HANDOUT STATS 230- polynomial regression & multicollinearity

P.B. Matheson adapted from A.S. Wagaman

MLR: Polynomial Regression (focus on quadratic and complete second-order models) 3.4

```r
set.seed(230)
#next command creates a new variable x with a normal distribution
x <- rnorm(100, 5, 3) #100 is the sample size, mean=5 and sd=3
#next command creates a new variable y that is not normally distribute with a
#curved relationship with x
y <- 6*x - 2*x^2 + rnorm(100, 0, 7) #Y=6X-2X^2+epsilon
```

set up some data for x and y that has a quadratic relationsip

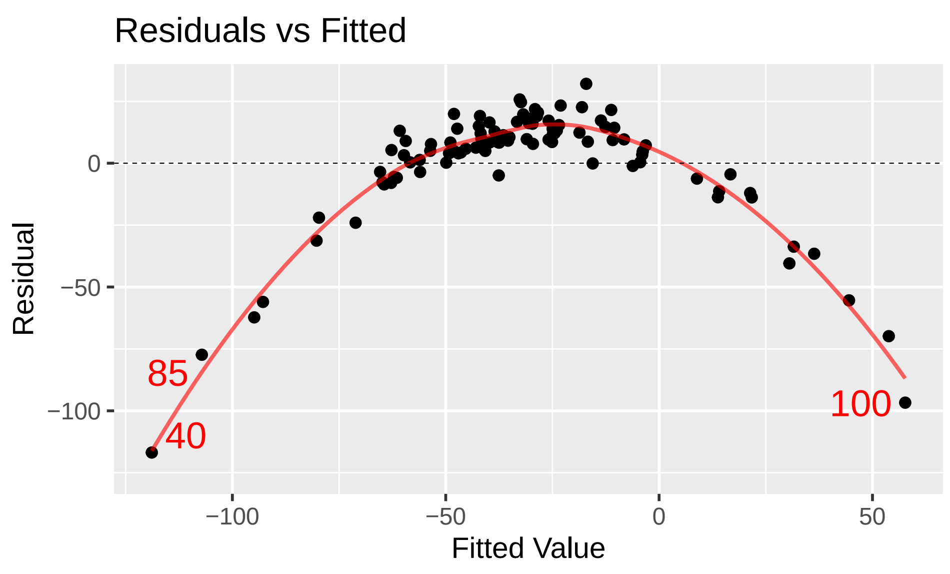
```r
#just SLR
mod1 <- lm(y ~ x)
msummary(mod1)
```

Try fitting a simple linear model

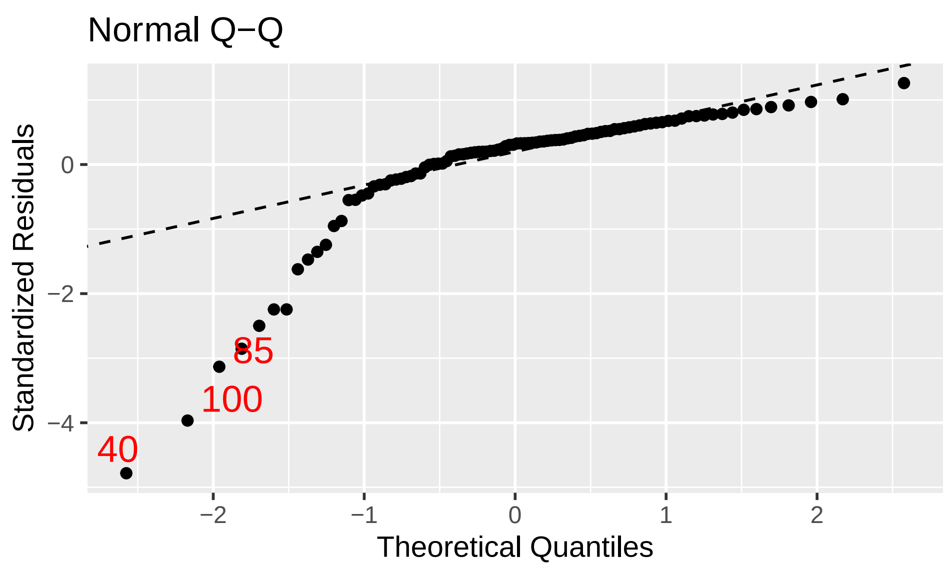
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.4018     5.1661    4.53 1.67e-05 ***
## x           -11.4781     0.9348  -12.28  < 2e-16 ***
##
## Residual standard error: 25.57 on 98 degrees of freedom
## Multiple R-squared:  0.6061, Adjusted R-squared:  0.602
## F-statistic: 150.8 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
mplot(mod1, which = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
mplot(mod1, which = 2)
```

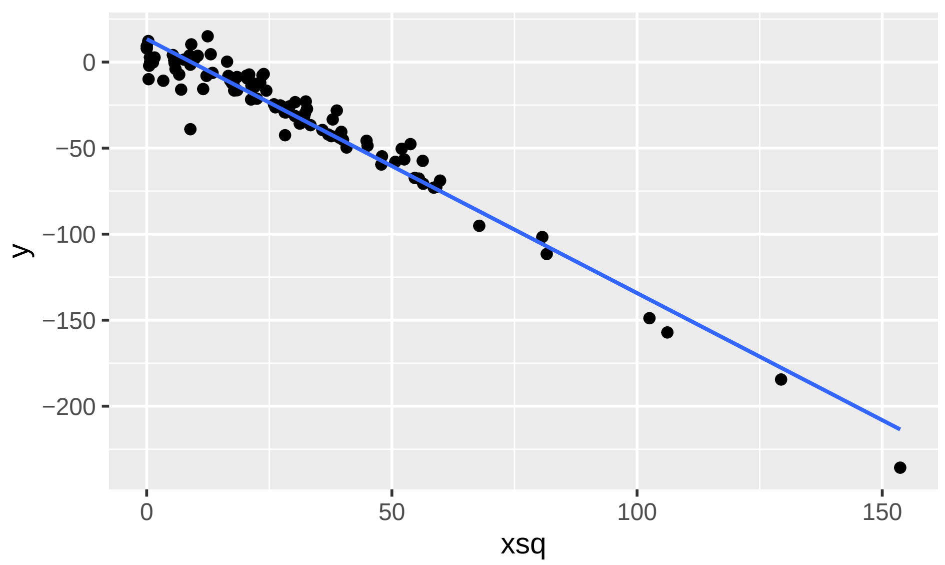


Does a simple linear model fit well?

Try a polynomial regression. Here we will try a quadratic regression with (xsquare) added as a term. Could also try a cubic regression with x cubed.

