Lab_05T

36-290 – Statistical Research Methodology

Week 5 Tuesday – Fall 2021

Preliminaries

Goal

The goal of this lab is to perform, and to interpret the results of, a multiple linear regression analysis.

Data

We'll begin by importing some data:

```
rm(list=ls())
file.path = "https://raw.githubusercontent.com/pefreeman/36-290/master/EXAMPLE_DATASETS/DM_GALAXY/Massive_Black_II.
Rdata"
load(url(file.path))
rm(file.path)
objects()
```

```
## [1] "pred.test"     "pred.train"    "resp.test.df"  "resp.train.df"
```

The split of the data into training and test sets was done for us by the client. For background on what these data represent, click here (https://github.com/pefreeman/36-290/tree/master/EXAMPLE_DATASETS/DM_GALAXY) and read the description. In short, the predictor variables are measurements relating to the dark matter haloes in which visible galaxies are embedded, and the response variables are measurements relating to the galaxies. To goal when building a statistical model would be to link the dark matter halo properties to a galaxy property, i.e., to be able to predict galaxy mass given a set of dark matter halo measurements, and/or determine which measurements most inform galaxy mass.

Digression: what is dark matter?

It turns out that when we look at large-scale structures, like gravitationally bound clusters of galaxies, the movements of those galaxies cannot be explained by the amount of light that we see. We basically understand balls of gas (stars) and thus when we see a galaxy of a particular brightness we expect it to have a certain mass. But galaxies and clusters of galaxies are more massive than they appear…about five times more massive. The leading explanation for why there is missing mass is that there is a constituent of the Universe that interacts with "normal," or "baryonic" matter via gravitation, but not via electromagnetism. The force carrier for electromagnetism is the photon: if matter does not interact via electromagnetism, then it must be "dark."

Dark matter is a "parameterization of ignorance." We don't know *what* it is, but we see what it does. This is kind of like gravitation itself between Newton and Einstein: in the 1800s, we could model gravity (we could see what it does and could build physical models based on mathematical laws) but we didn't know *what* gravity was. (The theory for gravity is General Relativity.)

Anyway: dark matter is such that it is easy to simulate the evolution of a Universe containing only dark matter particles. Basically, over time dark matter goes from being very uniformly distributed to clumping in structures dubbed "halos." In the simplest, most intuitive picture, you can envision a galaxy as sitting inside a halo: the halo is like a big sphere, and the galaxy is more like a disk inside the sphere. (Due to its properties, dark matter doesn't collapse to form structures like galaxies…the "light matter" has collapsed more over time, and the dark matter less.) The overall mass of a given halo is a random variable, with the underlying distribution of halo masses being something astronomers would like to accurately and precisely constrain. Problem is…we don't actually observe dark matter, we observe the "light matter," or baryonic matter. So: if we see a galaxy with particular properties (like mass and star-formation rate), can we predict the properties of the dark matter halo in which it is embedded? Or vice-versa?

Digression over…

To keep things simple, we'll concentrate on one response variable initially, galaxy stellar mass:

```
mass.train = resp.train.df$halos.m_star
mass.test  = resp.test.df$halos.m_star
rm(resp.test.df,resp.train.df)
```

Questions

To answer the questions below, it will help you to refer to Chapter 3 of ISLR (and especially Section 6); it might also help you to refer to your previous lab work (and, as always, to Google).

Question 1

The datasets above are *large*. Not too large to fit via multiple linear regression, but too large to effectively plot. Determine how many galaxies are in `pred.train` , set a random number seed, sample some number of indices, and histogram the predictors and the mass variable. Assess whether or not you should transform the mass variable in any way. For now, in this question, **you do not need to make any transformation…just note whether you think a transformation might be needed, and propose one**. (If so, look at the variable transformations text mentioned in the notes to propose possible transformations. Specifically, hone in on what John Tukey called the "ladder of transformations.") Also look at the predictor variables…while it is not explicitly required that one transform them to have a normal distribution (or any distribution for that matter), sometimes transformation can improve fits. Use facet wrapping for the predictors, and pass the argument `scales="free"` to `facet_wrap()` to allow the x-limits for each histogram to be different.

