Brain3.r: Explanation of code

Goals of code:
Focus on diagnostics for linear regression models using the brain dataset in R. The first few lines read the data and compute the log transform of each variable.

Regression diagnostics:
These are provided by still more helper functions. See `?influence.measures` for a full list. All accept the result from `lm()` and return a vector with the requested measure. The result has one value for each observation. The functions we need and a few more that I'll mention are:

- `rstandard()`: Standardized residuals. Residual divided by its standard error
- `rstudent()`: externally standardized residuals. Residual divided by its standard error, where s is computed without the $i'th$ observation.
- `cooks.distance()`: Cook's Distance

The usual way of plotting quantities like Cook's Distance is to use the observation number as the X variable. The line `nobs <- nrow(brain)` is one of various ways to extract the number of observations in the data set and save it as `nobs`. `nrow()` returns the number of rows in a data frame.

You could also use `dim(brain)[1]`. `dim()` returns a vector of two numbers (the "DIMensions") of the data set, i.e. the number of rows and number of columns. `[1]` extracts the first, the number of rows.

You could also use `length(d)`, which returns the length of a vector, i.e. the number of values in the vector. This is the number of observations.

Note: `length(brain)` does not work as you might expect. If brain is a data frame, this gives you the number of variables. If brain is a matrix, it gives you the total number of values (rows times columns).

Once you have the total number of observations, plotting values against the observation number is easy. Just use `1:nobs` as the X value, because `1:nobs` is the sequence: 1, 2, 3, $\cdots$ nobs. Or, store a column with those values in the data set: `brain$rownumber <- 1:nobs`.

The easiest way to get VIF, variance inflation factor, values for each variable is to use the `vif()` function in the car library. Note: there is a VIF library, but that does something different.

Ways for diagnostics:

- Extracting Predicted Values and Residuals: `predict() / residuals()`

- Requesting Standardized Residuals: `rstandard()`
- Calculating Cook's D: `cooks.distance()`
- Calculating Variance Inflation Factor (VIF): `vif()`

These diagnostics are crucial in validating the assumptions behind linear regression. They help ensure the reliability and validity of the model's predictions.

By identifying and addressing issues like outliers, influential points, and multicollinearity, you can improve the accuracy and interpretability of your model.

Identifying interesting observations:

- Outliers: data points that significantly differ from the majority of the data.

  Identification Methods:

  Look at the standardized residuals

- Influential points: observations that, if removed, would significantly change the regression model's results.

  Identification Methods:

  Cook's Distance: Measures the influence of each observation. Points with Cook's distance greater than a certain threshold (like 4/n or 1) are considered highly influential.

- Residual Analysis: to check the assumption of homoscedasticity (equal variances) and to identify outliers or unusual patterns.

  Methods:

  Standardized Residuals: Observations with usually large or unusually small standardized residuals should be investigated. If the model fits, 95% of the standardized residuals should be between -2 and 2 and very very few should be less than -4 or larger than 4.

- Multicollinearity Assessment: to identify predictor variables that are highly correlated with each other.

  Methods:

  Variance Inflation Factor (VIF): A VIF greater than 5-10 suggests significant multicollinearity.