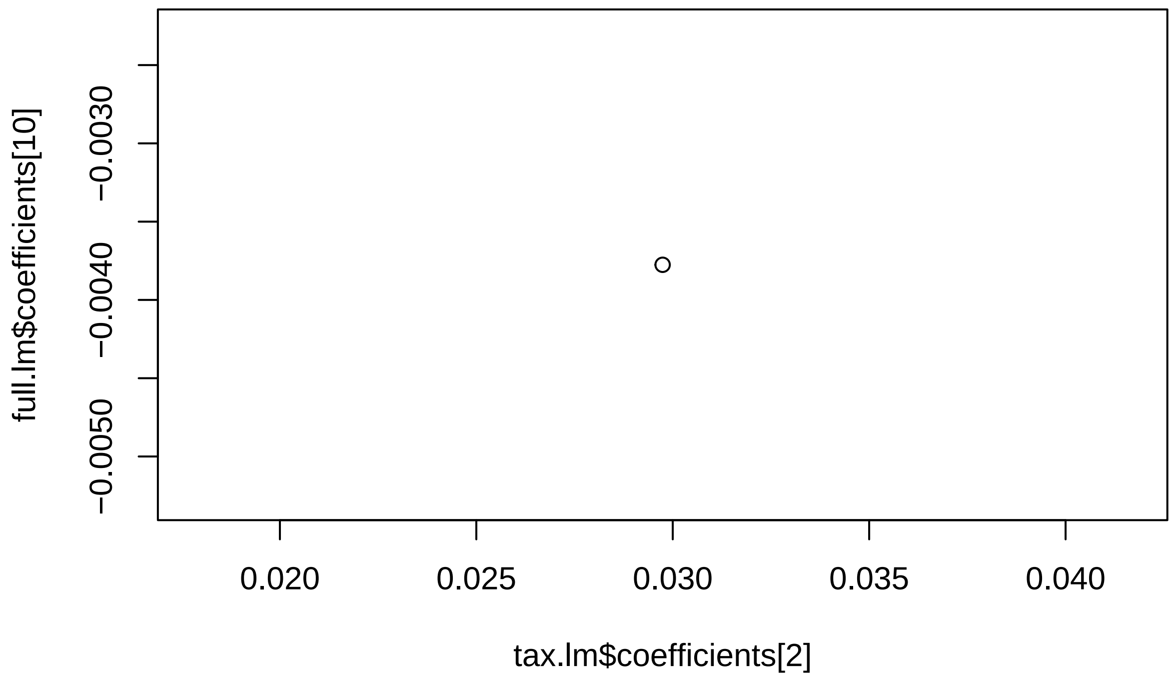

```
plot(dis.lm$coefficients[2],full.lm$coefficients[8])
```



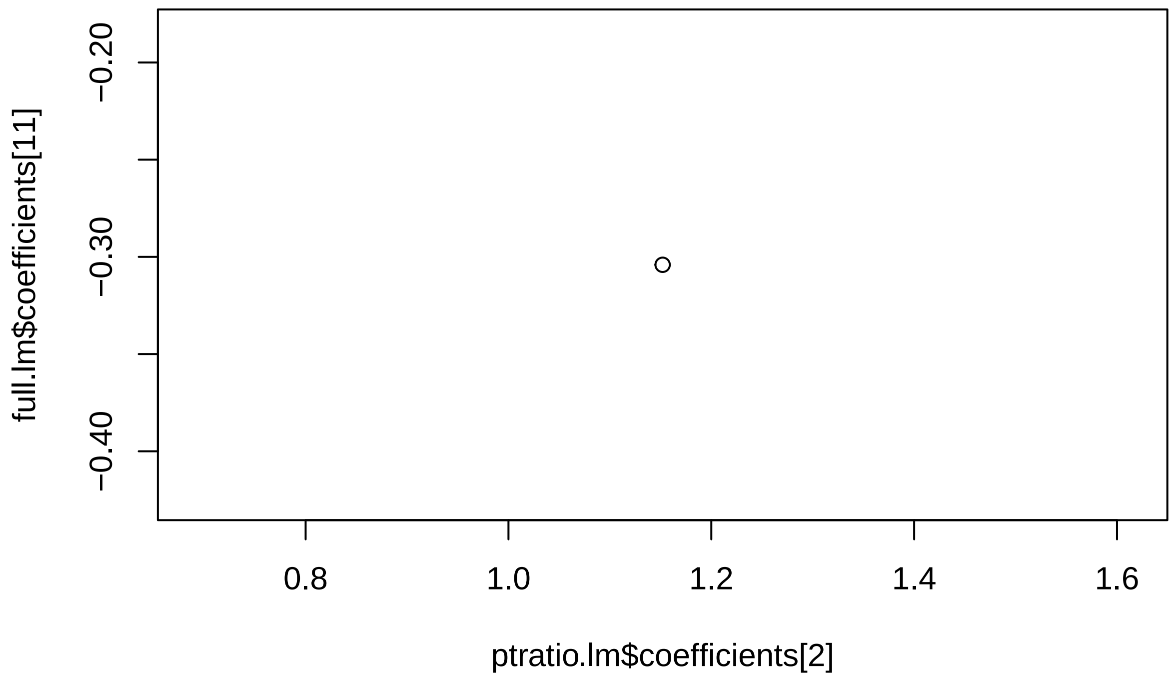

```r
plot(rad.lm$coefficients[2],full.lm$coefficients[9])
```



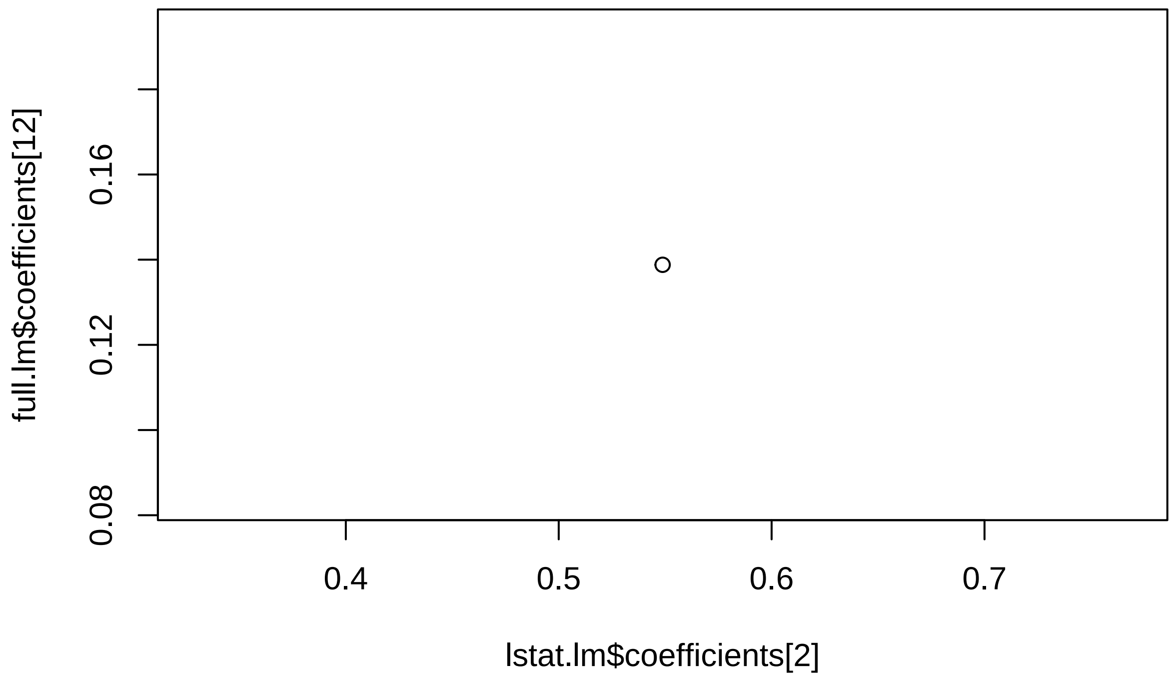

```
plot(tax.lm$coefficients[2],full.lm$coefficients[10])
```



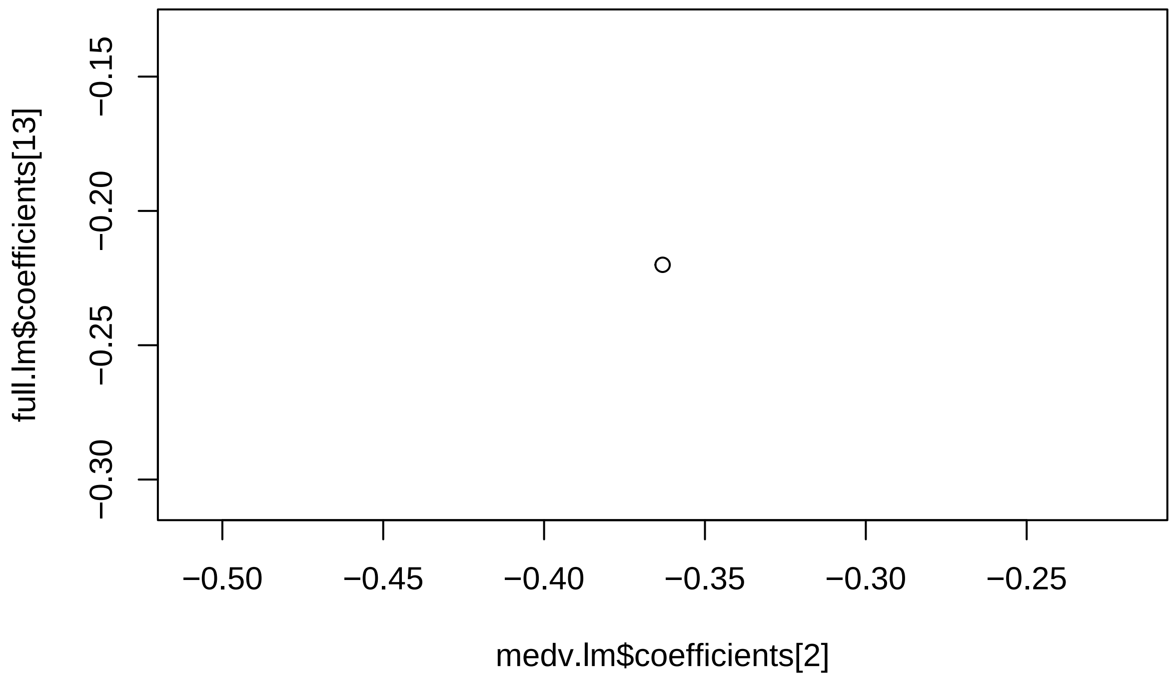

```
plot(ptratio.lm$coefficients[2],full.lm$coefficients[11])
```



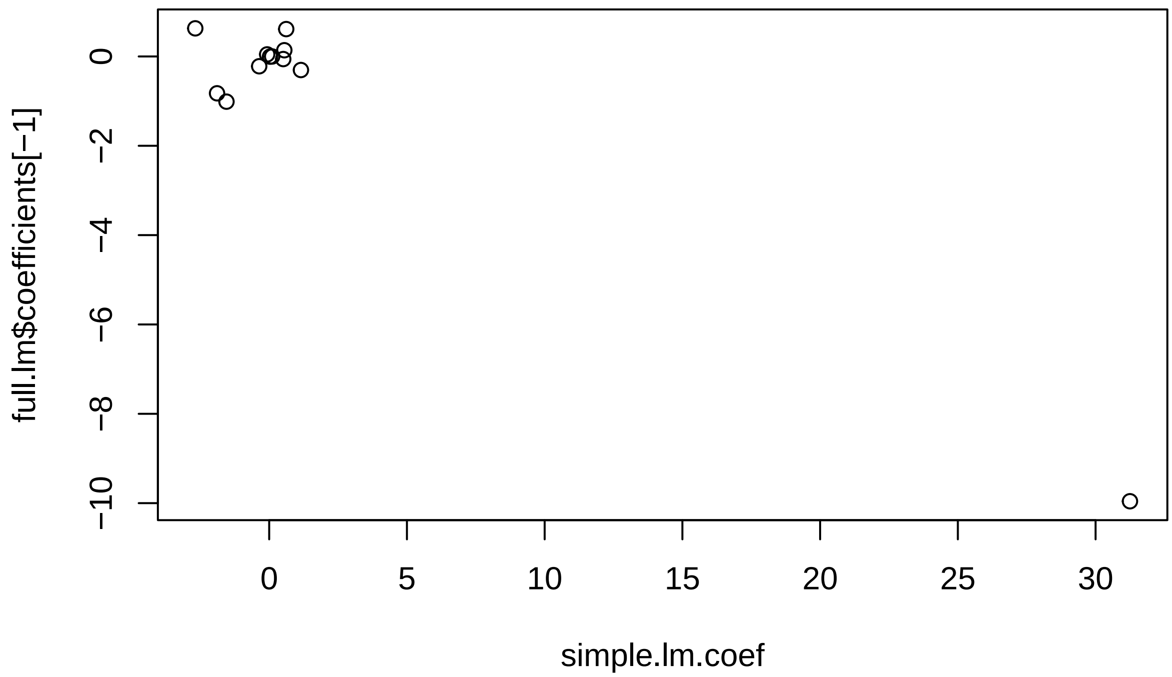

```r
plot(lstat.lm$coefficients[2],full.lm$coefficients[12])
```




```
plot(medv.lm$coefficients[2],full.lm$coefficients[13])
```




```
simple.lm.coef <- c(zn.lm$coefficients[2],indus.lm$coefficients[2],chas.lm$coefficients[2],nox.lm$coeffi
plot(simple.lm.coef,full.lm$coefficients[-1])
```




```
#chas only factor with 2 levels. Fitting nonlinear models against crim
zn.nlm <- lm(crim ~ poly(zn,3),data = Boston)
indus.nlm <- lm(crim ~ poly(indus,3),data = Boston)
nox.nlm <- lm(crim ~ poly(nox,3),data = Boston)
rm.nlm <- lm(crim ~ poly(rm,3),data = Boston)
age.nlm <- lm(crim ~ poly(age,3),data = Boston)
dis.nlm <- lm(crim ~ poly(dis,3),data = Boston)
rad.nlm <- lm(crim ~ poly(rad,3),data = Boston)
tax.nlm <- lm(crim ~ poly(tax,3),data = Boston)
ptratio.nlm <- lm(crim ~ poly(ptratio,3),data = Boston)
lstat.nlm <- lm(crim ~ poly(lstat,3),data = Boston)
medv.nlm <- lm(crim ~ poly(medv,3),data = Boston)

summary(zn.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(zn, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.821 -4.614 -1.294  0.473 84.130
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.6135     0.3722   9.709  < 2e-16 ***
```

```
## poly(zn, 3)1 -38.7498     8.3722  -4.628  4.7e-06 ***
## poly(zn, 3)2  23.9398     8.3722   2.859  0.00442 **
## poly(zn, 3)3 -10.0719     8.3722  -1.203  0.22954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.372 on 502 degrees of freedom
## Multiple R-squared:  0.05824,    Adjusted R-squared:  0.05261
## F-statistic: 10.35 on 3 and 502 DF,  p-value: 1.281e-06
```

