Linear Regression

Arnab Maity - Modified by Justin Post

Packages used in this set of notes:

library(MASS)  
library(klaR)  
library(tufte)  
library(tidyverse)  
library(lubridate)  
library(caret)  
library(rsample)  
library(ISLR2)  
library(knitr)  
library(AppliedPredictiveModeling)  
library(kableExtra)  
library(robustbase)

# Big Picture

For now, we stick to the regression task *where we have a quantitative response*. We need to select a model form to work with. We’ll start with a basic parametric model - *a model with a stronger structural form* - called the (Multiple) Linear Regression model.

The model is much simpler compared to other modern techniques; however, such models are still very useful in developing new methods. In fact, many flexible nonparametric models can be thought of generalizations of linear regression model. We could spend an entire course on this topic if we wanted to!

Recall that we have two major goals:

* Inference - *determine which predictors are important for the model and quantifying their effects and relationships*
* Prediction - *performing well at predicting responses for observations the data was not trained on*

We’ll see that most of the models we look at in this section can be used for either of these tasks!

# Introduction to the Linear Regression Model

We already investigated the simple linear regression model. This is a model where we have a single predictor being used to model the response.

Let’s reintroduce our data on bike sharing.

bike\_share <- read\_csv("https://www4.stat.ncsu.edu/online/datasets/SeoulBikeData.csv",  
 local = locale(encoding = "latin1"))

Rows: 8760 Columns: 14  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (4): Date, Seasons, Holiday, Functioning Day  
dbl (10): Rented Bike Count, Hour, Temperature(°C), Humidity(%), Wind speed ...  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

bike\_share |>   
 select(`Rented Bike Count`, everything()) |>  
 slice(1:4) |>  
 kable()

| Rented Bike Count | Date | Hour | Temperature(°C) | Humidity(%) | Wind speed (m/s) | Visibility (10m) | Dew point temperature(°C) | Solar Radiation (MJ/m2) | Rainfall(mm) | Snowfall (cm) | Seasons | Holiday | Functioning Day |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 254 | 01/12/2017 | 0 | -5.2 | 37 | 2.2 | 2000 | -17.6 | 0 | 0 | 0 | Winter | No Holiday | Yes |
| 204 | 01/12/2017 | 1 | -5.5 | 38 | 0.8 | 2000 | -17.6 | 0 | 0 | 0 | Winter | No Holiday | Yes |
| 173 | 01/12/2017 | 2 | -6.0 | 39 | 1.0 | 2000 | -17.7 | 0 | 0 | 0 | Winter | No Holiday | Yes |
| 107 | 01/12/2017 | 3 | -6.2 | 40 | 0.9 | 2000 | -17.6 | 0 | 0 | 0 | Winter | No Holiday | Yes |

As the variable names are non-standard in R, let’s quickly modify them to make our lives easier (and have better consistency). We’ll also make some variables factors, which are special character variables that only take on a few values (or levels).

bike\_share <- bike\_share |>  
 rename("date" = "Date",  
 "rented\_bike\_count" = `Rented Bike Count`,  
 "hour" = "Hour",  
 "temperature" = `Temperature(°C)`,  
 "humidity" = `Humidity(%)`,  
 "wind\_speed" = `Wind speed (m/s)`,  
 "visibility" = `Visibility (10m)`,  
 "dew\_point\_temperature" = `Dew point temperature(°C)`,  
 "solar\_radiation" = `Solar Radiation (MJ/m2)`,  
 "rainfall" = `Rainfall(mm)`,  
 "snowfall" = `Snowfall (cm)`,  
 "seasons" = "Seasons",  
 "holiday" = "Holiday",  
 "functioning\_day" = "Functioning Day"   
 ) |>  
 mutate(date = dmy(date), #convert the date variable from character  
 seasons = factor(seasons),  
 holiday = factor(holiday),  
 functioning\_day = factor(functioning\_day))

Ok, let’s graph the relationship the SLR model fits between rented\_bike\_count and temperature.

bike\_share |>  
 ggplot(aes(x = temperature, y = rented\_bike\_count)) +  
 geom\_point(size = 0.5) +  
 geom\_smooth(method = "lm")

|  |
| --- |
| Scatterplot with fitted SLR model overlayed |

The equation for the model is fit in the software and given here.

SLR\_fit <- lm(rented\_bike\_count ~ temperature, data = bike\_share)  
summary(SLR\_fit)$coefficients |>  
 kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | 329.9525 | 8.5410613 | 38.63132 | 0 |
| temperature | 29.0811 | 0.4861734 | 59.81631 | 0 |

Clearly, in real data, we often have relationships that are non-linear. However, sometimes these relationships can be *locally* linear. In other cases, even if the original relationship is not linear, one may transform the response and predictors (e.g., using a -transform) to get approximate linearity.

## Fitting the model

Once we’ve determined our model structure, we must **fit** or **estimate** our model. This is generally done by minimizing some criteria.

*The common criterion for the multiple linear regression model is least squares (equivalent to maximum likelihood with normal errors described shortly).*

## Conducting Inference Using the Model

If we want to do statistical tests, we usually need to make assumptions about the error terms of our linear regression model. The assumptions made to do inference are also dependent on how we fit the model.

*In the MLR setting, we usually assume our errors iid by a normal distribution with constant variance. These assumptions require us to investigate model diagnostics in order to understand the validity of the assumptions.*

## Using the Model to Predict

Once the model is fit, we often want to predict. This can be done by plugging in values of our predictor(s) we are interested in. The estimated values is called .

*In the SLR model we have*

Our plan will be to discuss the multiple linear regression model in detail and then look at extensions to the model such as the LASSO, Ridge Regression, Principle Component Regression, etc.