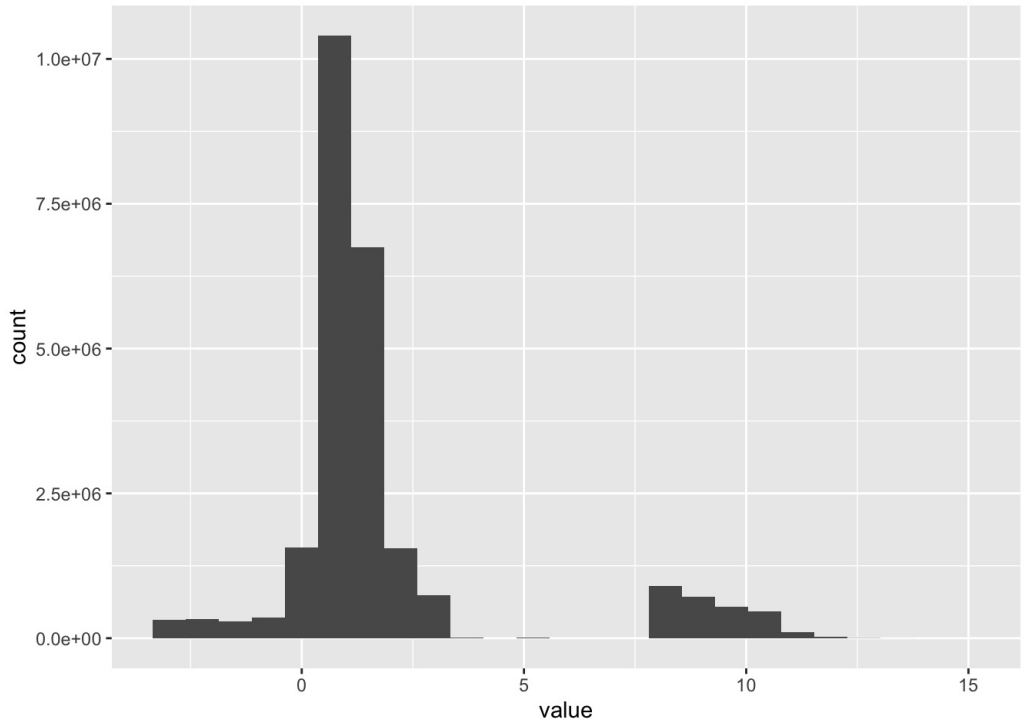
```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

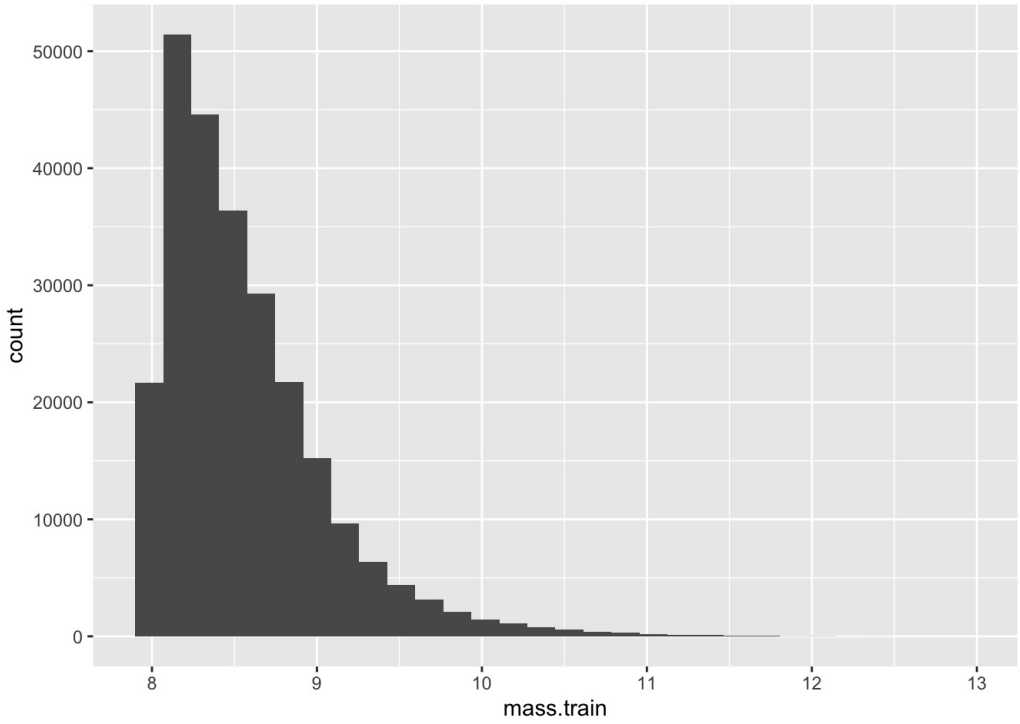
```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.4     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   2.0.1     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
set.seed(103)

df = data.frame(x= pred.train, y= mass.train)
sample = gather(sample(df, 100, replace=TRUE))
#sample = sample(1:nrow(pred.train),100, replace=TRUE)

ggplot(data=sample,mapping=aes(x=value),facet_wrap(scales='free')) + geom_histogram(bins=25)
```