```r
summary(indus.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(indus, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.278 -2.514  0.054  0.764 79.713
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.614      0.330  10.950  < 2e-16 ***
## poly(indus, 3)1   78.591      7.423  10.587  < 2e-16 ***
## poly(indus, 3)2  -24.395      7.423  -3.286  0.00109 **
## poly(indus, 3)3  -54.130      7.423  -7.292  1.2e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.423 on 502 degrees of freedom
## Multiple R-squared:  0.2597, Adjusted R-squared:  0.2552
## F-statistic: 58.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(nox.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(nox, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.110 -2.068 -0.255  0.739 78.302
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.3216  11.237  < 2e-16 ***
## poly(nox, 3)1  81.3720     7.2336  11.249  < 2e-16 ***
## poly(nox, 3)2 -28.8286     7.2336  -3.985 7.74e-05 ***
## poly(nox, 3)3 -60.3619     7.2336  -8.345 6.96e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.234 on 502 degrees of freedom
```

```
## Multiple R-squared:  0.297,  Adjusted R-squared:  0.2928
## F-statistic: 70.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(rm.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(rm, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.485  -3.468  -2.221  -0.015  87.219
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.6135     0.3703   9.758  < 2e-16 ***
## poly(rm, 3)1 -42.3794     8.3297  -5.088 5.13e-07 ***
## poly(rm, 3)2  26.5768     8.3297   3.191  0.00151 **
## poly(rm, 3)3  -5.5103     8.3297  -0.662  0.50858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.33 on 502 degrees of freedom
## Multiple R-squared:  0.06779,    Adjusted R-squared:  0.06222
## F-statistic: 12.17 on 3 and 502 DF,  p-value: 1.067e-07
```

