Week 4 Software Lesson: Multiple Linear Regression, Part 3 in R

Goals

The goal of this software lesson is to learn how to

- obtain model diagnostics to check assumptions and evaluate the model fit.

Required Packages

You will need to install the following packages in R for this software lesson:

- `jtools`
- `car`

DATASET

Let's use the Rheumatoid Arthritis dataset again.

Recall that the dataset, data dictionary, and dataset introduction can be found on Canvas, titled *RheumArth.csv*, *RheumArth Data Dictionary.pdf*, and *RheumArth Data Introduction.pdf*, respectively.

```
rheum_arth <- read.csv(file = file.choose(), header = TRUE)
```

MODEL DIAGNOSTICS

In this software lesson, you are going to obtain model diagnostics for an example used in a previous software lesson.

That is, in the Rheumatoid Arthritis study, you examined the relationship between the a disease activity measurement, `CDAI` (Clinical Disease Activity Indicator) value, and age groups (`AgeGp`), after adjusting for `Sex` and years since diagnosis (`Yrs_From_Dx`). The code for fitting this model is presented again (for reference).

```
model1 <- lm(CDAI ~ factor(AgeGp) + Sex + Yrs_From_Dx, data = rheum_arth)
```

Recall that you learned how to check model assumptions using plots in the previous course (PubH 6450: Biostatistics 1, see Week 14: Software Lessons). There is an additional plot, a histogram, that can be used to assess the normality of the residuals.

- To create a histogram of the residuals, use the `hist()` function on the saved `lm` object (`model1`) and the residuals information that is contained in that object (`$residuals`):

```
hist(model1$residuals)
```

R TIP

Objects in R can contain many things, such as variables, lists, or data frames. Some you can view directly when you examine the object and some are hidden within the object. You can examine what contents are contained in the object by typing `names(object.name)`.

Now you are going to obtain additional model diagnostics to evaluate the model fit.

Calculate measures of model fit

To measure model fit, we can use adjusted R^2 or PRESS.

Adjusted R^2

Adjusted R^2 provides information about model fit, with the higher the value, the more the model is able to explain the variability in the outcome (generally speaking).

Recall that you can obtain the adjusted R^2 value for the model using the `summ()` function from the `{jtools}` package.

```
summ(model1, confint= TRUE, ci.width = 0.95, digits = 4)
```

PRESS

The PRESS (Predicted REsidual Sum-of-Square) statistic tests to see how well the model will predict on future data. The smaller the PRESS value, the better the fit.

- To calculate the PRESS statistic,
  - first, create the `PRESS()` function by running the `PRESS <- function(model){...}` code (Note: There are not many built-in PRESS functions in R, and so you are doing it "by-hand" rather than use a built-in function).
  - Then, use the `PRESS()` function on the model:

```
## Creating the PRESS function
PRESS <- function(model) {
    i <- residuals(model)/(1 - lm.influence(model)$hat)
    sum(i*i)
}

## Using the PRESS() function on the model
PRESS(model1)
```

The output is the value of the PRESS statistic for the model. (Note: This value by itself isn't really meaningful. This statistic is best used when you are comparing models to see which model fits the data better.)

Check for outliers

To check for outliers, you can use various kinds of residual plots. Previously, you created a residual plot for evaluating model assumptions, but you can also examine standardized residual and studentized residual (i.e., "deleted residuals") plots to determine observations that seem *much* different than other observations.

Standardized residual plot

Standardized residuals are residuals that have been standardized by dividing by the estimated standard deviation of each residual, which takes into account the leverage/influence of an observation. An observation is worth investigating if the standardized residual value is larger than 3 (meaning it is more than 3 standard deviations away from the model-fitted value).

- To create a standardized residual plot, use the `plot()` function on the saved `lm` object (`model1`), specifying the third plot (`which = 3`, for the standardized residual plot) and how many points you want to identify by observation number starting with most extreme (`id.n =`), then use the `abline()` function to add a horizontal line at $+\sqrt{3}$ (`h = sqrt(3)`) for a visual indicator on observations worth investigating:

```
## Plotting the standardized residuals
plot(model1, which = 3, id.n = 10)

## Adding the threshold line
abline(h = sqrt(3), lty = 2)
```

R TIP

The `lty=` argument stands for "line type". See this resource for the different line types available in R.

Note: A line is added at $\sqrt{3}$ because R plots the square root of the absolute value of the standardized residuals.

You can use this plot to investigate those observations that seem *much* different than the other observations, based on the model. For example, you can see observation numbers 254 and 255 next to the points that have "extreme" standardized residual values.