```
ggplot(data=data.frame(mass.train),mapping=aes(x=mass.train)) + geom_histogram(bins=30)
```

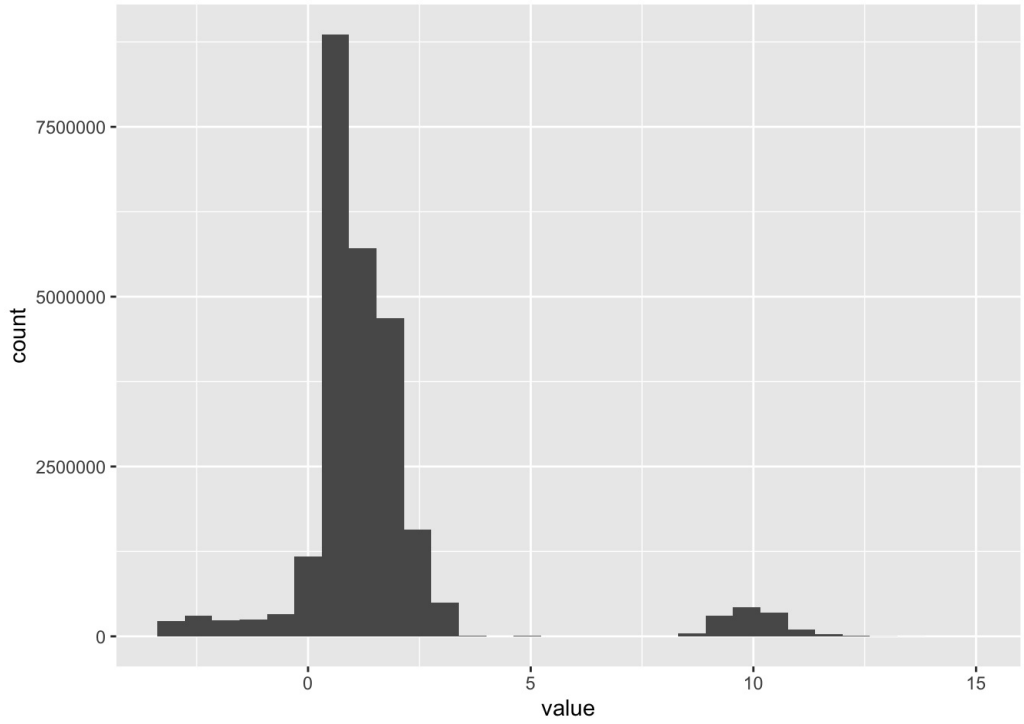



```
A transformation might be needed because the response variable is assumed to be distributed normally when we go to
run linear regression. Possible transformations: log transform, square root (more moderate of a transformation comp
ared to log, could be used if log overcorrects), negative reciprocal (stronger transformation,keeps the order of nu
mbers the same, might be used if log undercorrects data). I would maybe suggest the use of maybe a square root or l
og transformation to correct the upward straggle.


"ladder of transformations":


stronger:  -1/x^2

mild:  -1/x

(corrects upward straggle)

no shape change: x

(corrects downward straggle)

mild: x^2, x^3

stronger: antilog x
```

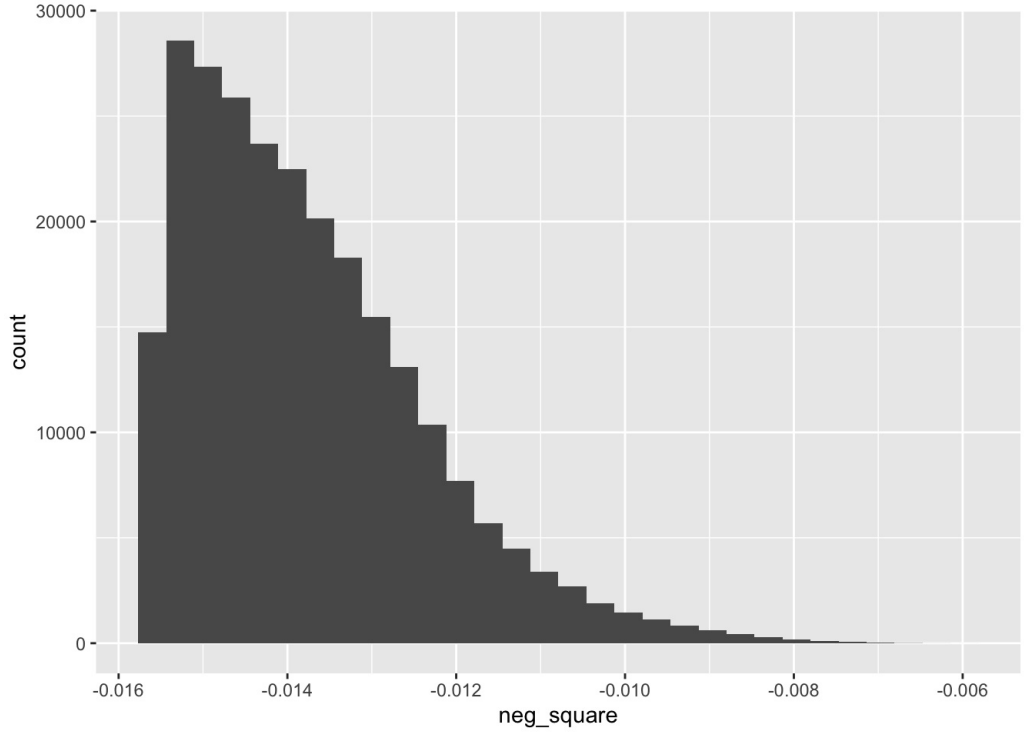
Question 2

If you stated in Q1 that a transformation would be helpful, apply it here and show the histogram. Did it help make the response variable more normal? (Make a new variable with the transformed response…do not permanently change the original response.) If you didn't think a transformation would be helpful, just plot the response as you did in Q1.