```r
summary(age.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(age, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.762 -2.673 -0.516  0.019 82.842
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.6135     0.3485  10.368  < 2e-16 ***
## poly(age, 3)1  68.1820     7.8397   8.697  < 2e-16 ***
## poly(age, 3)2  37.4845     7.8397   4.781 2.29e-06 ***
## poly(age, 3)3  21.3532     7.8397   2.724  0.00668 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.84 on 502 degrees of freedom
## Multiple R-squared:  0.1742, Adjusted R-squared:  0.1693
## F-statistic: 35.31 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(dis.nlm)
```

```
##
```

```
## Call:
## lm(formula = crim ~ poly(dis, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.757  -2.588   0.031   1.267  76.378
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.6135     0.3259  11.087  < 2e-16 ***
## poly(dis, 3)1 -73.3886     7.3315 -10.010  < 2e-16 ***
## poly(dis, 3)2  56.3730     7.3315   7.689 7.87e-14 ***
## poly(dis, 3)3 -42.6219     7.3315  -5.814 1.09e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.331 on 502 degrees of freedom
## Multiple R-squared:  0.2778, Adjusted R-squared:  0.2735
## F-statistic: 64.37 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(rad.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(rad, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.381  -0.412  -0.269   0.179  76.217
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.6135     0.2971  12.164  < 2e-16 ***