Studentized residual plot

Studentized residuals are similar to standardized ones, except its calculated from a model that does not include the observation in question. An observation is worth investigating if the studentized residual value is larger than $+/-2$.

- To create a studentized residual plot, use the `plot()` function, specifying the `rstudent`() function on the saved `lm` object (`model1`) and labeling the y-axis and title, then use the `abline()` function, twice, to add horizontal lines at -2 and $+2$ (`h = -2`, `h = 2`) for a visual indicator on observations worth investigating:

```
## Plotting the studentized residuals
plot(rstudent(model1),ylab="Studentized Residuals",main="Studentized Residuals")

## Adding the threshold lines
abline(h = -2, lty = 2)
abline(h = 2, lty = 2)
```

Similar to the standardized residual plot, you can use this plot to investigate those observations that seem *much* different than the other observations, based on the model. For example, you can see that there are several observations beyond $+2$.

- (optional) To obtain observation numbers for those "extreme" observations,
  - first, save and use the `which()` function on the absolute value (`abs()`) of the studentized residuals of the model (`rstudent(model1)`), specifying that you only want information for values greater than 2 (and piping to some other functions to be able to use the observation numbers in the next step).
  - Then, use the dataset name and then brackets, `[<row>, <column>]`, specifying the observation numbers in the "row" part, before the comma:

```
extreme.obs <- which(abs(rstudent(model1)) > 2) |>
  names() |> #observation number in original dataset
  as.numeric() #converts to a number

## Investigating observations with "extreme" values
rheum_arth[extreme.obs, ]
```

R TIP

You could carry out a similar task for investigating the standardized residuals, but using the `rstandard()` function.

The output displays the observations that were flagged as potential outliers, according to the studentized residual values.

Check for influential observations

The statistics to detect highly influential points are Cook's distance, DFFITs, and DFBETAs.

Cook's distance

Cook's distance (D) measures how much the model changes when an observation is removed from the model. An observation is worth investigating if the Cook's D value is larger than 1.

- To create a plot of Cook's D values verses observation number, use the `plot()` function on the saved `lm` object (`model1`), specifying the third plot (`which = 4`, for the Cook's distance plot), then use the `abline()` function to add a horizontal line at 1 (`h = 1`) for a visual indicator on observations worth investigating:

```
## Plotting the Cook's D
plot(model1, which=4)

## Adding the threshold line
abline(h = 1, lty = 2)
```

Similar to the other plots in this software lesson, you can use this plot to investigate those observations that have "large" influential values, based on the model. For example, you can see that there are no observations in this dataset that have "large" Cook's D values.

DFFITs

The DFFITs (DiFference of FITs) statistic measures how much the model changes when an observation is removed. An observation is worth investigating if the DFFITs value is larger than 1 for small datasets, or, larger than $2\sqrt{(\frac{p}{n})}$ for large datasets, where p = the number of predictors and n is the number of non-missing observations used in the model.

- To create a plot of DFFITs values verses observation number,
  - first, use the `plot()` function, specifying the `dffits()` function on the saved `lm` object (`model1`).
  - Then calculate the threshold value for what constitutes as "extreme" (i.e., values greater than $2\sqrt{(\frac{p}{n})}$ where p = the number of predictors (in other words, the number of coefficients - 1), and use the `abline()` function, twice, to add horizontal lines at $-2\sqrt{(\frac{p}{n})}$ and $+2\sqrt{(\frac{p}{n})}$ (`h = -thresh.dff`, `h = thresh.dff`) for a visual indicator on observations worth investigating:

```
## Plotting the DFFITs
plot(dffits(model1), ylab = "DFFITs")

## Calculating threshold for DFFITS
thresh.dff <- 2*sqrt((length(model1$coefficients)-1) / length(model1$fitted.values))

## Adding the threshold lines
abline(h = thresh.dff, lty = 2)
abline(h = -thresh.dff, lty = 2)
```

Note: The number of regression coefficients in the R model also includes the intercept, which is why 1 is subtracted off of the `length(model$coefficients)`. Also, because there are missing values in the dataset, the sample size is found by how many fitted values (one for each non-missing observation) are calculated from the R model.

Similar to the other plots in this software lesson, you can use this plot to investigate those observations that have "large" influential values, based on the model. For example, you can see that there are about a dozen observations in this dataset that have "large" DFFITs values.

DFBETAs

The DFBETAs (DiFference of BETAs) statistic measures how much a regression coefficient (betas, that is, intercept and predictor(s)) changes when an observation is removed. An observation is worth investigating if the DFBETAs value is larger than $\frac{2}{\sqrt{n}}$.