```
set.seed(103)

library(tidyverse)

l.mass.train = log(mass.train)

df = data.frame(x= pred.train, y= l.mass.train)

neg= -1/(mass.train)
neg_square = -1/(mass.train)^2
square =(mass.train)^2

sample = gather(sample(df, 100, replace=TRUE))


ggplot(data=sample,mapping=aes(x=value),facet_wrap(scales='free')) + geom_histogram(bins=30)
```




```
ggplot(data=data.frame(neg_square),mapping=aes(x=neg_square)) + geom_histogram(bins=30)
```




```
It didn't really help too much, the shape of it is still not close to normal.
```

Before moving on to performing linear regression, you should learn a bit about model syntax. Here we show this syntax within the context of a simple analysis:

```
> lm.out = lm(mass.train~.,data=pred.train)
> summary(lm.out)
> mass.test.pred = predict(lm.out,newdata=pred.test)
```

Let's break this down.

First, we call `lm()`, which stands for "linear model." For our model, we decide to regress the mass variable `mass.train` onto all the predictor variables (represented by the "."). (Note: that's a tilde before the period, not a minus sign! See below for what we would do if we don't want to include all the predictor variables when learning the model.) `R` doesn't know where these predictor variable are, so we specify that via the `data=` argument. We save the output as `lm.out`.

Second, we call the `summary()` function. `summary()` is a general function whose behavior depends on the class of object passed to it. If the object is a data frame, you get a numerical summary. (Try it with `pred.train`!) If the object is of class `lm`, then you get entirely different output. (Basically, you can think of it as `summary()` checking for the class, then calling another function depending on what the class is. Here,

`summary()` invisibly redirects `lm.out` to `summary.lm()`.) The `summary()` function provides the p-values for the individual coefficients and for the F statistic, plus the adjusted R^2, etc.

Third, we use the model embedded within `lm.out` to generate predictions for the mass for new data (the test data). `predict()` behaves like `summary()`; here, it redirects the arguments to `predict.lm()`.

Now, about the model syntax. For simplicity, assume that we have a predictor data frame `p`, with columns `a`, `b`, and `c`, and a response vector `r`.

- To include `a` only: `lm(r~a,data=p)`
- To include `a` and `c` only: `lm(r~a+c,data=p)` or `lm(r~.-b,data=p)`
- To regress through the origin: `lm(r~.-1,data=p)`
- To include `a`, `b`, and their interaction: `lm(r~a+b+a:b,data=p)`

There's more, but life is short and so's this course!

Question 3

Perform a multiple linear regression analysis. Use the `summary()` function to examine the output. Do you conclude that the linear model is informative or uninformative? (Also look at the p-values for the individual coefficients and for the F statistic…are these particularly informative here?)

```
lm.out = lm(mass.train~.,data=pred.train)
summary(lm.out)
```

```
##
## Call:
## lm(formula = mass.train ~ ., data = pred.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3713 -0.0763 -0.0039  0.0701  1.2256
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   3.7457991  0.0096359  388.733  < 2e-16 ***
## halos.vdisp  -1.0241673  0.0074772 -136.972  < 2e-16 ***
## halos.vcirc   4.6090480  0.0069571  662.492  < 2e-16 ***
## halos.rcirc   0.0771571  0.0014184   54.397  < 2e-16 ***
## halos.m_dm   -0.1602274  0.0012962 -123.611  < 2e-16 ***
## r_parent      0.0412111  0.0002906  141.814  < 2e-16 ***
## shapesDM.q3d -0.1115770  0.0040357  -27.647  < 2e-16 ***
## shapesDM.s3d -0.0107378  0.0035017   -3.066  0.00217 **
## thetaTid     -0.0072308  0.0005977  -12.098  < 2e-16 ***
## phiTid        0.0000710  0.0001295    0.548  0.58350
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1201 on 251058 degrees of freedom
## Multiple R-squared:  0.9392, Adjusted R-squared:  0.9392
## F-statistic: 4.31e+05 on 9 and 251058 DF,  p-value: < 2.2e-16
```