## poly(rad, 3)1 120.9074     6.6824  18.093  < 2e-16 ***
## poly(rad, 3)2  17.4923     6.6824   2.618  0.00912 **
## poly(rad, 3)3   4.6985     6.6824   0.703  0.48231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.682 on 502 degrees of freedom
## Multiple R-squared:    0.4,  Adjusted R-squared:  0.3965
## F-statistic: 111.6 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(tax.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(tax, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.273  -1.389   0.046   0.536  76.950
##
```

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.6135     0.3047  11.860  < 2e-16 ***
## poly(tax, 3)1  112.6458     6.8537  16.436  < 2e-16 ***
## poly(tax, 3)2   32.0873     6.8537   4.682 3.67e-06 ***
## poly(tax, 3)3   -7.9968     6.8537  -1.167    0.244
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.854 on 502 degrees of freedom
## Multiple R-squared:  0.3689, Adjusted R-squared:  0.3651
## F-statistic:  97.8 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(ptratio.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(ptratio, 3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.833 -4.146 -1.655  1.408 82.697
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.614      0.361  10.008  < 2e-16 ***
## poly(ptratio, 3)1    56.045      8.122   6.901 1.57e-11 ***
## poly(ptratio, 3)2    24.775      8.122   3.050  0.00241 **
## poly(ptratio, 3)3   -22.280      8.122  -2.743  0.00630 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.122 on 502 degrees of freedom
## Multiple R-squared:  0.1138, Adjusted R-squared:  0.1085
## F-statistic: 21.48 on 3 and 502 DF,  p-value: 4.171e-13
```

```r
summary(lstat.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(lstat, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.234  -2.151  -0.486   0.066  83.353
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.6135     0.3392  10.654   <2e-16 ***
## poly(lstat, 3)1   88.0697     7.6294  11.543   <2e-16 ***
## poly(lstat, 3)2   15.8882     7.6294   2.082   0.0378 *
## poly(lstat, 3)3  -11.5740     7.6294  -1.517   0.1299
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.629 on 502 degrees of freedom