# Multiple Linear Regression (MLR)

A linear regression model has the form

where

* is a quantitative response *(rented\_bike\_count)*
* are predictor variables *(temperature, hour, seasons, etc.)*
* is unobserved random error
* is the *intercept* *(expected bike count when all predictors are 0)*
* Coefficients are ‘slope’ terms associated with predictor
  + Value of indicates the strength, and direction, of the *linear* relationship between and
  + is the expected change in the response for a unit change in , holding all other predictors constant
  + *If interactions or quadratics are included, this interpretation changes!*

MLR\_fit <- lm(rented\_bike\_count ~ temperature + hour + wind\_speed, data = bike\_share)  
summary(MLR\_fit)$coefficients |>  
 kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | -48.71548 | 13.2239366 | -3.683887 | 0.0002311 |
| temperature | 26.91385 | 0.4473949 | 60.156813 | 0.0000000 |
| hour | 31.30939 | 0.8048948 | 38.898736 | 0.0000000 |
| wind\_speed | 26.97563 | 5.3387085 | 5.052838 | 0.0000004 |

## Assumptions on the Errors

### Mean Zero

Typically, we assume that the errors have mean zero, that is . Thus we can write the mean response as

We call the model described above **a linear model** because is **linear in parameters**, that is, linear in .

We can have nonlinear terms of , but the model would be still a linear model!

* *For example, the so-called polynomial regression,*
* Fractional polynomial regression (introduced by Royston & Sauerbrei in their 2004 paper, *A new approach to modelling interactions between treatment and continuous covariates in clinical trials by using fractional polynomials.* in Stat Med)
* *where the powers* are chosen from
* Models with interaction such as

A non-linear model, in contrast, is a model such as

Hopefully, you can see that linear models do not just capture linear effects of , they can capture nonlinear functions of as well! The figurebelow shows a few examples functions that can be captured by appropriate linear models.

|  |
| --- |
| Examples of functions that can be captured by appropriate linear models. |

### Common Distributional Assumptions

A normality assumption on the errors, enable us to perform statistical inference on the coefficients such as:

* Constructing confidence intervals for : a set/range of values which contain the “true” value of with high probability (in repeated sampling). This can be investigated by a -statistic based confidence interval.
* Perform hypothesis tests to determine whether the -the predictor has any linear association with , vs. . This can be investigated using a -test.
* Perform hypothesis tests to determine whether the *any* of the predictors has any linear association with , vs.  at least on is non-zero. We can use -test to answer this question.

Although not the only way to make the above inferences, our common assumption on the distribution of the errors is

Conceptually, this idea can be visualized in the simple linear regression model by the graph below.

|  |
| --- |
| At each value of the predictor, we assume the values of the response variable are normally distributed about the line. Figure from physicsforums.com |

## Fitting a Linear Model

We denote the estimated coefficients as and the estimated response as .

How we determine these estimated coefficients depends! Let’s go through common methods for fitting the model.

### Least Squares

A common estimation procedure for the regression coefficients is the *least squares* technique.

* The difference between the observed and predicted values are called *residuals*,
* We define the *residual sum of squares*, also known as *sum-of-squared errors (SSE)* as

We see that MSE is simply RSS divided by the sample size! (This could be found over the training set or a test set.)

The *ordinary least squares (OLS)* procedure estimates by minimizing the sum-of-squares

with respect to ’s (resulting estimates are called the OLS estimates).

For the *simple linear regression* model the solutions are easy to derive with calculus.

* For the general regression model with predictors, writing closed form expression is easier in matrix from. We can convert the original regression model in matrix form as
* where
* is the column vector of responses
* is the column vector of regression coefficients
* is the column vector of errors.
* is matrix
  + We call the *model matrix* or *design matrix*
  + The first column of has all elements equal to (corresponding to the intercept)
  + For the remaining part of , each row corresponds to an unit/individual and each column corresponds to a predictor.
    - For the special case of simple linear regression with one predictor , the model matrix is of size ,
    - In general, we have

For our bike count data, we can write out some of these terms for clarity. Let’s just work with the first 10 observations for brevity.

bike\_share\_first\_ten <- bike\_share[1:10, ]  
y <- bike\_share\_first\_ten$rented\_bike\_count  
X <- bike\_share\_first\_ten |>  
 mutate(intercept = rep(1, 10)) |>  
 select(intercept, temperature, hour, wind\_speed) |>  
 as.matrix()  
y

[1] 254 204 173 107 78 100 181 460 930 490

X[1:5, ] |>  
 kable()

| intercept | temperature | hour | wind\_speed |
| --- | --- | --- | --- |
| 1 | -5.2 | 0 | 2.2 |
| 1 | -5.5 | 1 | 0.8 |
| 1 | -6.0 | 2 | 1.0 |
| 1 | -6.2 | 3 | 0.9 |
| 1 | -6.0 | 4 | 2.3 |

A *unique* minimizer of the sum-of-squares exists under certain conditions!

We can find this in software easily.

#%\*% is the matrix muliplication operator in R  
#solve() gives the matrix inverse  
#t() gives the transpose  
solution <-solve(t(X)%\*%X)%\*%t(X)%\*%y

| Parameter | Estimate |
| --- | --- |
| intercept | -1010.80288587219 |
| temperature | -208.213745545646 |
| hour | 6.92592094166932 |
| wind\_speed | -34.1911240997721 |

Which is the same as the basic lm() fit!

MLR\_first\_ten <- lm(rented\_bike\_count ~ temperature + hour + wind\_speed,  
 data = bike\_share\_first\_ten)  
MLR\_first\_ten$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | -1010.802886 |
| temperature | -208.213745 |
| hour | 6.925921 |
| wind\_speed | -34.191124 |

#### Notes on Matrix Requirements

The estimator above depends on the term , that is, the inverse of . Such an inverse exists only if has full column-rank. Equivalently, must have the following two properties:

(C1) The sample size is larger than the number of regression coefficients in the model, .

(C2) None of the columns of can be written as a weighted sum (called a *linear combination*) of the remaining columns.

If violates either of these conditions, then a *unique* least squares estimator does not exist. If violates (C2) but not (C1), then we can replace by a *generalized inverse*[[1]](#footnote-37), but there will be many estimators that minimize the sum-of-squares. Interpreting them will be difficult in general. In practice, even if the predictors are not perfectly correlated, their correlation can be high enough to cause numerical instability. This issue is known as *multicollinearity* among predictors. We can avoid this issue by removing the collinear predictors from the model.

