Workshop Day 3

Day 3

Today we will switch gears to linear regression in R. We will use the same NHANES data that we used yesterday. We will use linear regression to understand the association between blood lead levels and systolic blood pressure, adjusting for possible confounders. To start, we create columns for systolic and diastolic blood pressure. If an observation has one blood pressure reading, then we use that value. If there is more than one blood pressure reading, then we drop the first observation and average the rest. We do a complete case analysis by dropping any observation with NA values. This leaves us with 30,405 observations.

```
suppressPackageStartupMessages(library(HDSinRdata))
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(broom))

data(NHANESsample)
```

```
NHANESsample$SBP <- apply(NHANESsample[,c("SBP1", "SBP2", "SBP3", "SBP4")], 1,
                          function(x) case_when(sum(!is.na(x)) == 0 ~ NA,
                                                sum(!is.na(x)) == 1 ~ mean(x, na.rm=TRUE),
                                                sum(!is.na(x)) > 1 ~ mean(x[-1], na.rm=TRUE)))
NHANESsample$DBP <- apply(NHANESsample[,c("DBP1", "DBP2", "DBP3", "DBP4")], 1,
                          function(x) case_when(sum(!is.na(x)) == 0 ~ NA,
                                                sum(!is.na(x)) == 1 ~ mean(x, na.rm=TRUE),
                                                sum(!is.na(x)) > 1 ~ mean(x[-1], na.rm=TRUE)))
nhanes_df <- na.omit(subset(NHANESsample, select= -c(SBP1, SBP2, SBP3, SBP4,
                                                     DBP1, DBP2, DBP3, DBP4)))
dim(nhanes_df)
```

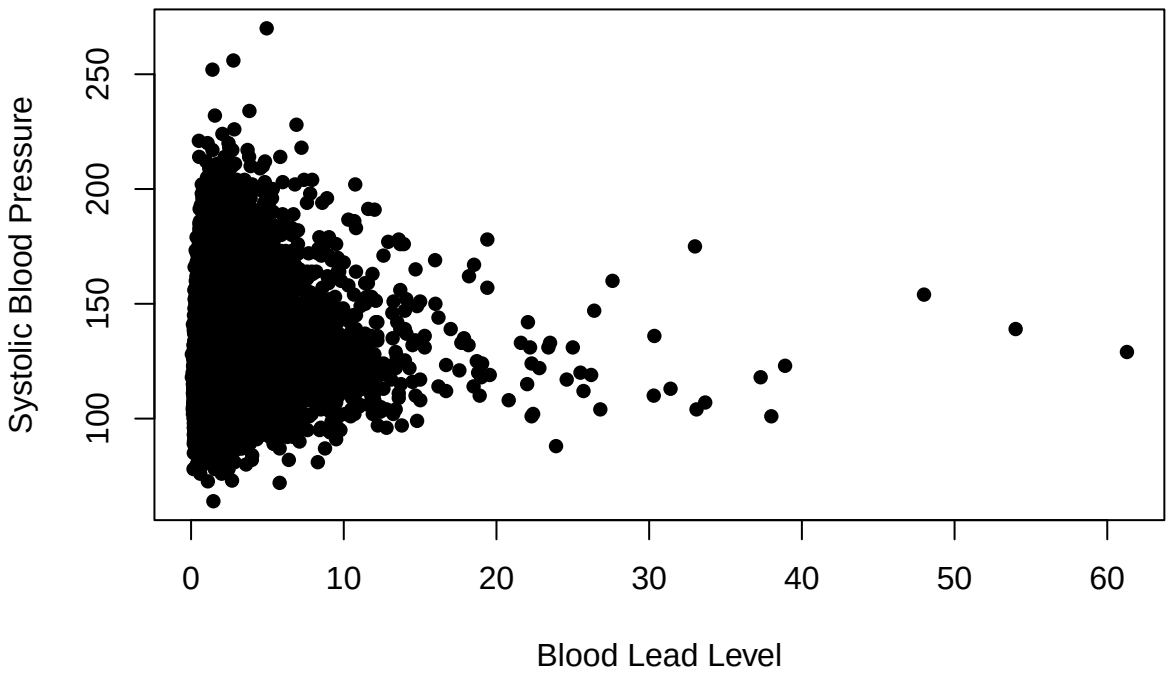
```
## [1] 30405    15
```

Next, we make sure any categorical variables are coded as factors.

```
nhanes_df$SEX <- as.factor(nhanes_df$SEX)
nhanes_df$RACE <- as.factor(nhanes_df$RACE)
nhanes_df$EDUCATION <- as.factor(nhanes_df$EDUCATION)
nhanes_df$BMI_CAT <- as.factor(nhanes_df$BMI_CAT)
nhanes_df$LEAD_QUANTILE <- as.factor(nhanes_df$LEAD_QUANTILE)
```

We will start with simple linear regression. Below, we plot the relationship between blood lead level and systolic blood pressure. For a simple linear regression scenario with a single continuous independent variable, a scatter plot allows us to easily visualize whether we meet the assumptions underlying linear regression. The survey sampling for the NHANES survey allows us to assume that each observation is independent. Looking at the plots below, we expect to see that the average systolic blood pressure increases linearly with blood lead level and that the observations look normally distributed with equal variance along that line. Below, we do not observe that to be the case. We will come back to this in the section on transformations and interactions.

```r
plot(nhanes_df$LEAD, nhanes_df$SBP,
     xlab = "Blood Lead Level", ylab = "Systolic Blood Pressure", pch=16)
```