## Multiple R-squared:  0.2179, Adjusted R-squared:  0.2133
## F-statistic: 46.63 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(medv.nlm)
```

```
##
## Call:
## lm(formula = crim ~ poly(medv, 3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.427  -1.976  -0.437   0.439  73.655
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.614      0.292  12.374  < 2e-16 ***
## poly(medv, 3)1  -75.058      6.569 -11.426  < 2e-16 ***
## poly(medv, 3)2   88.086      6.569  13.409  < 2e-16 ***
## poly(medv, 3)3  -48.033      6.569  -7.312 1.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.569 on 502 degrees of freedom
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.4167
## F-statistic: 121.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

a) For chas (which is a binary variable), there is no linear relationship. However, for the remaining continuous variables there appears to be some weak linear relationships (as supported by the correlation column as well). The rad variable has the strongest linear relationship to crim and from the p-values chas is the only variable that we cannot reject the null hypothesis of beta_1 = 0.
b) There is very little increase in variability accounted for by the full model compared to the rad simple linear regression. For the zn, dis, rad, and medv variables we can reject the null hypothesis of beta_j = 0.
c) The simple linear model for crim and rad is very close to the variability explained by the full model.
d) There seems to be some nonlinear relationship between rad, tax, and medv and crim, but the remaining models do not seem to have a strong association to crim.

HW2 KNN Regression

2025-02-13

```r
library(MASS)
set.seed(123)

# Subset the data to only the variables of interest: lstat and medv
data(Boston)
boston_sub <- Boston[, c("lstat", "medv")]