If violates (C1) then (C2) is automatically violated as a matrix can not have full column rank if it has more rows than columns.

In that case, one can take a few steps described below.

* We can *remove highly correlated predictors* to reduce the overall number of predictors.
* Use *variance inflation factor (VIF)* – we will learn it shortly – to diagnose multicollinearity. VIF tells us how correlated each predictor is with the remaining predictors.
* Apply *dimension reduction techniques*, such as principal component analysis (PCA) or partial least squares (PLS).
* Apply *shrinkage methods*, such as LASSO regression, to reduce small regression coefficients to zero.

### Fitting Via Maximum Likelihood (Optional)

If you are familiar with maximum likelihood, the solutions obtained from the least squares optimization are the same as the solutions from the model that assumes our iid Normal errors with constant variance.

Consider the SLR model with Normal errors. The likelihood is given by

We want to find the values of , , and that maximize this function. We can interpret these values as the ‘most likely values of the parameters to have produced the data we saw.’

Rather than optimize the likelihood directly, we usually optimize the log-likelihood (natural log) given by

We can take the derivatives with respect to each of our parameters, set the resulting equations equal to zero and solve simultaneously. But wait! Notice that the maximizing the second term over and is equivalent to minimizing our sum of squared errors (i.e. least squares)!

Therefore, we get the same coefficient estimates for our intercept and slope terms whether we use least squares or maximum likelihood!

### Other Methods for Fitting the Regression Model

A drawback of least squares (and, hence, maximum likelihood with Normally distributed errors)is that it is susceptible to influential points. That is, points that have an unduly high impact on the regression process!

The figure below shows an example of an outlier and its impact of the regression fit.

|  |
| --- |
| Example of an outlier (red point). Shown are two regression lines fit with OLS: before (red dashed) and after (blue solid) removing the outlier. |

We could simply remove the outlier from the data set but that must be done carefully.

We can also use a different minimization criterion that is more robust to influential points.

* We can use the *least absolute deviation (LAD)* criterion,

or

* *Huber function*, which uses squared residuals when their values are small, but uses absolute value for large residuals (above a certain cutoff).

|  |
| --- |
| Visual of the loss associated with squared error loss and huber loss. Source: Wikipedia |

* Let’s compare the coefficient estimates of two MLR models: one using absolute error loss and one using squared error loss.

#uses the robustbase package  
mlr\_ae\_fit <- lmrob.lar(x = as.matrix(bike\_share |>   
 mutate(intercept = rep(1, nrow(bike\_share))) |>  
 select(intercept, temperature, wind\_speed)),  
 y = bike\_share$rented\_bike\_count)  
mlr\_ae\_fit$coefficients |>  
 kable()

| x |
| --- |
| 160.78571 |
| 26.55844 |
| 72.24026 |

Compare with the MLR fit:

mlr\_ls\_fit <- lm(rented\_bike\_count ~ temperature + wind\_speed, data = bike\_share)  
mlr\_ls\_fit$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 175.22135 |
| temperature | 29.35675 |
| wind\_speed | 87.64519 |

Let’s create a similar plot that showed how the OLS line was affected by the outlier, but use the absolute error loss for the model.

|  |
| --- |
| Example of an outlier (red point). Shown are two regression lines fit with absolute error loss: before (red dashed) and after (blue solid) removing the outlier. |

In general, *robust regression* methods are often of interest if data are prone to large outliers or have a heavy tailed distribution.

## Fitting non-linear Relationships in the Linear Model

As we mentioned, we can include polynomial terms, interaction terms, and more to allow our model to be more flexible. Let’s look at how to do this in R and the implications for interpretation.

The formula notation in R is an extremely powerful, concise, way for specifying the structure of a model!

The general syntax is response ~ predictor terms

* The right hand side contains the regression formula.
  + The intercept is automatically included for most models
  + To remove the intercept, we need to specify “y ~ -1 + predictors...”
  + We can include more than one ‘main-effect’ by adding predictors “y ~ pred1 + pred2”
  + We can include interaction terms using an asterisk or colon. “y ~ pred1 + pred2 + pred1:pred2” is equivalent to “y ~ pred1\*pred2”
  + We can include polynomial terms using the poly() function or by using I() and specifying the relationship. “y ~ pred1 + I(pred1^2)” or “y ~ poly(pred1, 2, raw = TRUE)”
  + Of course we can transform a variable prior to adding it to the formula as well (creating a log\_pred1 variable for instance)

### Including Polynomial Terms

Let’s fit a quadratic term for temperature to our SLR model. (We’ll use OLS as the default unless otherwise specified!)

#fit the quadratic relationship  
#equivalently, specify poly(temperature, 2, raw = TRUE)  
quad\_ols <- lm(rented\_bike\_count ~ temperature + I(temperature^2),  
 data = bike\_share)  
quad\_ols$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 329.7463715 |
| temperature | 28.4116825 |
| I(temperature^2) | 0.0286107 |

|  |
| --- |
| Least squares fit using a quadratic term. |

We need to be careful of interpretations now!

* is no longer the (expected) change in bike count for a unit change in temperature!
* Inference can still be done though! We must be careful that the colinearity of temperature with temperature^2 is accounted for (note the correlation between the two is 0.8796)
  + *We can use orthogonal polynomials to avoid this (which is the default of poly() in R!)*

### Including Interaction Terms

An **interaction** between two predictors implies that the effect of one predictor depends on the value of the other predictor.

* For example, consider the effect of temperature and wind\_speed on the rented\_bike\_count
* We have an interaction if the effect on rented\_bike\_countassociated with temperature is different for lower wind\_speeds than it is for higher wind\_speeds
* Equivalently, this would imply a differ effect for wind\_speed on rented\_bike\_count for lower temperatures as compared to higher temperatures
* This would make sense here as cold and windy is very different than warm and windy on how likely people are to be out and about, possibly renting bikes!