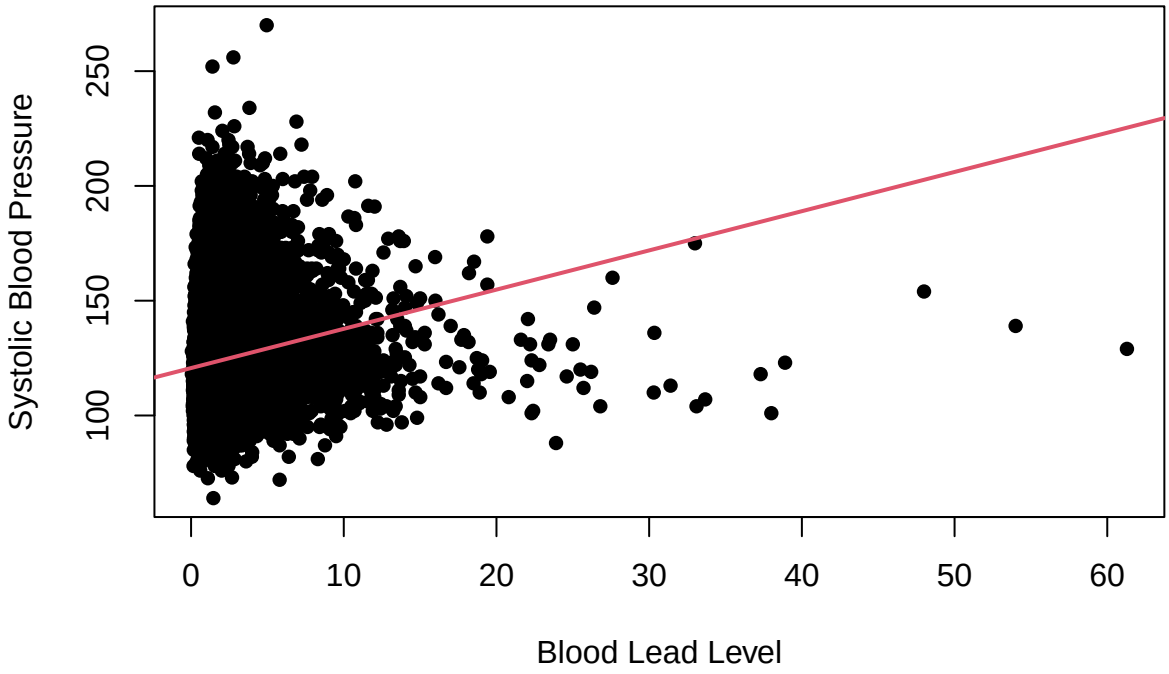
Despite our observations above, we will continue by fiting a simple linear regression model to explain the association between `SBP` and `LEAD`. The function `lm(formula = y ~ x, data)` fits a linear model in R. The first argument is the formula of the linear model: on the left hand side of the `~` we put the outcome variable, and on the right hand side we put the independent variable. When we have multiple indepedent variables we separate them with a `+` (e.g. `y~x1+x2`). The output of this function is an `lm` object. We can call the `summary()` function on this object to print a summary of the model, which includes the estimated coefficients, information about the residuals, the R-squared and adjusted R-squared values, and the F-statistic.

```r
simp_model <- lm(formula = SBP~LEAD, data = nhanes_df)
summary(simp_model)
```

```
##
## Call:
## lm(formula = SBP ~ LEAD, data = nhanes_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -96.360 -12.519  -2.791   9.363 140.880
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 120.66481    0.14950  807.11   <2e-16 ***
## LEAD          1.70820    0.05802   29.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.47 on 30403 degrees of freedom
## Multiple R-squared:  0.02772,    Adjusted R-squared:  0.02769
## F-statistic: 866.8 on 1 and 30403 DF,  p-value: < 2.2e-16
```

To visualize this model, we can add the estimated regression line to our scatter plot from above.

```
plot(nhanes_df$LEAD, nhanes_df$SBP,
     ylab=c("Systolic Blood Pressure"),
     xlab=c("Blood Lead Level"), pch=16)
abline(simp_model, col=2, lwd=2)
```



Multiple Linear Regression

We now create a model that is similar to the previous one except that it also adjusts for age and sex. To add these variables into the model, we have to specify a new formula. Below, we fit this model and then print a summary, again using the `summary()` function.

```
adj_model <- lm(SBP ~ LEAD + AGE + SEX, data = nhanes_df)
summary(adj_model)
```

```
##
## Call:
## lm(formula = SBP ~ LEAD + AGE + SEX, data = nhanes_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.623 -10.592  -1.551   8.547 131.595
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 101.785406   0.303532 335.337  < 2e-16 ***
## LEAD          0.400071   0.055246   7.242 4.54e-13 ***
## AGE           0.461932   0.005567  82.971  < 2e-16 ***
## SEXFemale    -2.777737   0.195674 -14.196  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.64 on 30401 degrees of freedom
## Multiple R-squared:  0.2116, Adjusted R-squared:  0.2115
## F-statistic:  2720 on 3 and 30401 DF,  p-value: < 2.2e-16
```

We can also extract the estimated regression coefficients from the model using the `coef()` function or by using the `tidy` function from the `broom` package. This function puts the coefficient estimates, standard errors, statistics, and p-values in a data frame. We can also add a confidence interval by specifying `conf.int = TRUE`. Below, we add a 95% confidence interval (which is the default value for `conf.level`).

```
tidy(adj_model, conf.int=TRUE, conf.level=0.95)
```

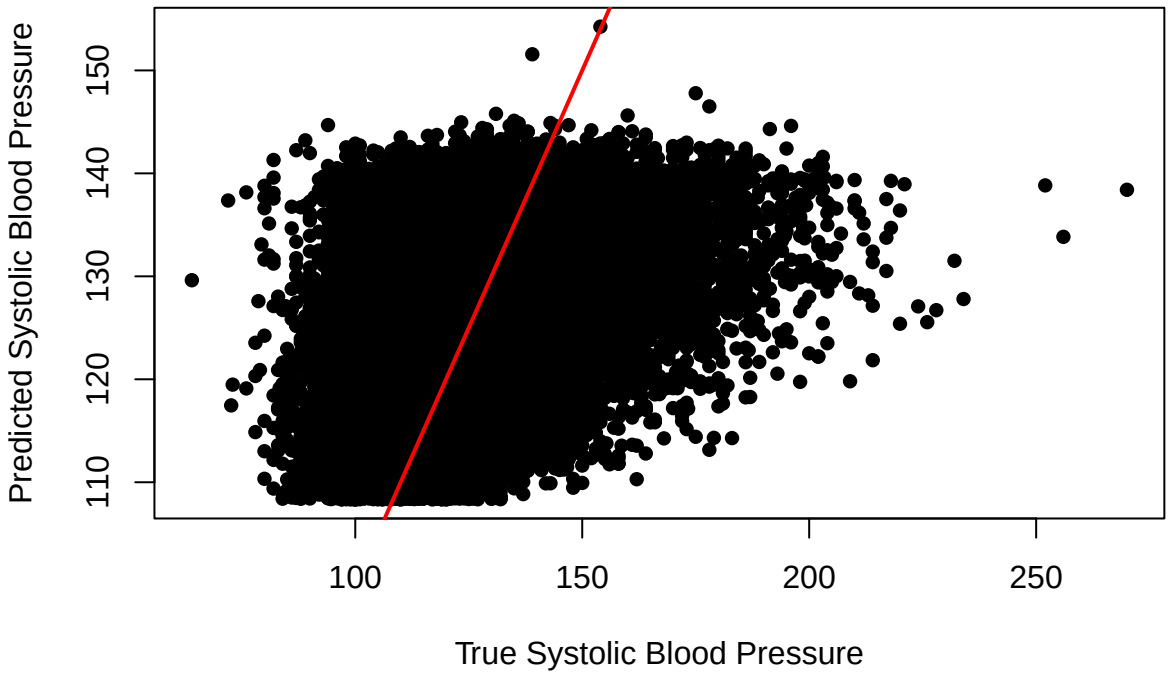
```
## # A tibble: 4 x 7
##   term        estimate std.error statistic  p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)  102.      0.304      335.   0         101.      102.
## 2 LEAD           0.400   0.0552       7.24 4.54e-13    0.292     0.508
## 3 AGE            0.462   0.00557     83.0  0           0.451     0.473
## 4 SEXFemale     -2.78    0.196      -14.2  1.36e-45   -3.16     -2.39
```