```
#create a new variable xsq which is x squared
xsq <- x^2

gf_point(y ~ xsq) %>%
   gf_lm()
```

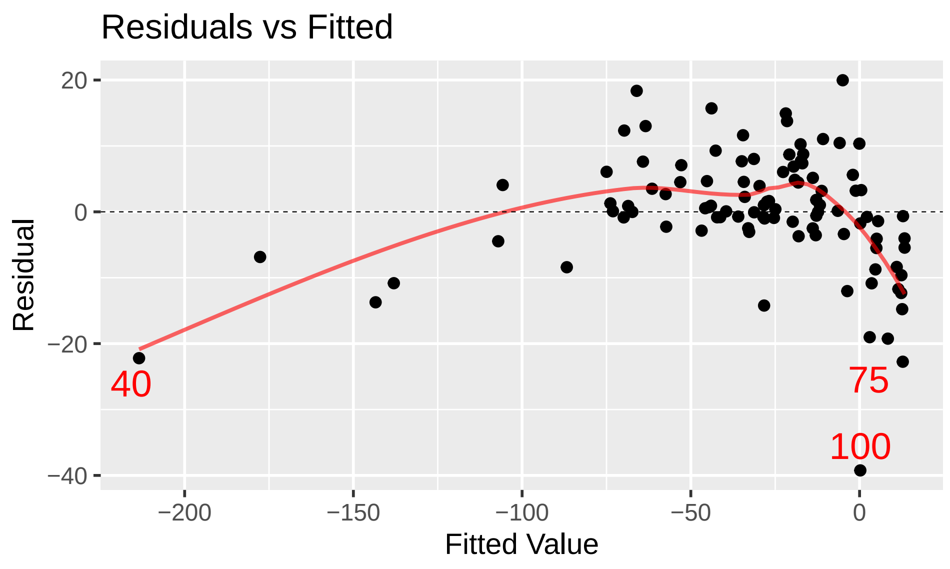


```
mod2 <- lm(y ~ xsq)
msummary(mod2)
```

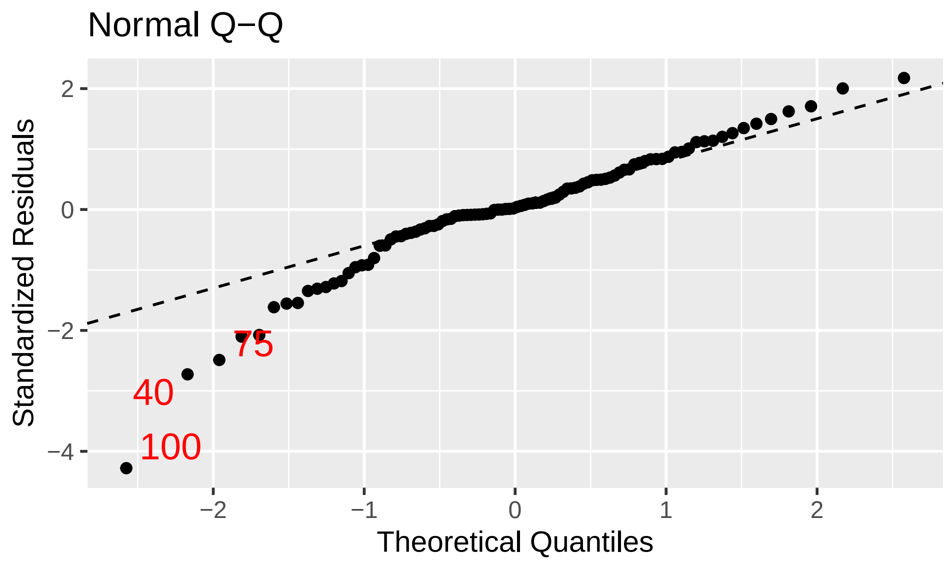
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.37922    1.40803   9.502 1.46e-15 ***
## xsq         -1.47655    0.03477 -42.472  < 2e-16 ***
##
## Residual standard error: 9.248 on 98 degrees of freedom
## Multiple R-squared:  0.9485, Adjusted R-squared:  0.9479
## F-statistic:  1804 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
mplot(mod2, which = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
mplot(mod2, which = 2)
```