```
mass.test.pred = predict(lm.out,newdata=pred.test)
```

```
I conclude that the linear model is informative. The p values for the coefficients are all less  than the .05 thres
hold except for phiTid. Also the F statistic has a significant p value of less than 2.2e-16.
```

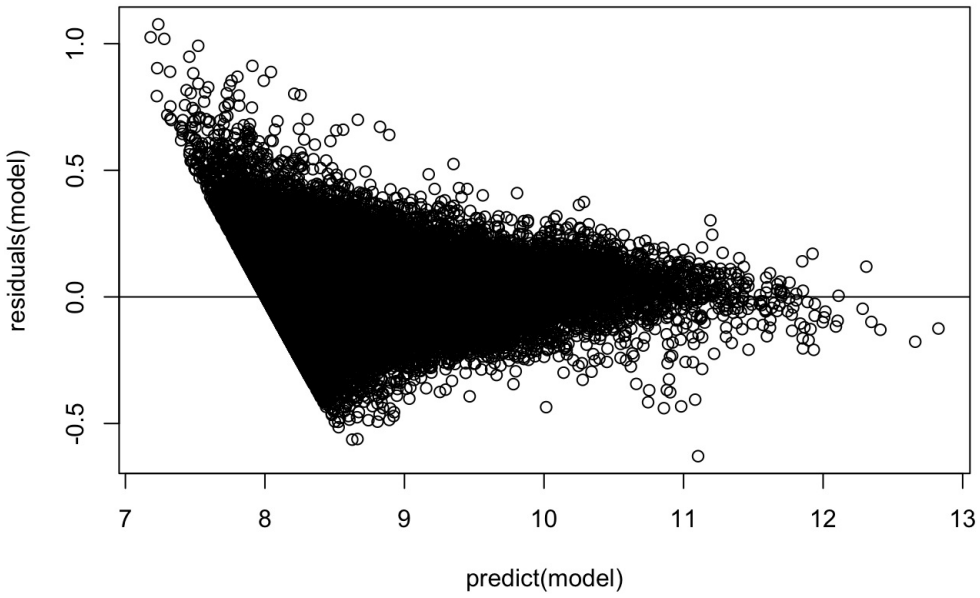
Question 4

Plot the residuals (observed minus predicted masses) versus the predicted mass, for the test data. What trends do you see? (Be sure to only plot a subset!) (Just note any trends…do not necessarily try to do anything about them.)

```
model = lm(mass.test~.,data=pred.test)
summary(model)
```

```
##
## Call:
## lm(formula = mass.test ~ ., data = pred.test)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.62900 -0.07649 -0.00366  0.07019  1.07658
##
## Coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   3.7115748  0.0136137  272.636  < 2e-16 ***
## halos.vdisp  -1.0761391  0.0105762 -101.751  < 2e-16 ***
## halos.vcirc   4.6676383  0.0098763  472.612  < 2e-16 ***
## halos.rcirc   0.0787344  0.0019985   39.396  < 2e-16 ***
## halos.m_dm   -0.1595573  0.0018243  -87.460  < 2e-16 ***
## r_parent      0.0412494  0.0004102  100.549  < 2e-16 ***
## shapesDM.q3d -0.1023050  0.0057189  -17.889  < 2e-16 ***
## shapesDM.s3d -0.0233639  0.0049631   -4.708 2.51e-06 ***
## thetaTid     -0.0061222  0.0008444   -7.251 4.17e-13 ***
## phiTid        0.0001271  0.0001826    0.696    0.486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1187 on 123651 degrees of freedom
## Multiple R-squared:  0.9399, Adjusted R-squared:  0.9399
## F-statistic: 2.148e+05 on 9 and 123651 DF,  p-value: < 2.2e-16
```