Some other useful summary functions are `resid()`, which returns the residual values for the model, and `fitted()`, which returns the fitted values or estimated y values. We can also predict on new data using the `predict()` function. Below we look at the distribution of the residual values and then plot the fitted vs. true values. We observe some extreme residual values as well as the fact that the absolute residual values increase with increased blood pressure values.

```
summary(resid(adj_model))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -65.624 -10.592  -1.551   0.000   8.547 131.595
```

```
plot(nhanes_df$SBP, fitted(adj_model),
     xlab ="True Systolic Blood Pressure", ylab="Predicted Systolic Blood Pressure", pch=16)
abline(a=0, b=1, col="red", lwd=2)
```



We can next perform a nested hypothesis test between our simple linear regresion model and our adjusted model using the `anova()` function. We pass both models to this function along with the argument `test="F"` to indicate that we are performing an F-test. The `print()` function shows the two tested models along with the associated p-value, which indicates a significantly better fit for the adjusted model.

```
print(anova(simp_model, adj_model, test="F"))
```

```
## Analysis of Variance Table
##
## Model 1: SBP ~ LEAD
## Model 2: SBP ~ LEAD + AGE + SEX
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  30403 10375769
## 2  30401  8413303  2   1962467 3545.6 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The model summary for the adjusted model displays the estimated coefficient for `sex` as `SEXFemale`, which indicates that the reference level for sex is male. If we want to change our reference level, we can reorder the factor variable either by using the `factor()` function and specifying `Female` as the first level or by using the `relevel()` function. The `ref` argument in the `relevel()` function specifies the new reference level. Now, when we run the model, we can see that the estimated coefficient for `sex` is labeled as `SEXMale`.

```
nhanes_df$SEX <- relevel(nhanes_df$SEX, ref="Female")
adj_model2 <- lm(SBP ~ LEAD + AGE + SEX, data = nhanes_df)
tidy(adj_model2)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   99.0     0.293      338.   0
```

```
## 2 LEAD           0.400   0.0552       7.24 4.54e-13
## 3 AGE            0.462   0.00557     83.0  0
## 4 SEXMale        2.78    0.196       14.2  1.36e-45
```

The formula passed to the `lm()` function also allows us to use the `.` to indicate that we would like to include all remaining columns as independent variables or the `-` to exclude variables. Below, we show how we could use these to fit a model with `LEAD`, `AGE`, and `SEX` as included covariates by excluding all other variables instead of by specifying these three variables themselves.

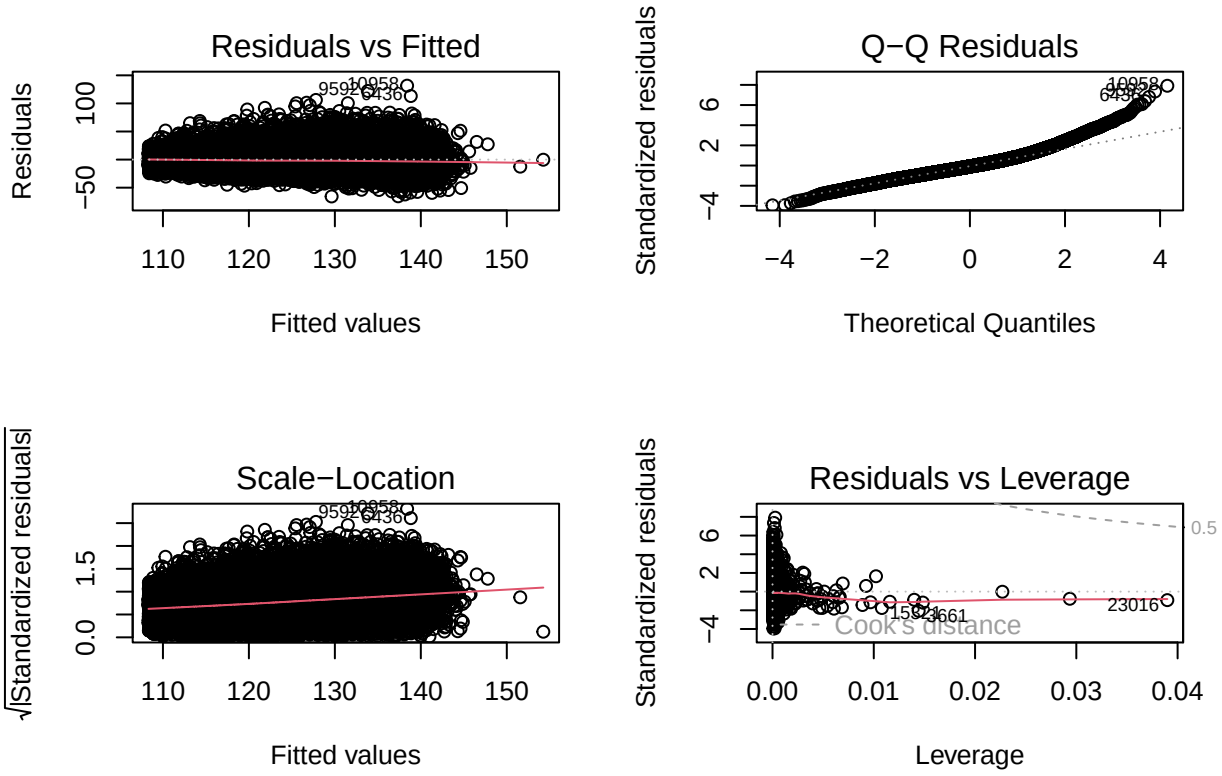
```
lm(SBP ~ . - ID - RACE - EDUCATION - INCOME - SMOKE - YEAR - BMI_CAT -
   LEAD_QUANTILE - DBP - ALC - HYP - RACE, data = nhanes_df)
```