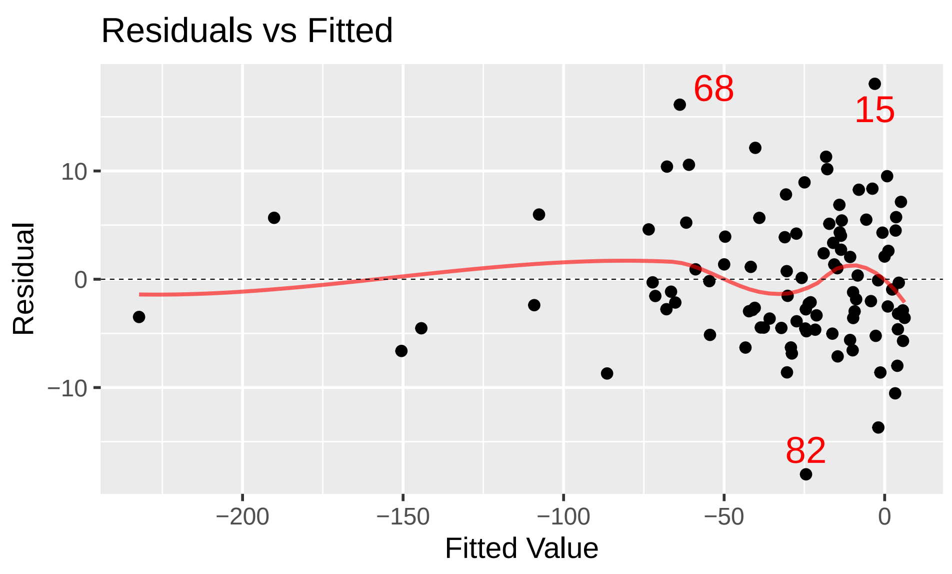
This model seems to improve the fit, but we still see some pattern in the residuals. We might need to keep the linear term (x) and add the quadratic re-expression (xsq).

```
mod3 <- lm(y ~ x + I(x^2)) #quadratic model
msummary(mod3)
```

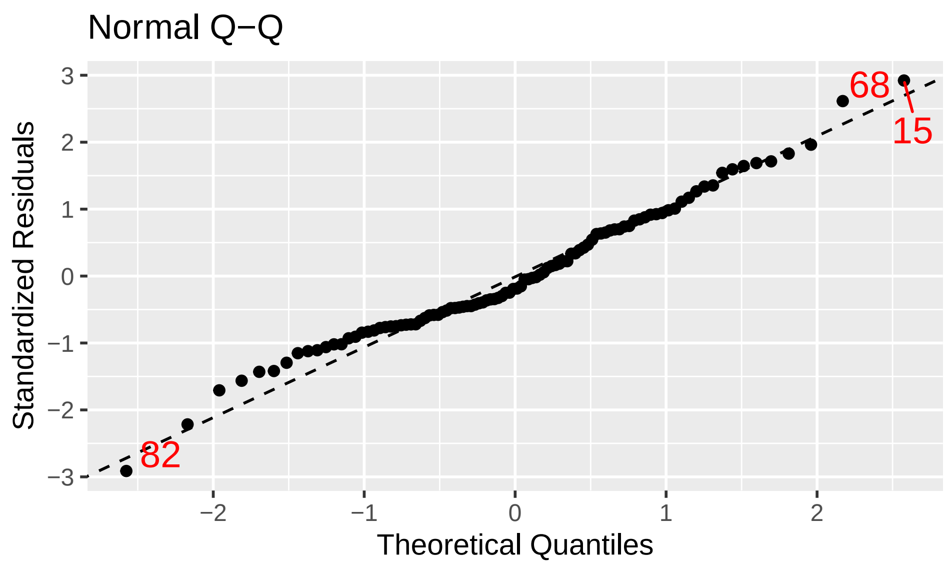
```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.68344    1.36305   1.969   0.0518 .
## x            5.25703    0.48143  10.920   <2e-16 ***
## I(x^2)      -1.95290    0.04951 -39.448   <2e-16 ***
##
## Residual standard error: 6.226 on 97 degrees of freedom
## Multiple R-squared:  0.9769, Adjusted R-squared:  0.9764
## F-statistic:  2050 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
mplot(mod3, which = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
mplot(mod3, which = 2)
```



IMPORTANT NOTE FOR POWERS: R will not understand xˆ2 in the lm command as xˆ2 unless you put I() around it. The same goes for other powers of either variable. You could also re-express like we did above, add the new variable to your data set (here we could use xsq we already created).

Finally, we can see what a cubic model would look like.