```
plot(predict(model), residuals(model))
abline(0, 0)
```




```
There is kind of a more broader spread of points near the left and then they narrow in towards the right.
```

Question 5

Load the `car` library (install it if necessary!) and use the `vif()` function to check for possible (multi)collinearity. If present, remove a variable and redo the fit. Rinse, lather, and repeat until the `vif()` outputs are all less than 5. Do the results change markedly from those in Q3? (Make a mental note that before you interpret your R^2 value in a linear model fit, you need to check for multicollinearity first, since the latter can increase the value of the former. How does the R^2 look after you are done here?)

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
vif(lm.out)
```

```
##  halos.vdisp  halos.vcirc  halos.rcirc   halos.m_dm     r_parent shapesDM.q3d
##    36.732766    21.344405     3.407582    13.045960     1.511159     1.896764
## shapesDM.s3d     thetaTid       phiTid
##     1.881207     1.021637     1.000024
```

```
removed.lm.out = lm(mass.train~halos.rcirc+r_parent+shapesDM.q3d+shapesDM.s3d+thetaTid+phiTid,data=pred.train)


vif(removed.lm.out)
```

```
##  halos.rcirc     r_parent shapesDM.q3d shapesDM.s3d     thetaTid       phiTid
##     1.093093     1.101090     1.877044     1.874004     1.013848     1.000013
```

```
summary(removed.lm.out)
```

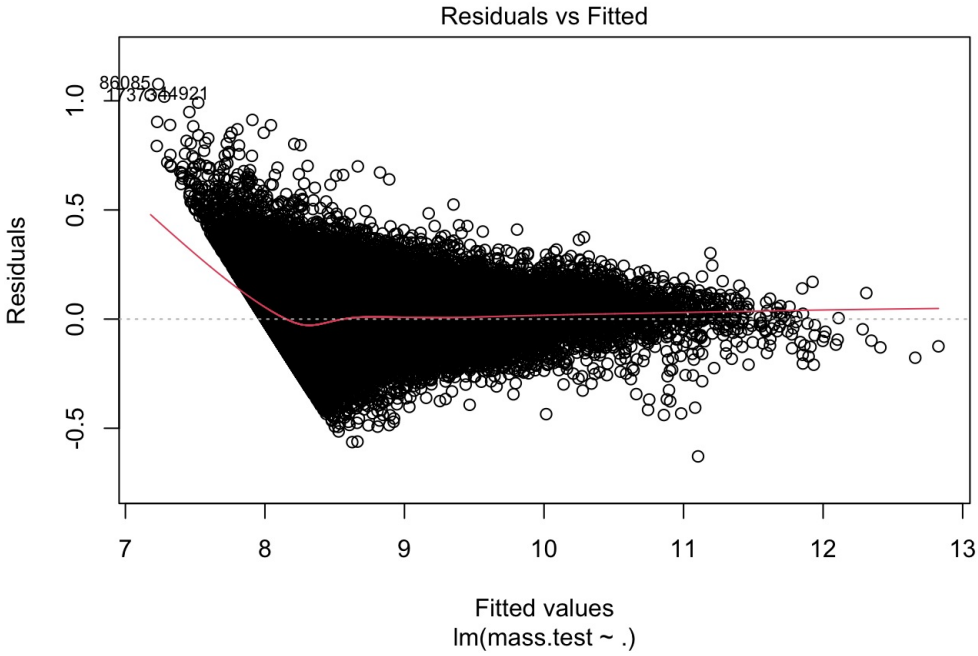
```
##
## Call:
## lm(formula = mass.train ~ halos.rcirc + r_parent + shapesDM.q3d +
##     shapesDM.s3d + thetaTid + phiTid, data = pred.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5164 -0.2557 -0.0385  0.2178  4.7893
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.3056203  0.0089943 812.251   <2e-16 ***
## halos.rcirc   0.9871853  0.0025722 383.795   <2e-16 ***
## r_parent      0.1118088  0.0007942 140.776   <2e-16 ***
## shapesDM.q3d  0.2624464  0.0128543  20.417   <2e-16 ***
## shapesDM.s3d  0.2948745  0.0111903  26.351   <2e-16 ***
## thetaTid     -0.0533758  0.0019064 -27.998   <2e-16 ***
## phiTid       -0.0001484  0.0004146  -0.358     0.72
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3845 on 251061 degrees of freedom
## Multiple R-squared:  0.3769, Adjusted R-squared:  0.3769
## F-statistic: 2.531e+04 on 6 and 251061 DF,  p-value: < 2.2e-16
```