```
##
## Call:
## lm(formula = SBP ~ . - ID - RACE - EDUCATION - INCOME - SMOKE -
##     YEAR - BMI_CAT - LEAD_QUANTILE - DBP - ALC - HYP - RACE,
##     data = nhanes_df)
##
## Coefficients:
## (Intercept)          AGE      SEXMale         LEAD
##     99.0077       0.4619       2.7777       0.4001
```

Diagnostic Plots and Measures

We can tell from the above plot that our model doesn't have a great fit. We will use some further diagnostic plots and measures to learn more. R has some built-in plots available for linear regression models, which can be displayed using the `plot()` function. The four plots include (a) Residuals vs. Fitted, (b) a QQ-plot for the residuals, (c) Standardized residuals (sqrt) vs. Fitted, and (d) Standardized Residuals vs. Leverage. In the last plot, you may observe that there is a dashed line. Any points outside of these lines have a Cook's distance of greater than 0.5. Additionally, points with labels correspond to the points with the largest residuals, so this last plot summarizes the outliers, leverage, and influential points. The plots below show that our residuals do not look normally distributed and that we have may have some high leverage points.

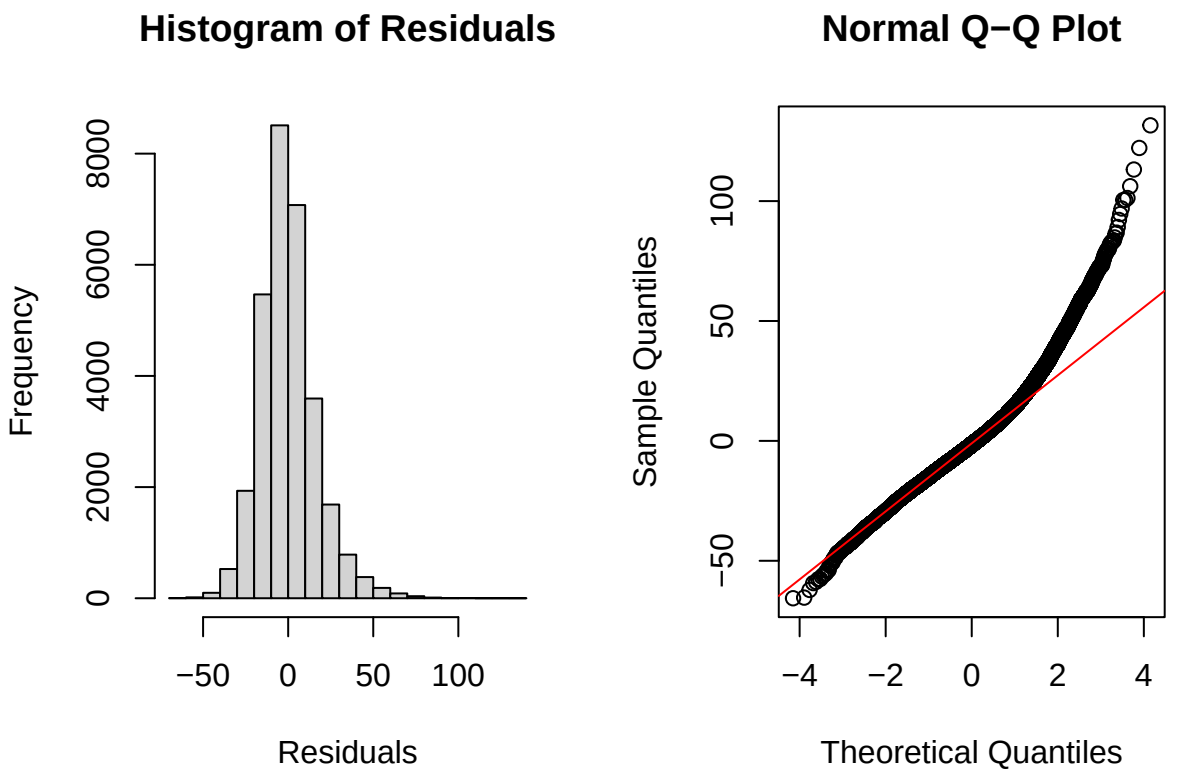
```
par(mfrow=c(2,2)) # plots all four plots together
plot(adj_model)
```



Normality

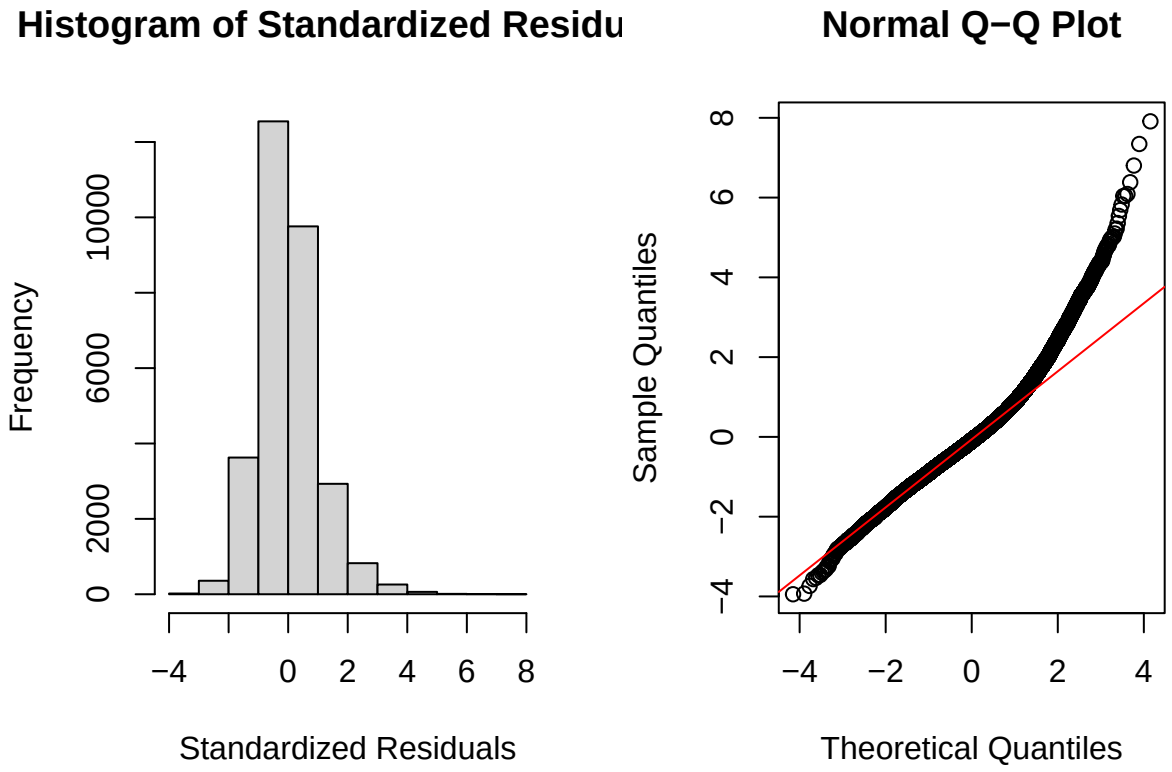
Beyond the default plots, we can also plot a histogram of the residuals and a qq-plot. The `qqnorm()` and `qqline()` functions can take in the residuals from our model as an argument. The latter adds the theoretical red line for reference. As both the histogram and qq-plot shown, the residuals are positively skewed, and thus the assumption of normality is not satisfied for our residuals. Later in this chapter, we will discuss how we might transform this dataset and/or model to satisfy this assumption.