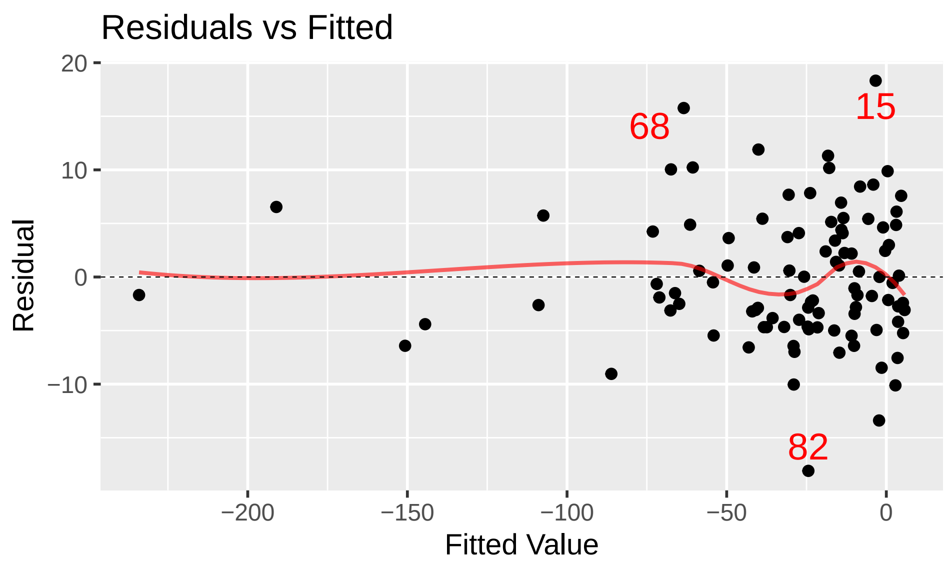
```
mod4 <- lm(y ~ x + I(x^2) + I(x^3)) #cubic model
msummary(mod4)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.35911    1.47193   1.603    0.112
## x             5.01809    0.62778   7.993 2.93e-12 ***
## I(x^2)       -1.85633    0.16950 -10.952  < 2e-16 ***
## I(x^3)       -0.00701    0.01176  -0.596    0.553
```

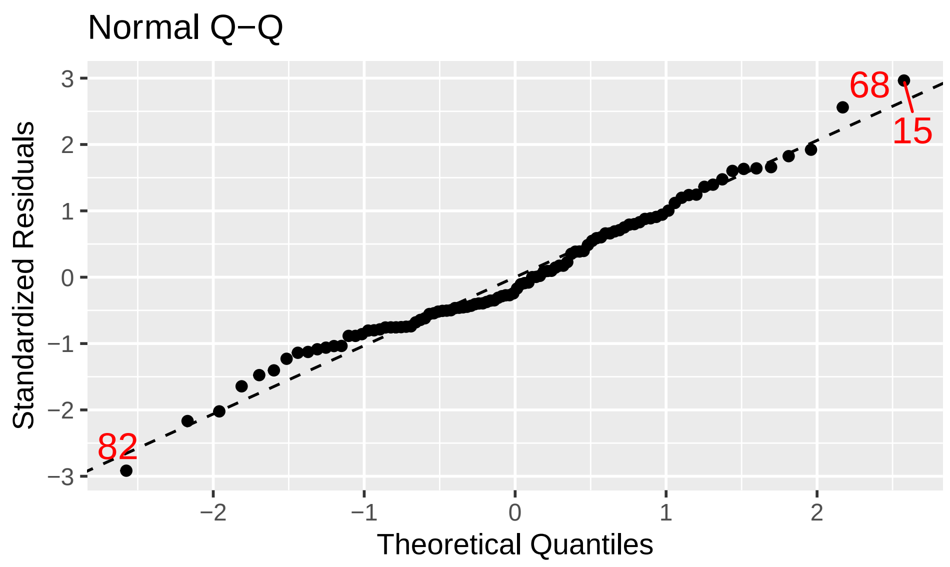
```
##
## Residual standard error: 6.246 on 96 degrees of freedom
## Multiple R-squared:  0.977,  Adjusted R-squared:  0.9763
## F-statistic:  1358 on 3 and 96 DF,  p-value: < 2.2e-16
```

```
mplot(mod4, which = 1)
```

```
## 'geom_smooth()' using formula 'y ~ x'
```



```
mplot(mod4, which = 2)
```



The cubic model doesn't seem to give us anything useful, although, we can see the cubed term has a non-significant slope, suggesting we can remove it from the model with the quadratic and linear terms left in.

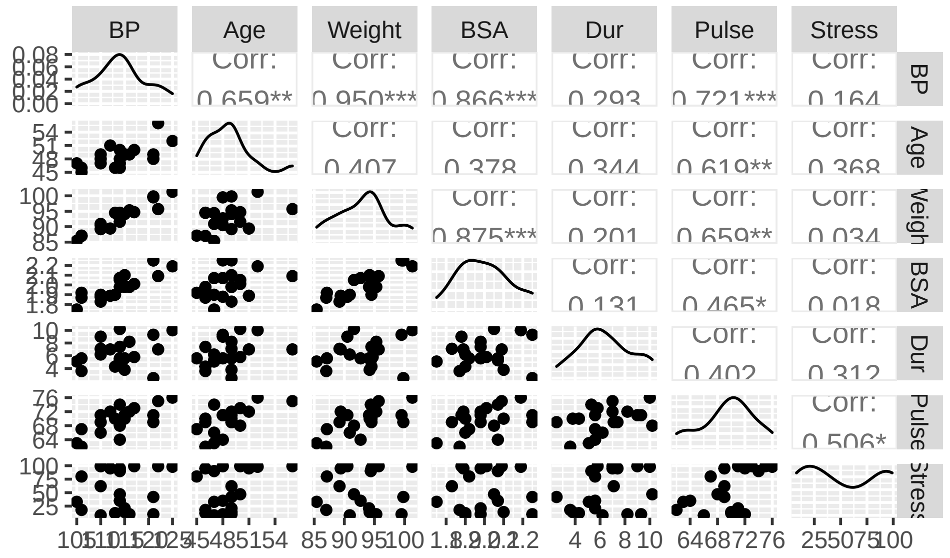
MLR: Multicollinearity and VIFs (3.5)