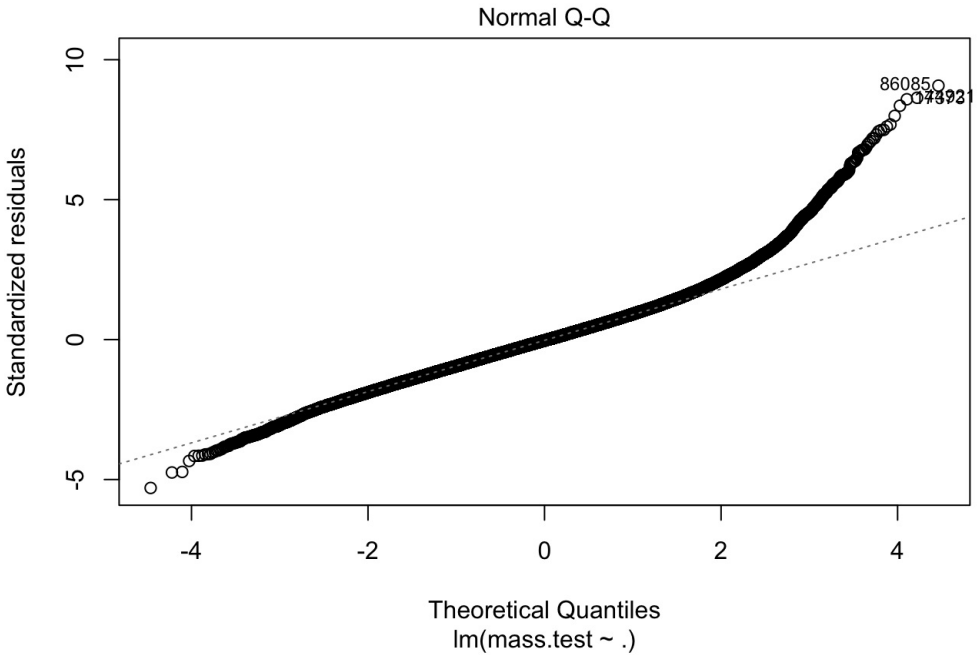
```
The R squared value is a lot lower compared to before the variables were removed.
```