In R, we can fit interaction terms using the X1\*X2 notation.

* The formula Y ~ X1\*X2 will include *main effects* of and , and the *two-way interaction effect* in the model. - Alternatively, we can explicitly specify the interaction term in the formula: “Y ~ X1 + X2 + X1:X2”

#fit an interaction relationship  
interaction\_ols <- lm(rented\_bike\_count ~ temperature\*wind\_speed,  
 data = bike\_share)  
interaction\_ols$coefficients |>  
 summary()

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 4.685 16.504 28.326 84.523 96.344 276.753

* = (276.753) + (20.444)temperature + (36.208)wind\_speed + (4.685)(temperature)(wind\_speed)
* For temperature = 0, the slope on wind\_speed is (36.208)+0\* (4.685) = 36.208
* For temperature = 50, the slope on wind\_speed is (36.208)+50\*(4.685) = 270.473
* For temperature = 100, the slope on wind\_speed is (36.208)+100\*(4.685) = 504.738
* Similarly, the slope on temperature depends on wind\_speed!

|  |
| --- |
|  |

Note:

* We usually *retain all lower-order terms corresponding to an interaction* in the model
  + If the model has the term , we also retain terms and !

### Including Qualitative Predictors

So far we have only discussed models where ’s are continuous variables. We can accommodate categorical predictors as well. To do so, we need to create new *binary* predictors representing each of the categories of the original predictors.

#### Main Effects with Qualitative Predictors

As an example, consider the binary variable *holiday*. This takes on “Holiday” or “No Holiday”.

* We can create an **indicator variable** to replace this variable
* This takes on 1 (Holiday) or 0 (No holiday)
* Then we can include this as a predictor in our model and our design matrix!
* lm() automatically creates the variable for us (not all modeling functions do this)

MLR\_binary\_pred <- lm(rented\_bike\_count ~ temperature + holiday,  
 data = bike\_share)  
MLR\_binary\_pred$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 211.6793 |
| temperature | 28.9532 |
| holidayNo Holiday | 126.1416 |

The indicator variable created is called holidayNo Holiday. This variable takes on 1 when holiday takes on No Holiday. We can then see that our model has different intercepts but the same slope.

|  |
| --- |
| Regression line for No Holiday (0) and Holiday (1) based on a linear model with main effects for holiday and temperature. |

#### Interaction Effects with Qualitative Predictors

Now consider what happens when we add an interaction term between temperature and holiday.

MLR\_binary\_interaction <- lm(rented\_bike\_count ~ temperature\*holiday,  
 data = bike\_share)  
MLR\_binary\_interaction$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 151.024485 |
| temperature | 35.049304 |
| holidayNo Holiday | 190.309009 |
| temperature:holidayNo Holiday | -6.365576 |

Again, the indicator variable created is called holidayNo Holiday. This variable takes on 1 when holiday takes on No Holiday.

We can then see that our model has different intercepts and different slopes!

`geom\_smooth()` using formula = 'y ~ x'

|  |
| --- |
| Regression line for No Holiday (0) and Holiday (1) based on a linear model with main effects and an interaction effect for holiday and temperature. |

#### Qualitative Predictors with More Than Two Levels

The ideas presented above can be generalized to categorical variables with more that two levels. Suppose that is a variables with three levels, “L1”, “L2” and “L3”. Then we need to create *dummy variables* (or *indicator variables*):

We do not need a new dummy for level “L3” since

would encode “L3”.

In R, the first level alphabetically becomes the ‘reference’ level.

MLR\_qualitative\_interaction <- lm(rented\_bike\_count ~ temperature\*seasons,  
 data = bike\_share)  
MLR\_qualitative\_interaction$coefficients |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 405.987806 |
| temperature | 29.290777 |
| seasonsSpring | -360.073397 |
| seasonsSummer | -15.236868 |
| seasonsWinter | -153.707832 |
| temperature:seasonsSpring | 23.145254 |
| temperature:seasonsSummer | -5.090065 |
| temperature:seasonsWinter | -18.765620 |

* Finally, we should be cautious when there are many categorical variables in the data. Since we need to expand each of them into multiple indicator variables, the number of predictors can increase quickly.
* We need to be careful if we intend use data splitting methods like CV as well. Some levels may not appear in a training set in cases of levels that occur infrequently.
* Consider for example the ames\_raw housing data from the AmesHousing R package. We consider the variable SalePrice as response, and the rest as covariates.

AmesHousing::ames\_raw |>  
 select(1:5) |> #grab first five columns  
 slice(1:5) |>  
 kable()

| Order | PID | MS SubClass | MS Zoning | Lot Frontage |
| --- | --- | --- | --- | --- |
| 1 | 0526301100 | 020 | RL | 141 |
| 2 | 0526350040 | 020 | RH | 80 |
| 3 | 0526351010 | 020 | RL | 81 |
| 4 | 0526353030 | 020 | RL | 93 |
| 5 | 0527105010 | 060 | RL | 74 |

* Out of about 80 covariates, about 40 are categorical. However, after expanding each categorical variable into dummy variables, we will in fact have about predictors!
* If the sample size were smaller, say , standard techniques like 5-fold CV (training set size will be ) or split (training set size ) may produce unreliable results due to number of predictors being close to training set size. In general, we should always check the size of the model matrix before choosing a proper data splitting method.

## Inference

The linear model has many great properties that allow us to make inference! Let’s cover a few of this briefly.

### Standard errors

The estimated coefficients, and thus the fitted regression line are random quantities since they change from sample to sample. Thus we need a way to quantify the variability associated with the estimates!

To this end, we require additional assumptions on the error terms .

For simplicity, we will use the following set of assumptions:

* The errors are independently and identically distributed as .
* Errors are independent of the covariates

Note: There are ways to relax these assumptions. For example, if we have large sample size, we might relax the normality assumption under certain conditions on .

The assumptions above imply that

One way to quantify the variability associated with estimation of ’s is to compute the *standard error (SE)* of the estimates, defined as,

For a general multiple linear regression model with model matrix , the standard error of can be computed as

where corresponds to the intercept .