Researchers observed the following data on 20 individuals with high blood pressure:

- blood pressure (BP, in mm Hg)
- age (Age, in years)
- weight ($Weight$, in kg)
- body surface area (BSA, in m^2)
- duration of hypertension (Dur, in years)
- basal pulse ($Pulse$, in beats per minute)
- stress index ($Stress$)

Our goal is to build a model for blood pressure as a function of (some subset) of the other variables. In this case all of our variables are quantitative, so we can get a quick look at their relationships using the *ggpairs* command.

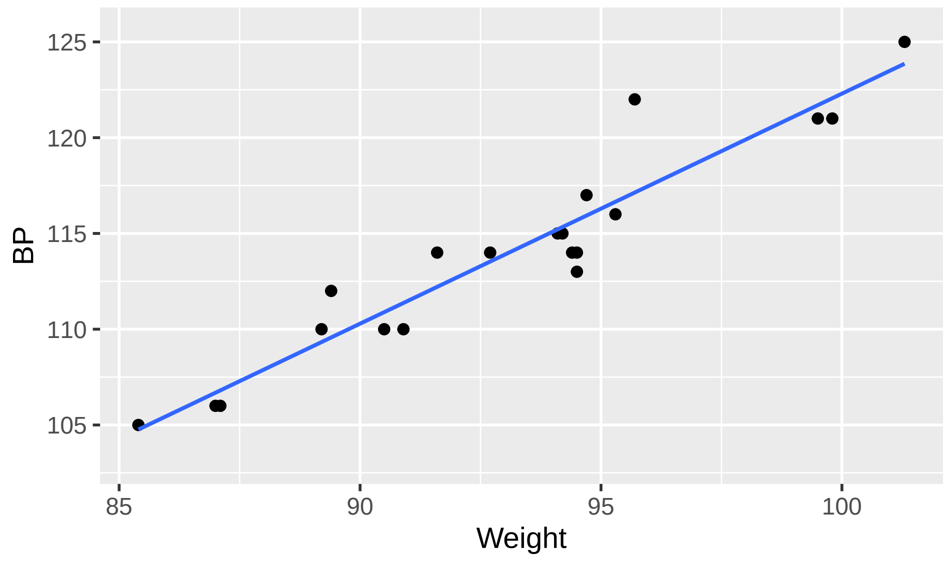
```r
BP <- read.csv("https://pmatheson.people.amherst.edu/stat230/bloodpress.csv")
ggpairs(BP)
```



What do you see in these scatterplots? Which of the variables are most highly correlated with BP?

A First Model $Weight$ seems to be highly correlated with BP, so as a first step, we should understand how well a simple linear model for blood pressure as a function of weight works. Keep in mind that it makes sense that that there would be a strong link between a person's weight and their blood pressure. This is face validity.

```r
gf_point(BP ~ Weight, data = BP) %>%
  gf_lm()
```



```
fm1 <- lm(BP ~ Weight, data=BP)
msummary(fm1)
```

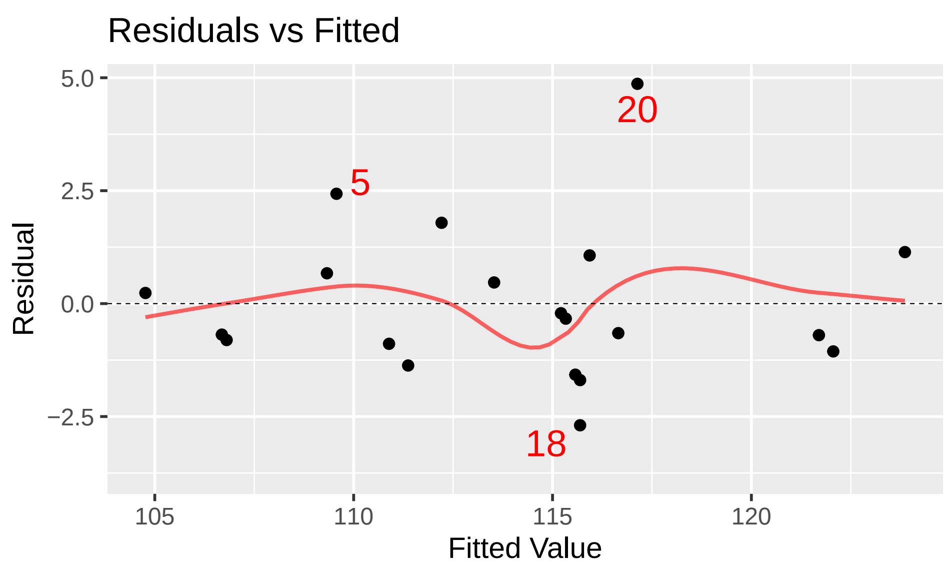
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.20531    8.66333   0.255    0.802
## Weight       1.20093    0.09297  12.917 1.53e-10 ***
##
## Residual standard error: 1.74 on 18 degrees of freedom
## Multiple R-squared:  0.9026, Adjusted R-squared:  0.8972
## F-statistic: 166.9 on 1 and 18 DF,  p-value: 1.528e-10
```

How well does the SLR model for blood pressure work?

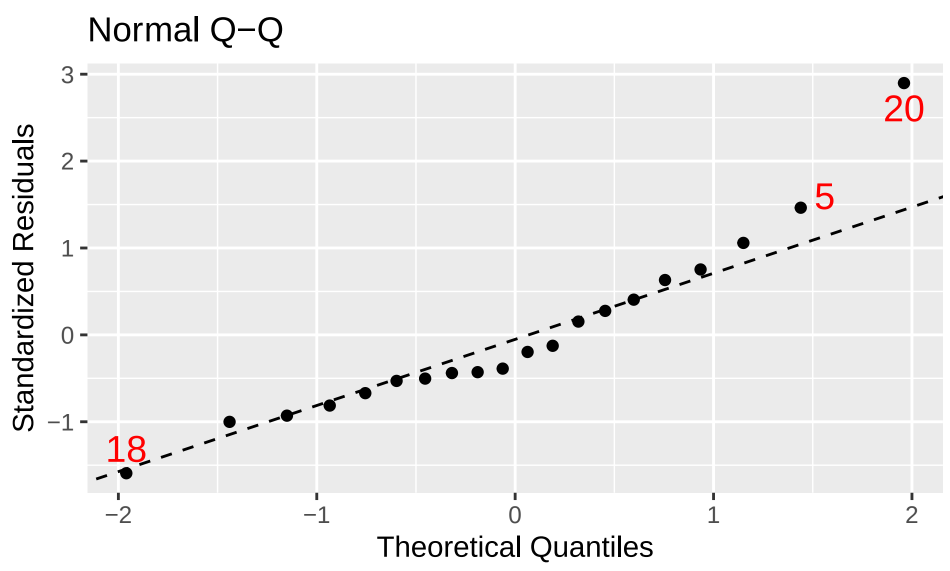
Check the assumptions for SLR. Are they met?