```
par(mfrow=c(1,2)) # plot next to each other
hist(resid(adj_model), xlab="Residuals", main="Histogram of Residuals")
qqnorm(resid(adj_model))
qqline(resid(adj_model),col="red")
```



Instead of using the direct residuals, we can create the plots above using the standardized residuals with the function `rstandard()`. The standardized residuals are the raw residuals divided by an estimate of the standard deviation for the residual, which will be different for each observation.

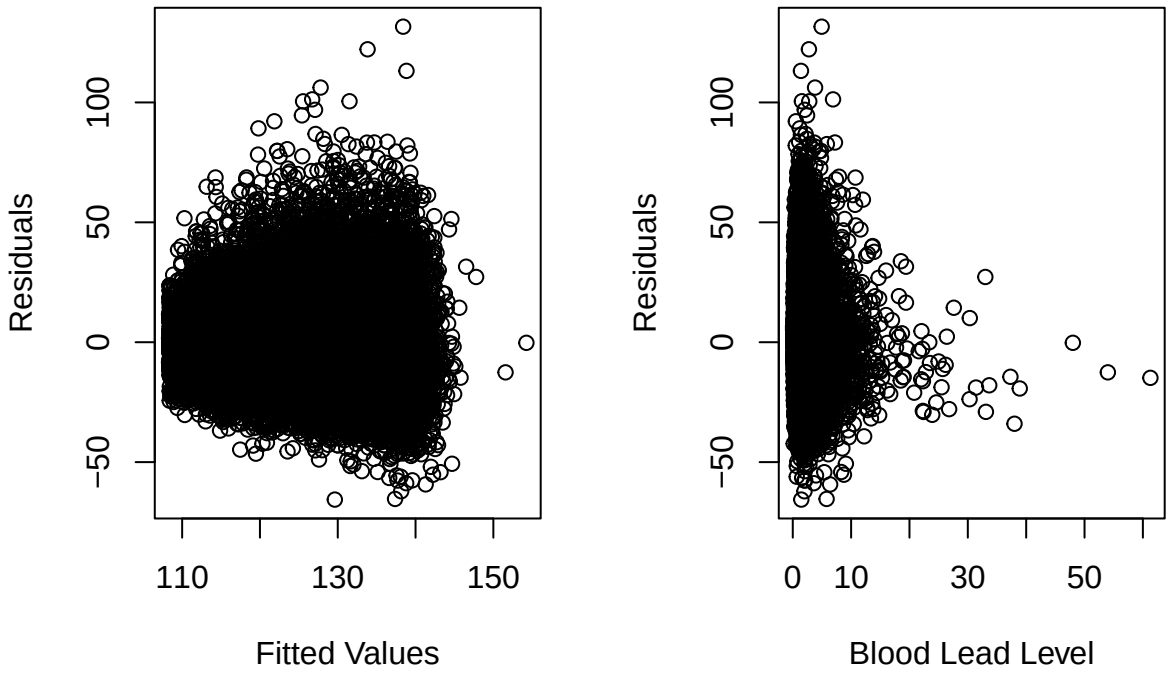
```
par(mfrow=c(1,2))
hist(rstandard(adj_model), xlab="Standardized Residuals",
     main="Histogram of Standardized Residuals")
qqnorm(rstandard(adj_model))
qqline(rstandard(adj_model),col="red")
```



Homoscedasticity, Linearity, and Collinearity

We can also create a residual vs. fitted plot or plot the residuals against included covariates. Below, we plot the blood lead level against the residuals. In both plots, we are looking for the points to be spread roughly evenly around 0 with no discerning pattern. However, both plots shows a tunnel shape, indicating a growing and shrinking variance of residuals by level, respectively. This indicates that we are violating the homoscedasticity assumption.

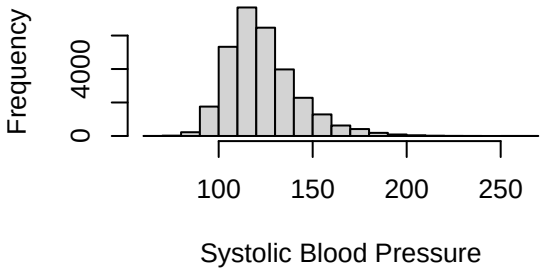
```
par(mfrow=c(1,2))
plot(fitted(adj_model), resid(adj_model), xlab="Fitted Values", ylab="Residuals")
plot(nhanes_df$LEAD, resid(adj_model), xlab="Blood Lead Level", ylab="Residuals")
```

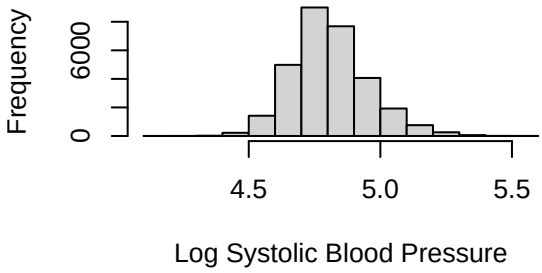


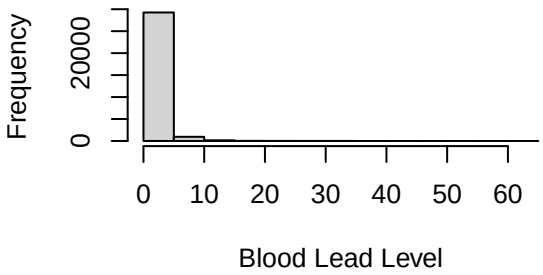
Interactions and Transformations

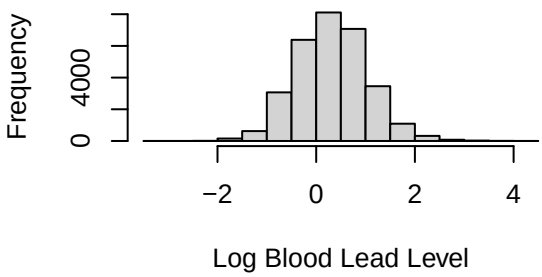
We now try to improve our model. To start, we look at potential transformations for our outcome variable. We will consider a log transformation for both our outcome, systolic blood pressure, and our predictor of interest, blood lead level. Both of these variables have a fairly skewed distribution and may benefit from such a transformation. Below, you can see that the transformed variables have distributions that are more symmetrical.

```
par(mfrow=c(2,2))
hist(nhanes_df$SBP, xlab="Systolic Blood Pressure", main="")
hist(log(nhanes_df$SBP), xlab="Log Systolic Blood Pressure", main="")
hist(nhanes_df$LEAD, xlab="Blood Lead Level", main="")
hist(log(nhanes_df$LEAD), xlab="Log Blood Lead Level", main="")
```









To add a transformation to a model, we can simply apply the transformation in the formula for `lm()`. We will calculate the adjusted R-squared for each potential model to compare their fits in addition to plotting the four qq-plots. Both indicate that the model with the log-log transformation (that is, with a log transformation applied to both the `SBP` and the `LEAD` variables) is the best fit.