* A special case is the simple linear regression, where the standard errors, given fixed values of are
* Examining the expression of standard errors of and shows that SE will be smallest if the denominator is maximized.
* Thus, is smallest if the predictor values, ’s, are more spread out from their center.

Note that the standard error expressions depend on the error variance , which is an unknown quantity.

* We can estimate , and , using the residual sum of squares.
  + **Residual standard error (RSE)**

where is the number of columns in the model matrix . The resulting estimator of is known as **residual standard error (RSE)**

* We then plug-in in place of in the expressions of standard errors, to obtain *estimated standard errors*, . For simplicity of presentation, we will still denote the estimated standard error by .

The denominator in the expression of merits some discussion.

* We can view this number as “sample size - number of parameters in the mean function”.
* In SLR we have two parameters in mean function, and thus we have the term .
* In general, with predictors, the total number of parameters in the mean function is (since we need to include the intercept).

We can see the standard errors in R using the summary() function.

MLR\_ols <- lm(rented\_bike\_count ~ temperature + wind\_speed,  
 data = bike\_share)  
summary(MLR\_ols)$coefficients |>  
 kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | 175.22135 | 12.8919988 | 13.59148 | 0 |
| temperature | 29.35675 | 0.4796875 | 61.19975 | 0 |
| wind\_speed | 87.64519 | 5.5290775 | 15.85168 | 0 |

The first column of the output above are the estimates that we have discussed before. The second column gives the standard errors of the estimates.

Alternatively, we can directly use the following code:

* vcov() function produces the matrix
* diag() then extracts the diagonal elements of the matrix
* Taking the square root yields the SEs

se <- sqrt(diag(vcov(MLR\_ols)))  
se |>  
 kable()

|  | x |
| --- | --- |
| (Intercept) | 12.8919988 |
| temperature | 0.4796875 |
| wind\_speed | 5.5290775 |

The last part of the summary output above shows Residual standard error: 535.8952161. This is our .

* We can use the sigma() function on the model fit to directly obtain this value.

sigma(MLR\_ols)

[1] 535.8952

The standard errors, along with normality assumption on the errors, further enable us to perform statistical inference on the coefficients:

* Construct confidence intervals of : a set/range of values which contain the “true” value of with high probability. This can be investigated by a -statistic based confidence interval.
* Perform hypothesis tests to determine whether the -the predictor has any linear association with , vs. . This can be investigated using a -test.
* Perform hypothesis tests to determine whether the *any* of the predictors has any linear association with , vs.  at least on is non-zero. We can use -test to answer this question.

We discuss each of the items below.

### Confidence interval

Without going into mathematical details, we can obtain a confidence interval for using the standard errors.

**Confidence interval for**

* A confidence interval for is
* where denotes the quantile of a distribution.
* Again, the number is the same as we see in the estimator . This number is called the *degrees of freedom* of the -distribution used above.

|  |
| --- |
| t PDF and quantiles. The shaded region has area p, and the x-axis value corresponding to the solid vertical line represents the (1-p)-quantile. In this example, we have p = 0.05, and the vertical line represents the upper 0.95-quantile. |

In R, we can use the function confint() on our model fit to obtain *individual* confidence intervals for the regression coefficients.

## 95\% confidence intervals  
ci <- confint(MLR\_ols, level = 0.95)  
ci |>  
 kable()

|  | 2.5 % | 97.5 % |
| --- | --- | --- |
| (Intercept) | 149.95001 | 200.49270 |
| temperature | 28.41645 | 30.29705 |
| wind\_speed | 76.80690 | 98.48348 |

* We can interpret the intervals for intercept by saying that when temperature and wind\_speed are 0, on average rented\_bike\_count is estimated to be between 149.95 and 200.49 with confidence.
* We can interpret the interval for the temperature slope as follows: we estimate, with confidence, that rented\_bike\_count, on average, decreases between 28.42 and 30.3 for 1 unit increase in temperature if we hold wind\_speed constant.

### -test

We can also perform hypothesis tests on the regression coefficients.

* If our main interest is in testing the association between and , we test for
* Note that implies that is not in the model, and thus not associated with . We can use the -statistic to perform the test:

**-test for**

* The test statistic is
* The test statistic measures how far away the estimated value of is from zero compared to the variability of the estimate measured by .
* We expect the statistic to have a distribution *if* is true.
* We reject if the observed value of is very large or very small compared to what we expect from a distribution.

where is the CDF of the distribution evaluated at .

* We reject if the p-value if smaller than , where we set to be a small value (usually set to ).
* The quantity is called the type I error of the test (probability of rejecting when it should not be rejected).

|  |
| --- |
| Two-tailed p-value for a t-test. |

In R, we can use the summary() function to obtain the test results.

summary(MLR\_ols)$coefficients |>  
 kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | 175.22135 | 12.8919988 | 13.59148 | 0 |
| temperature | 29.35675 | 0.4796875 | 61.19975 | 0 |
| wind\_speed | 87.64519 | 5.5290775 | 15.85168 | 0 |

We see that the p-value associated with (coefficient of temperature) is very small, and thus reject . We conclude that temperature has a linear relationship with rented\_bike\_count, even in the presence of wind\_speed.

* Another way to test at is to check whether the value zero is in the confidence interval of or not.

confint(MLR\_ols) |>  
 kable()

|  | 2.5 % | 97.5 % |
| --- | --- | --- |
| (Intercept) | 149.95001 | 200.49270 |
| temperature | 28.41645 | 30.29705 |
| wind\_speed | 76.80690 | 98.48348 |

**Remember**: Caution must be taken to interpret results from model with interaction terms.

* Let us investigate the summary of a model fit on the Boston housing data we looked at previously.
* Let’s use main effects and an interaciton between lstat and age

boston\_interaction <- lm(medv ~ lstat\*age, data = Boston)  
summary(boston\_interaction)$coefficients |>  
 kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | 36.0885359 | 1.4698355 | 24.5527726 | 0.0000000 |
| lstat | -1.3921168 | 0.1674555 | -8.3133524 | 0.0000000 |
| age | -0.0007209 | 0.0198792 | -0.0362621 | 0.9710878 |
| lstat:age | 0.0041560 | 0.0018518 | 2.2442828 | 0.0252491 |

* Note that the interaction term is statistically significant but the *main effect of age* is not significant.
* As the significant interaction implies an effect of age on medv that differs with respect to lstat, we still conclude that age is important!
* We also do not want to remove the age main effect for interpretability purposes.