```
mplot(fm1, which = 1)
```

```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



```r
mplot(fm1, which = 2)
```



The Kitchen Sink The SLR model for blood pressure is pretty effective, but there is one person who generates a large residual. We want to use the additional data that we have to build a better model for blood pressure.

```r
BP[20,]
```

```
##     BP Age Weight  BSA Dur Pulse Stress
## 20 122  56   95.7 2.09   7    75     99
```

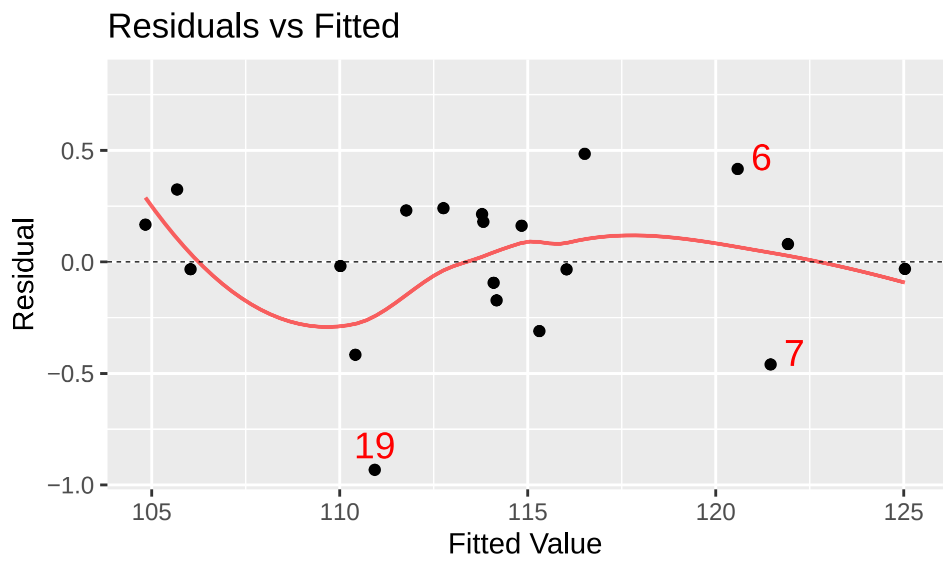
Without any intuition, one way to proceed is to simply throw all of the variables into our regression model. In a sense, we are throwing everything but the kitchen sink into the model.

```
# Using the . in the formula interface includes all possible predictor variables in the data frame (exc
fm.full <- lm(BP ~ ., data = BP)
msummary(fm.full)
```