# Create a train/test split (70% training, 30% testing)
n <- nrow(boston_sub)
train_index <- sample(1:n, size = round(0.7 * n))
train_data <- boston_sub[train_index, ]
test_data  <- boston_sub[-train_index, ]

# --- Least Squares Fit ---
lm_fit <- lm(medv ~ lstat, data = train_data)
lm_pred <- predict(lm_fit, newdata = test_data)
lm_mse <- mean((test_data$medv - lm_pred)^2)
cat("Least Squares MSE:", lm_mse, "\n")
```

```
## Least Squares MSE: 41.08968
```

```r
# --- KNN Regression ---

knn_reg <- function(x_train, y_train, x_test, k) {
  predictions <- sapply(x_test, function(x) {
    # Compute Euclidean distance
    distances <- abs(x_train - x)
    # Find indices of the k nearest neighbors
    neighbor_indices <- order(distances)[1:k]
    # Return the average response of the neighbors
    mean(y_train[neighbor_indices])
  })
  return(predictions)
}

k <- 5
knn_pred <- knn_reg(train_data$lstat, train_data$medv, test_data$lstat, k)
knn_mse <- mean((test_data$medv - knn_pred)^2)
cat("KNN (k =", k, ") MSE:", knn_mse, "\n")
```

```
## KNN (k = 5 ) MSE: 36.91268
```

```r
# --- Compare the Two Methods ---
if (knn_mse < lm_mse) {
  cat("KNN regression performs better on the test set (lower MSE).\n")
} else if (lm_mse < knn_mse) {
```

```
  cat("Least Squares regression performs better on the test set (lower MSE).\n")
} else {
  cat("Both methods perform equally well on the test set.\n")
}
```

```
## KNN regression performs better on the test set (lower MSE).
```

```
plot(test_data$lstat, test_data$medv, pch = 16, col = "gray",
     xlab = "lstat", ylab = "medv", main = "Model Predictions vs Actual")
points(test_data$lstat, lm_pred, col = "blue", pch = 17)
points(test_data$lstat, knn_pred, col = "red", pch = 18)
legend("topright", legend = c("Actual", "Least Squares", "KNN"),
       col = c("gray", "blue", "red"), pch = c(16, 17, 18))
```