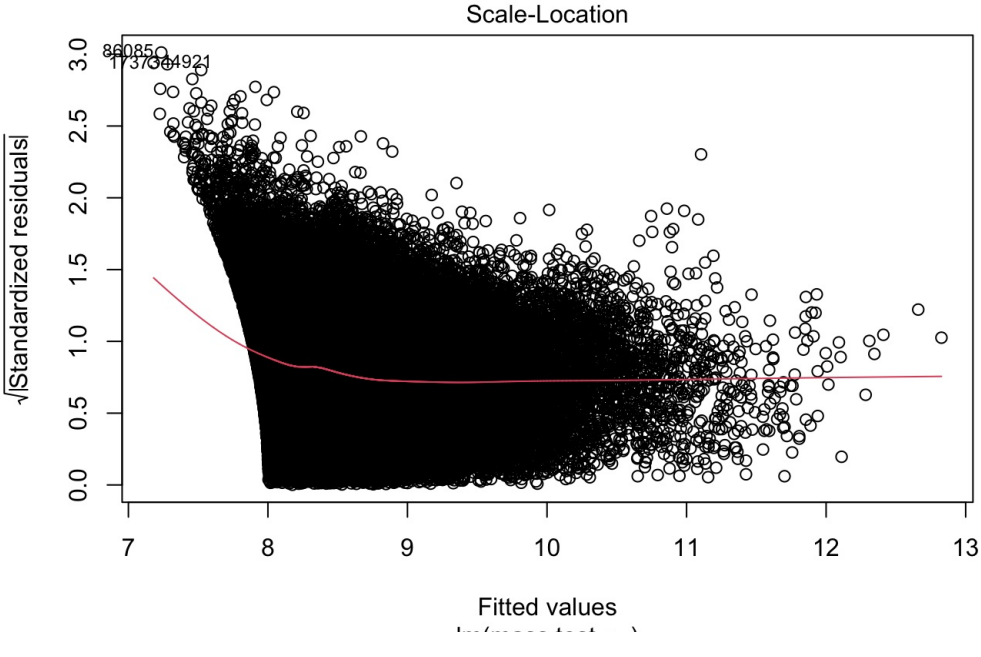
Question 6

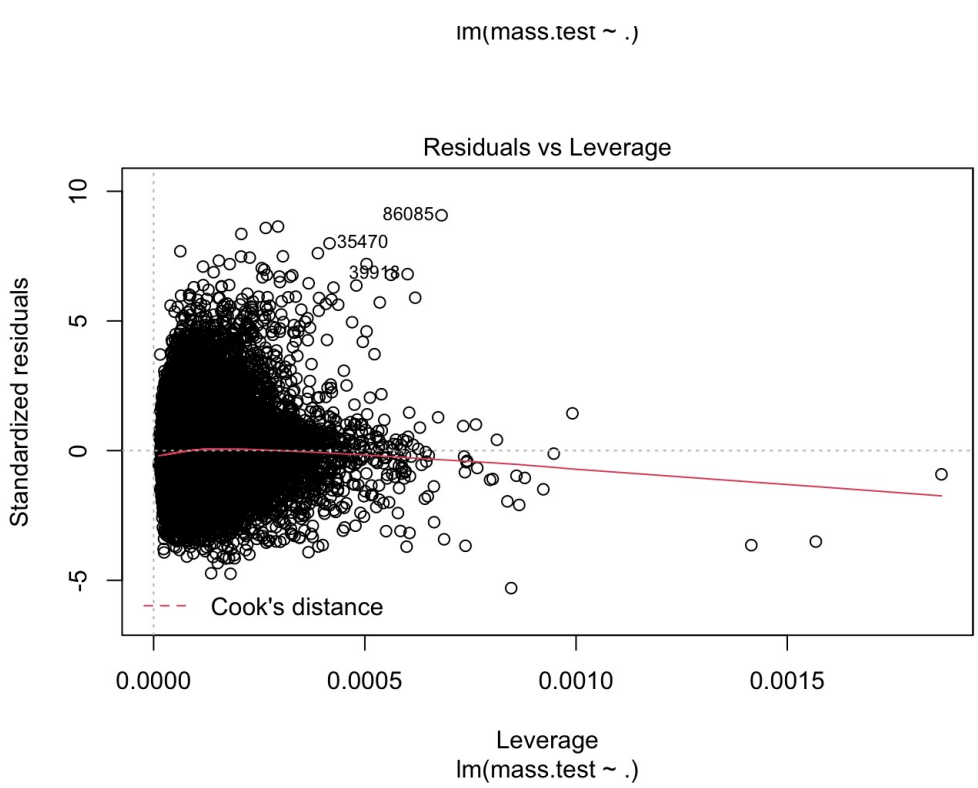
Create two diagnostic plots wherein you show the predicted mass versus the observed mass, and compute the test MSEs, for analyses with the full set of predictor variables and for the vif-reduced set of predictor variables. For the plot, make sure to make the limits the same for the x and y axes both, and make sure to draw a diagonal line. Make a mental note of how mitigating multicollinearity affected predictive ability in this particular instance.

```
model = lm(mass.test~.,data=pred.test)
plot(model)
```

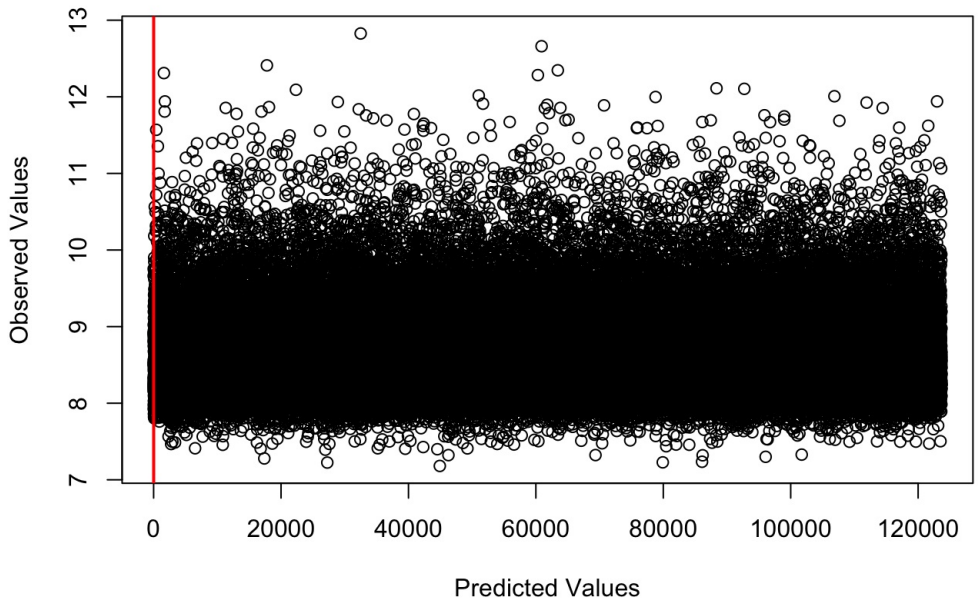

Residuals vs Fitted
Residuals
Fitted values
lm(mass.test ~ .)


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(mass.test ~ .)


Scale-Location
√|Standardized residuals|
Fitted values




```
plot(predict(model),
     pred.test$mass.test,
     xlab = "Predicted Values",
     ylab = "Observed Values")
abline(a = 0,
       b = 1,
       col = "red",
       lwd = 2)
```




```
sum.removed.lm.out= summary(removed.lm.out)
sum.lm.out= summary(lm.out)


mse1 = mean(sum.lm.out$residuals^2)
mse1
```

```
## [1] 0.01442085
```

```
mse2 = mean(sum.removed.lm.out$residuals^2)
mse2
```

```
## [1] 0.1478386
```