```
model_nlog_nlog <- lm(SBP ~ LEAD + AGE + SEX, data = nhanes_df)
model_log_nlog <- lm(log(SBP) ~ LEAD + AGE + SEX, data = nhanes_df)
model_nlog_log <- lm(SBP ~ log(LEAD) + AGE + SEX, data = nhanes_df)
model_log_log <- lm(log(SBP) ~ log(LEAD) + AGE + SEX, data = nhanes_df)
```

```
summary(model_nlog_nlog)$adj.r.squared
```

```
## [1] 0.2115378
```

```
summary(model_log_nlog)$adj.r.squared
```

```
## [1] 0.2146037
```

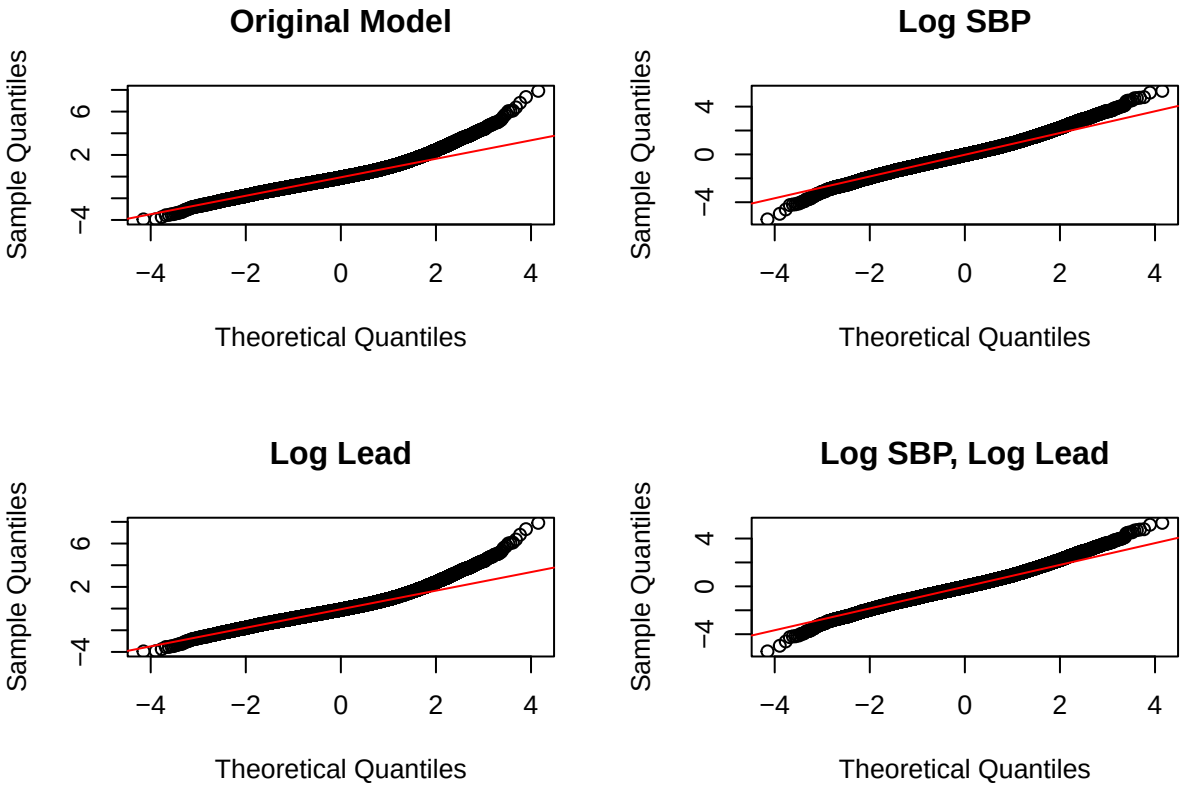
```
summary(model_nlog_log)$adj.r.squared
```

```
## [1] 0.211921
```

```
summary(model_log_log)$adj.r.squared
```

```
## [1] 0.2150189
```

```
par(mfrow=c(2,2))
qqnorm(rstandard(model_nlog_nlog), main="Original Model")
qqline(rstandard(model_nlog_nlog),col="red")
qqnorm(rstandard(model_log_nlog), main="Log SBP")
qqline(rstandard(model_log_nlog),col="red")
qqnorm(rstandard(model_nlog_log), main="Log Lead")
qqline(rstandard(model_nlog_log),col="red")
qqnorm(rstandard(model_log_log), main="Log SBP, Log Lead")
qqline(rstandard(model_log_log),col="red")
```



Additionally, we might consider polynomial transformations. The `poly(x, degree=1)` function allows us to specify a polynomial transformation where we might have higher degree terms. We do not pursue this for this particular example, but we show some example code below for creating such a transformation (in this case, a cubic transformation for blood lead level).

```
model_poly <- lm(SBP ~ poly(LEAD, 3) + AGE + SEX, data = nhanes_df)
```

We can summarize the outcome for our log-log model using the `tidy()` function again. We observe small p-values for each estimated coefficient.

```
tidy(model_log_log)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  4.62    0.00239     1932.   0
## 2 log(LEAD)    0.00891 0.00118        7.53 5.34e-14
```

```
## 3 AGE          0.00349 0.0000457     76.4  0
## 4 SEXMale      0.0254  0.00155       16.4  2.06e-60
```

Another component that we may want to add to our model is an interaction term. For example, we may consider an interaction between sex and blood lead level. We add an interaction to the formula using a `:` between the two variables. The output below shows that the coefficient for this interaction is indeed significant.

```
model_interaction <- lm(log(SBP) ~ log(LEAD) + AGE + SEX + SEX:log(LEAD), data=nhanes_df)
summary(model_interaction)
```

```
##
## Call:
## lm(formula = log(SBP) ~ log(LEAD) + AGE + SEX + SEX:log(LEAD),
##     data = nhanes_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69806 -0.08161 -0.00491  0.07524  0.65992
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         4.621e+00  2.387e-03 1936.15   <2e-16 ***
## log(LEAD)           2.364e-02  1.681e-03   14.07   <2e-16 ***
## AGE                 3.448e-03  4.578e-05   75.30   <2e-16 ***
## SEXMale             3.323e-02  1.668e-03   19.92   <2e-16 ***
## log(LEAD):SEXMale -2.660e-02  2.159e-03  -12.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1279 on 30400 degrees of freedom
## Multiple R-squared:  0.219,  Adjusted R-squared:  0.2189
## F-statistic:  2131 on 4 and 30400 DF,  p-value: < 2.2e-16
```

Evaluation Metrics

Besides the adjusted R-squared, there are a few other metrics that can help us to understand how well our model fits the data and to help with model selection. The `AIC()` and `BIC()` functions find the Akaike information criterion (AIC) and Bayesian information criterion (BIC) values, respectively. Both AIC and BIC balance the trade-off between model complexity and goodness of fit. AIC takes into account both the goodness of fit (captured by the likelihood of the model) and the complexity of the model (captured by the number of parameters used). Lower AIC values are preferable. BIC is similar to AIC but has a stronger penalty for model complexity compared to AIC. Both measures indicate a preference for keeping the interaction term.