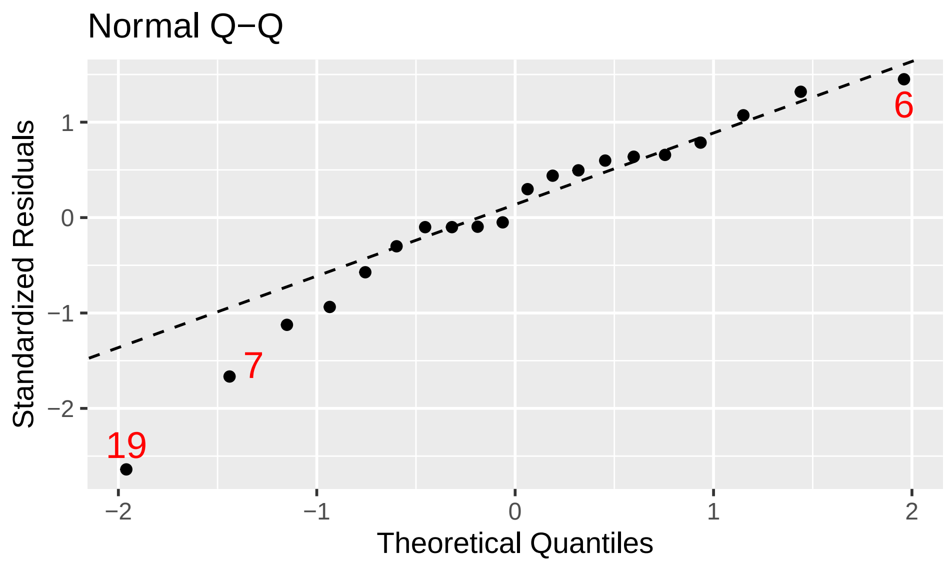
```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -12.870476   2.556650  -5.034 0.000229 ***
## Age           0.703259   0.049606  14.177 2.76e-09 ***
## Weight        0.969920   0.063108  15.369 1.02e-09 ***
## BSA           3.776491   1.580151   2.390 0.032694 *
## Dur           0.068383   0.048441   1.412 0.181534
## Pulse        -0.084485   0.051609  -1.637 0.125594
## Stress        0.005572   0.003412   1.633 0.126491
##
## Residual standard error: 0.4072 on 13 degrees of freedom
## Multiple R-squared:  0.9962, Adjusted R-squared:  0.9944
## F-statistic: 560.6 on 6 and 13 DF,  p-value: 6.395e-15
```

```
mplot(fm.full, which = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
mplot(fm.full, which = 2)
```



We have a very high R^2 with the kitchen sink model. However, there are still some issues with the normality of the residuals. Moreover, the coefficient for $Pulse$ is negative, suggesting that people with higher pulse rates have lower blood pressure. Does this make sense? Three of the coefficients in the model are close to zero and not significant at the 10% level.

Checking for Multicollinearity - are the predictors correlated with each other? The easiest way to check for multicollinearity is by computing the Variance Inflation Factor (VIF) for each explanatory variable. This function is in the *car package*, so it is called here from the package. You could alternatively load the package then just call *vif*, but since we only need the package for THIS function, and *car* can cause issues with other functions (conflicts) this seems easier.

```
# Note that this computation requires the "car" package! be sure you call it.
car::vif(fm.full) #calls for vif function from car package without loading rest of car
```

```
##      Age   Weight      BSA      Dur    Pulse   Stress
## 1.762807 8.417035 5.328751 1.237309 4.413575 1.834845
```

VIFs higher than 5, such as those for $Weight$, BSA, and $Pulse$ (4.4 is still high compared to the rest) indicate that those variables are highly associated with the other explanatory variables or some linear combination of the other explanatory variables. To see this, we can manually compute the VIF for $Weight$, by regressing it against the other explanatory variables. This lets us look for the linear combination of other variables that $Weight$ is related to.

```
# here we are predicting weight (which was one of the original predictors from the other predictors)
fm.vif <- lm(Weight ~ Age + BSA + Dur + Pulse + Stress, data = BP)
msummary(fm.vif)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.674438   9.464742   2.079  0.05651 .
## Age         -0.144643   0.206491  -0.700  0.49510
## BSA         21.421654   3.464586   6.183 2.38e-05 ***
## Dur          0.008696   0.205134   0.042  0.96678
## Pulse        0.557697   0.159853   3.489  0.00361 **
## Stress      -0.022997   0.013079  -1.758  0.10052
##
## Residual standard error: 1.725 on 14 degrees of freedom
## Multiple R-squared:  0.8812, Adjusted R-squared:  0.8388
## F-statistic: 20.77 on 5 and 14 DF,  p-value: 5.046e-06
```

```
#the next command demonstrates how the VIF from above was calculated.  This is only for demonstration.
1 / (1 - rsquared(fm.vif))
```

```
## [1] 8.417035
```

Note that if this kind of regression works well, then R^2 will be high, and the VIF will become very large. A similar computation can be performed for each of the explanatory variables if you would like to see it.

Another window into the entangled web of our explanatory variables is to examine the correlation between every pair. This helps you identify pairwise correlations but not more complicated relationships that the linear combinations might result in.

```
# correlation between every pair of variables in the data set; works if all are quantitative
# otherwise, use select to remove categorical variables
cor(BP)
```

```
##               BP       Age     Weight        BSA       Dur     Pulse     Stress
## BP     1.0000000 0.6590930 0.95006765 0.86587887 0.2928336 0.7214132 0.16390139
## Age    0.6590930 1.0000000 0.40734926 0.37845460 0.3437921 0.6187643 0.36822369
## Weight 0.9500677 0.4073493 1.00000000 0.87530481 0.2006496 0.6593399 0.03435475
## BSA    0.8658789 0.3784546 0.87530481 1.00000000 0.1305400 0.4648188 0.01844634
## Dur    0.2928336 0.3437921 0.20064959 0.13054001 1.0000000 0.4015144 0.31163982
## Pulse  0.7214132 0.6187643 0.65933987 0.46481881 0.4015144 1.0000000 0.50631008
## Stress 0.1639014 0.3682237 0.03435475 0.01844634 0.3116398 0.5063101 1.00000000
```

Note that BSA and $Weight$ are strongly correlated. Can you think of why this might be?
Moreover, $Pulse$ is fairly correlated with Age, $Weight$, and $Stress$.
Since we know that $Weight$ is strongly correlated with BP to begin with, we'll want to keep this variable in our model. You DON'T have to remove the variable with the highest VIF!