- To create plots of DFBETAs values verses observation number,
  - first, save and calculate the DFBETAs values for the model, using the `dfbetas()` function on the saved `lm` object (`model1`).
  - Then calculate the threshold value for what constitutes as "extreme" (i.e., values greater than $\frac{2}{\sqrt{n}}$).
  - Then figure out the names of the columns in the saved `DFBETAS` object (`colnames(DFBETAS)`).
  - Use `par(mfrow = c(rows, columns)` function to modify the plot settings so more than one plot can be displayed at a time. (Note: Be sure to set the plot settings back to default after you are done creating multiple plots.)
  - Finally, for every regression coefficient, use the `plot()` function on the saved `DFBETAS` object, specifying the regression coefficient name each time, and use the `abline()` function, twice, to add horizontal lines at $-\frac{2}{\sqrt{n}}$ and $+\frac{2}{\sqrt{n}}$ (`h = -thresh.dfb`, `h = thresh.dfb`) for a visual indicator on observations worth investigating:

```
## Computing the DFBETAs
DFBETAS <- dfbetas(model1)

## Calculating threshold for DFBETAs
thresh.dfb <- 2 / sqrt(length(model1$fitted.values))

## Be sure to check the spelling of the predictors and enter them accordingly in each plot() call below
colnames(DFBETAS)

## Modifying the plot settings, to create 2 plots: 2 rows and 2 columns
## (if you have fewer coefficients, you can modify the rows and columns)
par(mfrow=c(2, 2))

## Plotting the data for the Intercept and adding threshold lines
plot(DFBETAS[, "(Intercept)"], ylab="Intercept")
abline(h = thresh.dfb, lty = 2)
abline(h = -thresh.dfb, lty = 2)

## Plotting the data for the AgeGp and adding threshold lines
plot(DFBETAS[, "factor(AgeGp)2"], ylab="AgeGp")
abline(h = thresh.dfb, lty = 2)
abline(h = -thresh.dfb, lty = 2)

## Plotting the data for Sex and adding threshold lines
plot(DFBETAS[, "factor(Sex)1"], ylab="Sex")
abline(h = thresh.dfb, lty = 2)
abline(h = -thresh.dfb, lty = 2)
```

```
## Plotting the data for Yrs_From_Dx and adding threshold lines
plot(DFBETAS[, "Yrs_From_Dx"], ylab="Yrs_From_Dx")
abline(h = thresh.dfb, lty = 2)
abline(h = -thresh.dfb, lty = 2)

## Setting the plot settings to the default 1 plot
par(mfrow=c(1, 1))
```

Similar to the other plots in this software lesson, you can use this plot to investigate those observations that have "large" influential values, based on the model. For example, you can see that there are several observations for each regression coefficient that have "large" DFBETAs values.

Collinearity and multicollinearity

Another consideration when evaluating model fit is whether there are issues of collinearity or multicollinearity.

Collinearity

Collinearity happens when one predictor is highly correlated with another predictor. When there is collinearity, it can make the model unstable and give odd results. To investigate collinearity between predictors, calculate a correlation matrix.

- To create a correlation matrix between predictors, use the `cor()` function on the dataset and then brackets, `[<row>, <column>]`, specifying the variable names in the "column" part, after the comma, and using only non-missing observations (`use = "pairwise.complete.obs"`):

```
cor(rheum_arth[ , c("AgeGp", "Sex", "Yrs_From_Dx")], use = "pairwise.complete.obs")
```

R TIP

Note that the `cor()` function only works on variables in `int` or `num` form. In addition, the function will return `NA` if there is any missing data, so that is why it was necessary to add the argument `use = "pairwise.complete.obs"`.

The output displays the correlations between the possible pairs of variables. The correlations along the diagonal of the table are all equal to 1 because each variable is perfectly correlated with itself.You only need to look at the upper triangle OR lower triangle, for the information, since they both present the same values.

Multicollinearity

Multicollinearity happens when one predictor is highly correlated with some combination of two or more other predictors. When there is multicollinearity, it can make the coefficient (beta) estimates less precise, leading to less precise inference. This is called "variance inflation".

Variance inflation is measured by the Variance Inflation Factor (or VIF.) The variance inflation factor ranges from 1 to infinity. A VIF below 4 indicates the variable is minimally correlated with any other predictors, a VIF between 4 and 10 indicates moderate correlation, and a VIF above 10 indicates high enough correlation to affect the precision of the coefficient estimates and cast doubt on the effectiveness of the inferential results.

- To compute the VIF for each predictor, use the `vif()` function from the `{car}` package on the saved `lm` object (`model1`):

```
vif(model1)
```

The output displays the VIF values for each predictor in the model.