```
AIC(model_log_log)
```

```
## [1] -38610.47
```

```
AIC(model_interaction)
```

```
## [1] -38759.96
```

```
BIC(model_log_log)
```

```
## [1] -38568.86
```

```
BIC(model_interaction)
```

```
## [1] -38710.02
```

The `predict()` function allows us to calculate the predicted `y` values. When called on a model with no data specified, it returns the predicted values for the training data. We could also specify new data using the `newdata` argument. The new data provided must contain the columns given in the model formula. Below, we use the `predict()` function to find the predicted values from our model and then calculate the mean absolute error (MAE) and mean squared error (MSE) for our model. MAE is less sensitive to outliers compared to MSE. The mean absolute error indicates that our model has fairly high residuals on average. While this model may be helpful to understand the relationship between blood lead level and systolic blood pressure, it would not be very useful as a tool to predict the latter.

```
pred_y <- predict(model_interaction)
```

```
mae <- mean(abs(nhanes_df$SBP - pred_y))
mae
```

```
## [1] 118.9627
```

```
mse <- mean((nhanes_df$SBP- pred_y)^2)
mse
```

```
## [1] 14501.92
```

Stepwise Selection

So far we have ignored the other variables in the data frame. When performing variable selection, there are multiple methods to use. We will end this chapter by demonstrating how to implement one such method, **stepwise selection**, in R. The `step()` function takes in an initial model to perform stepwise selection on along with a direction `direction` ("forward", "backward", or "both"), and a scope `scope`. The scope specifies the lower and upper model formulas to consider. Below, we use forward selection so the lower formula is the formula for our current model and the upper formula contains the other covariates we are considering adding in. These two formulas must be nested - that is, all terms in the lower formula must be contained in the upper formula.

By default, the `step()` function prints each step in the process and uses AIC to guide its decisions. We can set `trace=0` to avoid the print behavior and update the argument `k` to `log(n)` to use BIC, where `n` is the number of observations. Below we see that the algorithm first adds in race, then BMI, then income, then education, and then smoking status. In fact, all variables were added to the model! The final output is an `lm` object that we can use just like the ones earlier in this chapter. We get the summary of the final model and see that the adjusted R-squared has improved to 0.2479.

```
mod_step <- step(model_interaction, direction = 'forward',
                 scope = list(lower = "log(SBP) ~ log(LEAD) + AGE + SEX:log(LEAD)",
                              upper = "log(SBP) ~ log(LEAD) + AGE + SEX:log(LEAD) + SEX + RACE + EDUCATI
             INCOME + BMI_CAT"))
```

```
## Start:  AIC=-125047.6
## log(SBP) ~ log(LEAD) + AGE + SEX + SEX:log(LEAD)
##
##             Df Sum of Sq    RSS     AIC
## + RACE       4    9.1605 488.19 -125605
## + BMI_CAT    2    8.9722 488.38 -125597
## + INCOME     1    2.8709 494.48 -125222
## + EDUCATION  2    1.8966 495.46 -125160
## + SMOKE      2    0.3496 497.00 -125065
## <none>                   497.35 -125048
##
## Step:  AIC=-125604.9
## log(SBP) ~ log(LEAD) + AGE + SEX + RACE + log(LEAD):SEX
##
##             Df Sum of Sq    RSS     AIC
## + BMI_CAT    2    7.1612 481.03 -126050
## + INCOME     1    1.7985 486.39 -125715
## + EDUCATION  2    1.3404 486.85 -125684
## + SMOKE      2    0.1297 488.06 -125609
## <none>                   488.19 -125605
##
## Step:  AIC=-126050.2
## log(SBP) ~ log(LEAD) + AGE + SEX + RACE + BMI_CAT + log(LEAD):SEX
##
##             Df Sum of Sq    RSS     AIC
## + INCOME     1   1.61678 479.42 -126151
## + EDUCATION  2   1.11221 479.92 -126117
## + SMOKE      2   0.26139 480.77 -126063
## <none>                   481.03 -126050
##
## Step:  AIC=-126150.5
## log(SBP) ~ log(LEAD) + AGE + SEX + RACE + BMI_CAT + INCOME +
##     log(LEAD):SEX
##
##             Df Sum of Sq    RSS     AIC
## + EDUCATION  2   0.41789 479.00 -126173
## + SMOKE      2   0.25818 479.16 -126163
## <none>                   479.42 -126151
##
## Step:  AIC=-126173
## log(SBP) ~ log(LEAD) + AGE + SEX + RACE + BMI_CAT + INCOME +
##     EDUCATION + log(LEAD):SEX
##
##         Df Sum of Sq    RSS     AIC
## + SMOKE  2   0.28648 478.71 -126187
## <none>               479.00 -126173
##
## Step:  AIC=-126187.2
## log(SBP) ~ log(LEAD) + AGE + SEX + RACE + BMI_CAT + INCOME +
##     EDUCATION + SMOKE + log(LEAD):SEX
```

```
summary(mod_step)
```

```
##
```

```
## Call:
## lm(formula = log(SBP) ~ log(LEAD) + AGE + SEX + RACE + BMI_CAT +
##     INCOME + EDUCATION + SMOKE + log(LEAD):SEX, data = nhanes_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67127 -0.07992 -0.00393  0.07377  0.67974
##
## Coefficients:
##                            Estimate Std. Error  t value Pr(>|t|)
## (Intercept)              4.614e+00  3.316e-03 1391.511  < 2e-16 ***
## log(LEAD)                2.282e-02  1.694e-03   13.473  < 2e-16 ***
## AGE                      3.485e-03  4.848e-05   71.872  < 2e-16 ***
## SEXMale                  3.466e-02  1.655e-03   20.945  < 2e-16 ***
## RACEOther Hispanic      -7.106e-03  3.223e-03   -2.205   0.0275 *
## RACENon-Hispanic White  -4.449e-03  2.198e-03   -2.024   0.0430 *
## RACENon-Hispanic Black   3.369e-02  2.466e-03   13.662  < 2e-16 ***
## RACEOther Race           6.270e-03  3.388e-03    1.850   0.0643 .
## BMI_CAT25<BMI<30         1.513e-02  1.838e-03    8.230  < 2e-16 ***
## BMI_CATBMI>=30           3.783e-02  1.835e-03   20.623  < 2e-16 ***
## INCOME                  -3.886e-03  4.996e-04   -7.778 7.57e-15 ***
## EDUCATIONHS             -1.935e-05  2.186e-03   -0.009   0.9929
## EDUCATIONMoreThanHS     -8.692e-03  2.068e-03   -4.203 2.64e-05 ***
## SMOKEQuitSmoke          -7.556e-03  1.796e-03   -4.207 2.60e-05 ***
## SMOKEStillSmoke         -4.037e-03  1.943e-03   -2.077   0.0378 *
## log(LEAD):SEXMale       -2.606e-02  2.122e-03  -12.278  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1255 on 30389 degrees of freedom
## Multiple R-squared:  0.2483, Adjusted R-squared:  0.2479
## F-statistic: 669.1 on 15 and 30389 DF,  p-value: < 2.2e-16
```

Generalized Linear Models in R

The `glm(formula, data, family)` function in R is used to fit generalized linear models. The three main arguments we must specify to the function are the * `formula` - specifies the relationship between the independent variables and the outcome of interest, * `data` - the dataset used to train the model, and * `family` - a description of the error distribution and link function to be used in the model.

In binary logistic regression, we assume a binomial outcome and use the logit link function. We can specify this by setting `family = binomial`. By default, this will assume the link function is the logit function. Note that we can even use the `glm()` function to implement linear regression by setting `family = gaussian`.

Our outcome of interest will be current e-cigarette use, `e_cig_use`. We need to create this variable from the variables currently in the data. We set `e_cig_use` to 0 if the respondent answered that they have not used e-cigarettes in the last 30 days and 1 otherwise. We can see that there are only 1,435 respondents who reported e-cigarette use. This is a low percentage of the overall sample and will likely impact our results.