### -test

In the multiple linear regression with predictors, we investigate whether the linear model is at all useful by testing vs.  at least on is non-zero. This is called the **global F-test**

* In general, we can test for *any subset* of the predictors using -test, that is,
* where we are testing the effects of the last predictors (last columns of ).

**F$-test for**

* Let be the residual sum of squares of the model where we fit all the predictors **except the last** .
* Recall denotes the residual sum of squares for the full model.
* The -statistic is
* We reject if the observed value of is “large enough” or we calculate a p-value =

Let us revisit the interaction model for rented\_bike\_count with temperature, holiday and temperature\*holiday as predictors

* Mathematically, we write the model as
* where and correspond to temperature and a holiday indicator variable, respectively.
* Suppose we want to test the overall model utility, that is, jointly test the effects of all the three terms, . The F-test results can be found at the bottom of the summary output for this model:

summary(MLR\_binary\_interaction)$coefficients |>  
 kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | 151.024485 | 35.134674 | 4.298446 | 0.0000174 |
| temperature | 35.049304 | 2.362993 | 14.832588 | 0.0000000 |
| holidayNo Holiday | 190.309009 | 36.218011 | 5.254541 | 0.0000002 |
| temperature:holidayNo Holiday | -6.365576 | 2.414655 | -2.636225 | 0.0083982 |

summary(MLR\_binary\_interaction)$fstatistic |>  
 kable()

|  | x |
| --- | --- |
| value | 1206.041 |
| numdf | 3.000 |
| dendf | 8756.000 |

Note the large F-stat value. This corresponds to a p-value very close to 0.

* The very small p-value indicates that we reject , and the model is useful in predicting .
* Alternatively, we can fit two models: the full model (MLR\_binary\_interaction) and another model with only an intercept term.
* Then we can conduct -test ourselves using the anova() function in R.

# Reduced model with only intercept  
MLR\_red <- lm(rented\_bike\_count ~ 1, data = bike\_share)  
anova(MLR\_red, MLR\_binary\_interaction) |>  
 kable()

| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| --- | --- | --- | --- | --- | --- |
| 8759 | 3643934363 | NA | NA | NA | NA |
| 8756 | 2578468870 | 3 | 1065465493 | 1206.041 | 0 |

* Now suppose we want to test “whether *holiday* may have any association with response” or not. Since we have the interaction term, we have to test for both main and interaction effects of holiday.
* We can use the -test method to test

# Reduced model with only intercept and temperature is in SLR\_fit  
anova(SLR\_fit, MLR\_binary\_interaction) |>  
 kable()

| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| --- | --- | --- | --- | --- | --- |
| 8758 | 2587029843 | NA | NA | NA | NA |
| 8756 | 2578468870 | 2 | 8560973 | 14.53573 | 5e-07 |

Here we have observed -statistic with a p-value of . We reject , and can conclude that holiday does have association with the rented\_bike\_count, even in the prescense of temperature.

* Note: When , that is, we are testing for one predictor only, the -test and -test are equivalent.
* In fact, the square of the -statistic will give the -statistic!

#### Why is the Global -test Needed?

So why do we need the -test when we can examine -test results for each predictor in our model? You might think that if at least one -test gives a significant result we know our model is useful.

* The main issue is the *type I error* or *level* of the tests. That is, the probability of rejecting a null hypothesis when the null hypothesis is true.
* Typically, we set as level of the tests.
  + That means, for individual -tests, there is a chance that some effect will be detected as significant when it is actually not significant.
  + Looking at individual -tests, we should on average have about many false significant results, just by chance, even if none of the predictors might be useful.
  + In fact, for large , it is very likely we will observe at least one false significance just by chance!
* Using the F-statistic to test the model utility first does not suffer from this problem because it adjusts for the number of predictors.
  + If is true, there is only a chance that the F-statistic detect a false significance regardless of the number of predictors.

### Model diagnostics

To estimate standard errors, and to perform inference, we needed certain assumptions on the errors and the model as a whole:

* The relationship between and ’s is reasonably described by the linear regression model.
* Constant error variance .
* Errors are iid normally distributed.

Other practical issue include:

* Multicollinearity among predictors
* Presence of influential points

For our inference to be valid, we need to make sure the assumptions mentioned above are satisfied to some extent. We present some diagnostics methods to address each of the issues mentioned above.

#### Deviation from Linearity

To evaluate whether the relationship posited by the fitted regression model actually captures the true relationship, we can use *residual plots*.

* For simple linear regression, we can plot the residuals vs. the predictor.
* For multiple linear regression, it is easier to plot residual vs. the predicted responses.

If the model specification is adequate, there should be *no clear pattern* in the residual plot.

In contrast, any pattern in the residual plot would indicate the model does not capture the relationship between and well.

* In this case transforming data (either or or both) might prove useful.
* Alternatively, a non-linear/non-parametric regression might be considered as well.
* In an R session, you can use plot() on a fitted lm object to obtain some basic diagnostic plots. We’ll discuss these plots for one of our models after discussing other diagnostics.

#### Non-constant Variance of Errors

The standard errors, confidence intervals, and hypothesis testing procedures discussed so far depends on the assumption of constant variance of the errors:

* We call such errors *homoscedastic*.
* If errors have different variance, such phenomenon is called *heteroscedasticity*.

Large deviations from homoscedasticity may require a transformation of our variable(s).

#### Normality of errors

Normality of errors are needed for development of confidence intervals and testing procedures discussed above. However, this assumption can be relaxed for large enough sample size.

* Usually, visual displays such as a **normal Q-Q plot** of the residuals is used to check normality assumption.
* If the points align with the diagonal line well enough, we can conclude that the normality assumption is satisfactory. However, keep in mind that Q-Q plot is merely a visual tool, and often samples from non-normal distributions can produce normal like Q-Q plot (and vice-versa).

#### Influential points

Outliers and high leverage points can be detected using residual plots with *studentized residuals* and *leverage statistics*.

Recall our linear model is We estimate the regression parameter as Thus we we can predict the entire response vector by plugging-in as

where . The matrix is called the *hat matrix*. Detection of influential points depend on the following two results:

* It can be shown that variance of the -th residual, is equal to , where is the th diagonal entry of .
* The -th diagonal entry of , , is called the *leverage* of the -th observation.

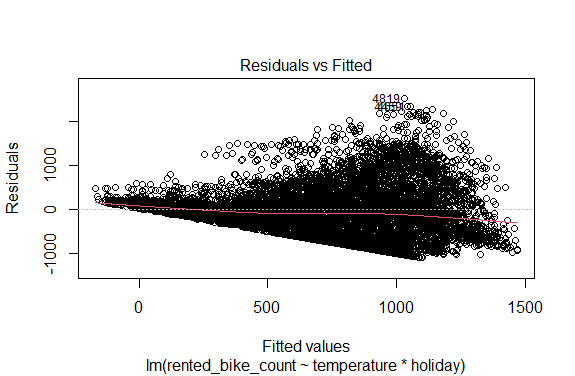
We define *studentized residuals* as residuals divided by their standard deviations.

* We can plot the studentized residuals against fitted values to detect outliers.
* Observations whose studentized residuals are quite far away from the rest (say more than three in absolute value) are possible outliers.

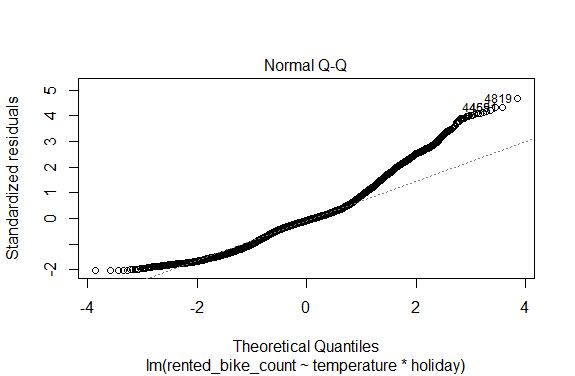
#### Investigating Assumptions

Let’s consider our MLR model for rented\_bike\_count that included temperature, holiday, and their interaction as predictors.

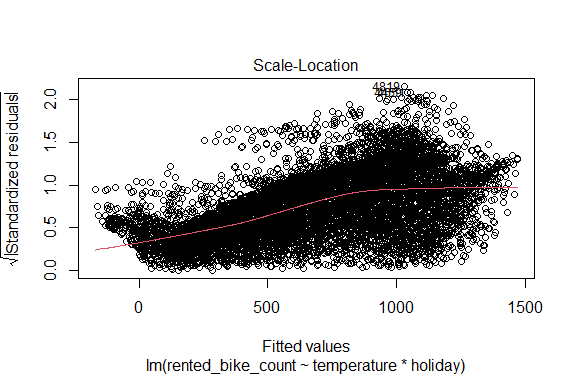
Running plot() on the fitted object yields the following four plots:



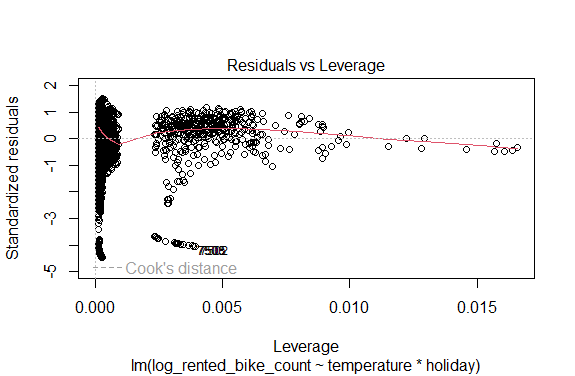
* This plot shows a reasonably distinct pattern in the residuals. There are a few more ‘large’ positive residuals as we have higher fitted values. This indicates that our model may not be doing as well as it could be at modeling the relationships in the data.
* We also see a definitive ‘trumpet’ shape. This indicates a non-constant variance.



* This qq-plot shows that the middle of the data roughly falls along the diagonal but there is a clear deviation for the lower and upper values. With a large sample size, this isn’t too concerning though.



* This plot is similar to the first but a bit more sophisticated as it uses the square root of the standardized residuals.
* These standardized residuals are given by taking the residual and dividing by the standard deviation times the square root of one minus the leverage:
* We again see a bit of a pattern through the residuals with smaller residuals for lower fitted values and larger for high fitted values.



* Nothing too striking appears here overall.

Overall, adding more predictors into the model would likely be quite useful here!

As mentioned earlier, we can use transformations to help make these assumptions more reasonable. Let’s do a **log** transform of our response (natural log).

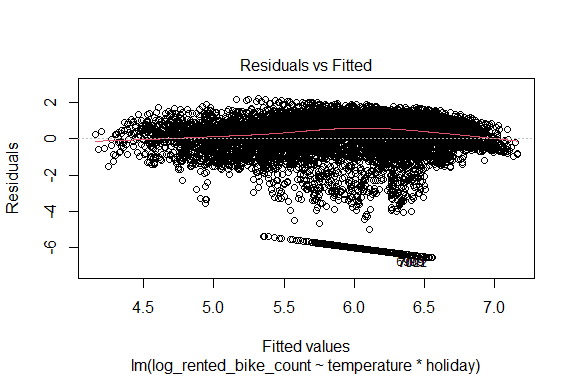
bike\_share <-   
 bike\_share |>  
 mutate(log\_rented\_bike\_count = log(rented\_bike\_count+1)) #add an offset for log(0) issues  
MLR\_log\_int <- lm(log\_rented\_bike\_count ~ temperature\*holiday, data = bike\_share)

First, let’s look at the summary of the model fit.