```
nyts$e_cig_use <- as.factor(ifelse(nyts$num_e_cigs==0, "0", "1"))
table(nyts$e_cig_use)
```

```
##
##     0     1
```

```
## 18683  1435
```

Looking at the covariate of interest, survey setting, we can see that there are 85 respondents that took the survey in "Some other place". Since we are interested in the impact of taking the survey at school compared to other settings, we will simplify this variable to have two levels: "school" and "home/other".

```
table(nyts$location)
```

```
##
##     At home (virtual learning) In a school building/classroom
##                           8738                           10737
##               Some other place
##                             85
```

```
nyts$location <- as.factor(ifelse(nyts$location == "In a school building/classroom", "school", "home/ot
```

To start, we will create a model to predict e-cigarette use from school setting adjusting for identified covariates sex, school level, and race and ethnicity. Note that we specify our formula and data as with the `lm()` function. We then use the `summary()` function again to print a summary of this fitted model. The output is slightly different from an `lm` object. Now, we see the null and residual deviances are reported along with the AIC. Adding transformations and interactions is equivalent to that in the `lm()` function and is not demonstrated in this chapter.

```
mod_start <- glm(e_cig_use ~ grade + sex + race_and_ethnicity + location,
                 data = nyts, family = binomial)
summary(mod_start)
```

```
##
## Call:
## glm(formula = e_cig_use ~ grade + sex + race_and_ethnicity +
##     location, family = binomial, data = nyts)
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                              -4.60174    0.15388 -29.905  < 2e-16
## grade7th                                  0.44606    0.17532   2.544 0.010951
## grade8th                                  0.96768    0.16071   6.021 1.73e-09
## grade9th                                  1.38300    0.15494   8.926  < 2e-16
## grade10th                                 1.91832    0.15132  12.677  < 2e-16
## grade11th                                 2.13852    0.14913  14.340  < 2e-16
## grade12th                                 2.42858    0.14916  16.282  < 2e-16
## gradeUngraded or Other Grade              2.52132    0.44875   5.619 1.93e-08
## sexFemale                                 0.19221    0.05795   3.316 0.000912
## race_and_ethnicitynon-Hispanic Black     -0.66139    0.11209  -5.900 3.63e-09
## race_and_ethnicitynon-Hispanic other race -0.10205    0.15152  -0.674 0.500606
## race_and_ethnicitynon-Hispanic White      0.19835    0.07388   2.685 0.007260
## locationschool                            0.72234    0.06484  11.141  < 2e-16
##
## (Intercept)                              ***
## grade7th                                 *
## grade8th                                 ***
## grade9th                                 ***
```

```
## grade10th                                    ***
## grade11th                                    ***
## grade12th                                    ***
## gradeUngraded or Other Grade                 ***
## sexFemale                                    ***
## race_and_ethnicitynon-Hispanic Black         ***
## race_and_ethnicitynon-Hispanic other race
## race_and_ethnicitynon-Hispanic White         **
## locationschool                               ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9754.9  on 18746  degrees of freedom
## Residual deviance: 8886.8  on 18734  degrees of freedom
##   (1666 observations deleted due to missingness)
## AIC: 8912.8
##
## Number of Fisher Scoring iterations: 6
```

We can use the `tidy()` function from the `broom` package to display the estimated coefficients. This time we add the `exponentiate = TRUE` argument to exponentiate our coefficients so we can interpret them as estimated change in odds rather than log odds. For example, those who answered at school have double the estimated odds of reporting e-cigarette use compared to those who took the survey at home/other, adjusting for grade, sex, and race and ethnicity.

```
tidy(mod_start, exponentiate=TRUE)
```

```
## # A tibble: 13 x 5
##    term                                  estimate std.error statistic   p.value
##    <chr>                                    <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                             0.0100    0.154    -29.9   1.68e-196
##  2 grade7th                                1.56      0.175      2.54  1.10e-  2
##  3 grade8th                                2.63      0.161      6.02  1.73e-  9
##  4 grade9th                                3.99      0.155      8.93  4.41e- 19
##  5 grade10th                               6.81      0.151     12.7   7.94e- 37
##  6 grade11th                               8.49      0.149     14.3   1.23e- 46
##  7 grade12th                              11.3       0.149     16.3   1.32e- 59
##  8 gradeUngraded or Other Grade           12.4       0.449      5.62  1.93e-  8
##  9 sexFemale                               1.21      0.0580     3.32  9.12e-  4
## 10 race_and_ethnicitynon-Hispanic Black    0.516     0.112     -5.90  3.63e-  9
## 11 race_and_ethnicitynon-Hispanic other ~  0.903     0.152     -0.674 5.01e-  1
## 12 race_and_ethnicitynon-Hispanic White    1.22      0.0739     2.68  7.26e-  3
## 13 locationschool                          2.06      0.0648    11.1   7.93e- 29
```

Exercises

1. Construct a linear model using `DBP` as the output and `LEAD`, `AGE`, and `EVER_SMOKE` as features, and print the output.
2. Use forward stepwise selection to determine whether to add any interactions to the linear model from the previous question.

3. Look at some diagnostic plots for the model and use what you observe from these plots to choose a transformation that will improve the fit of this model. Then, fit and summarize this new model with the transformation included. How do the AIC and BIC of the new model compare to the previous one?

4. Report the MAE and MSE of the model developed in Question 3. Then, find the row numbers of the observations with the top 10 largest standardized residuals for this model. What do you notice about these observations?