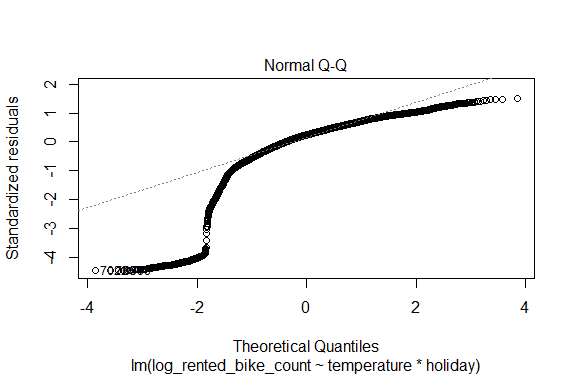
summary(MLR\_log\_int)$coefficients

Estimate Std. Error t value Pr(>|t|)  
(Intercept) 4.63036495 0.094855468 48.814950 0.000000e+00  
temperature 0.06659200 0.006379533 10.438382 2.332192e-25  
holidayNo Holiday 0.67252124 0.097780226 6.877886 6.491350e-12  
temperature:holidayNo Holiday -0.01937478 0.006519009 -2.972043 2.966298e-03

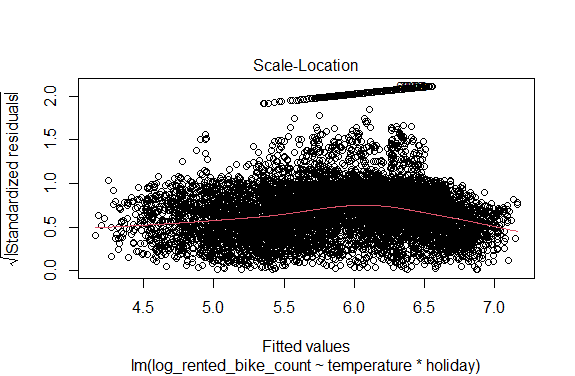
Great, now let’s look at the diagnostic plots to see if any issues were fixed.



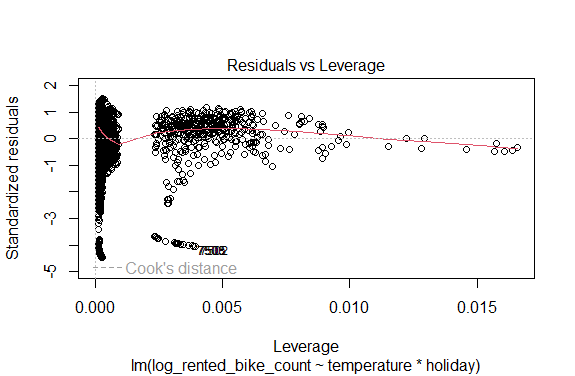
-Generally, our residuals seem to have far less of a trend. - However, there are some conserning observations that have very large residuals. These should be investigated further



* This got much worse but a little better at the same time. Again, the large sample size indicates we don’t have to be as concerned with the normality assumption.



* Here we again see vast improvement for the most part.
* The chunk of odd values at the top warrant further investigation.



* This looks a good bit worse due to the large negative observations. However, that may mostly be due to the group of weird observations mentioned earlier.

#### Collinearity

Lastly, we should discuss **collinearity**. This refers to high correlation between two or more predictors.

* Presence of such high correlation may lead to numerical instability of linear model fitting, reducing accuracy of estimation of regression coefficients, and reducing power of hypothesis tests.

Consider the two linear model fits for Boston data:

* Model A: tax and rad on medv
* Model B: lstat and tax on medv

The results are shown below:

| term | estimate | std.error | statistic | p.value |
| --- | --- | --- | --- | --- |
| **Model A** | | | | |
| (Intercept) | 35.6359 | 1.3465 | 26.4652 | 0.0000 |
| tax | -0.0386 | 0.0052 | -7.4847 | 0.0000 |
| rad | 0.2762 | 0.0997 | 2.7703 | 0.0058 |
| **Model B** | | | | |
| (Intercept) | 34.6128 | 0.5676 | 60.9848 | 0.0000 |
| lstat | -0.9326 | 0.0444 | -20.9986 | 0.0000 |
| rad | -0.0293 | 0.0364 | -0.8057 | 0.4208 |

Notice that in presence of tax the estimate and standard errors of rad changes drastically. This is because, tax and rad are highly correlated

* The figure below shows the correlation plot of Boston data, where we see that indeed tax and rad have high correlation.

|  |
| --- |
| Correlation plot of Boston data. |

If more that two predictors are closely related, we call the situation *multicollinearity*. Such situations can not be detected by simply inspecting the correlation plot. Instead, we may look at the *variance inflation factor (VIF)*.

**Variance inflation factor (VIF)**

* The variance inflation factor is the ratio of the variance of when fitting the full model to the variance if fit on its own.
* The VIF can be computed as follows:
* where is the value from a regression of onto the remaining predictors.

The minimum value of VIF is . As a rule of thumb, a VIF value larger than 5 or 10 indicates a problematic amount of multicollinearity.

* We can use car::vif() to calculate VIFs. In our example above, the VIF for each model are shown below.
* Model A
* Model B

In presence of multicollinearity, we can exclude the problematic predictors. Alternatively, we can combine the collinear predictors., e.g., taking an average.

We’ll investigate many different methods for building models that can account for this issue!

# Recap

We have now fully described our MLR model.

* We often fit the model using OLS or (a form of) maximum likelihood under the normal errors assumption
* We can conduct inference reasonably easily using this model. Diagnostic plots can help us feel more confident in the inferences we make
* We can include interaction terms, polynomial terms, and qualitative predictors through the use of indicator variables

The more predictors we include, the more flexible our model. But how flexible is too flexible? When do we have enough? That is what we take up next!

1. Generalized inverse of is a matrix such that . Although there are other definitions used by various authors. [↑](#footnote-ref-37)