Let's try removing BSA and $Pulse$ though.

```
fm2 <- lm(BP ~ Weight +  Age + Dur + Stress, data = BP)
msummary(fm2)
```

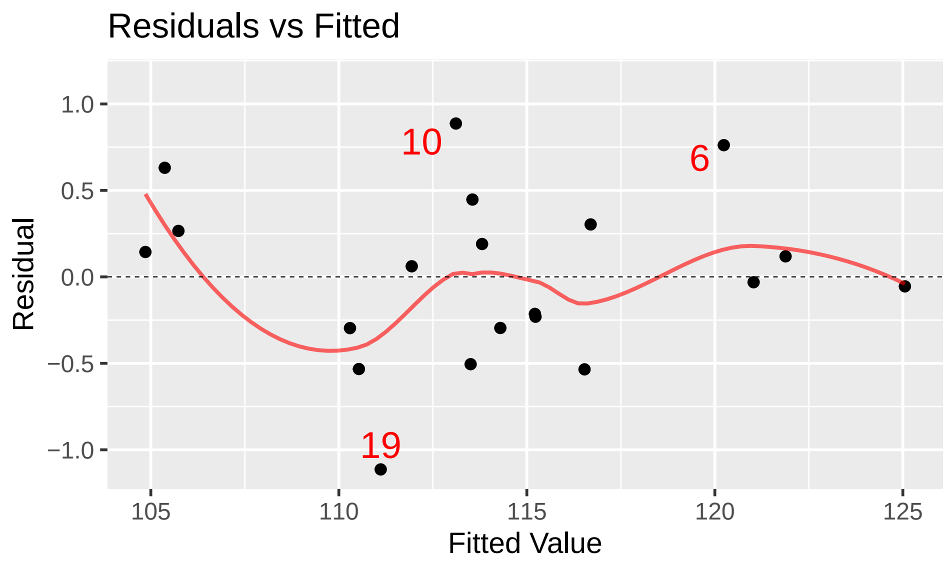
```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -15.869829   3.195296  -4.967 0.000169 ***
## Weight        1.034128   0.032672  31.652 3.76e-15 ***
## Age           0.683741   0.061195  11.173 1.14e-08 ***
## Dur           0.039889   0.064486   0.619 0.545485
## Stress        0.002184   0.003794   0.576 0.573304
##
## Residual standard error: 0.5505 on 15 degrees of freedom
## Multiple R-squared:  0.9919, Adjusted R-squared:  0.9897
## F-statistic: 458.3 on 4 and 15 DF,  p-value: 1.764e-15
```

```
car::vif(fm2)
```

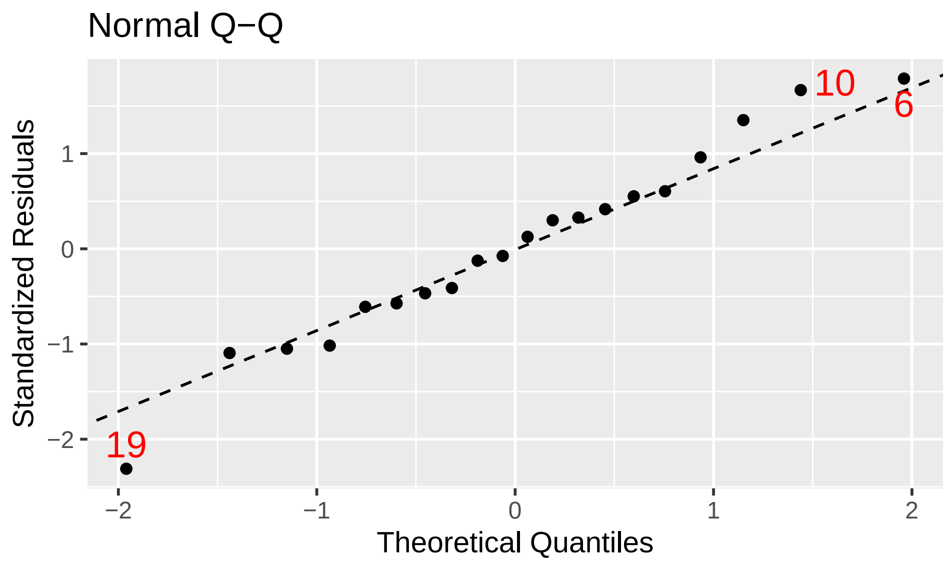
```
##   Weight      Age      Dur   Stress
## 1.234653 1.468245 1.200060 1.241117
```

```
mplot(fm2, which = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
mplot(fm2, which = 2)
```



This simpler model explains almost as much of the variation as the full model, but with two fewer explanatory variables. Moreover, the residuals are closer to being normal, and the high VIFs have been eliminated. This model should be preferred to the full model.

Still, two of the variables in the reduced model (*Dur* and *Stress*), do not seem to be contributing much to the model (why do I make that claim?). Let's see what happens if we eliminate them.

```
fm3 <- lm(BP ~ Weight +  Age, data = BP)
msummary(fm3)
```

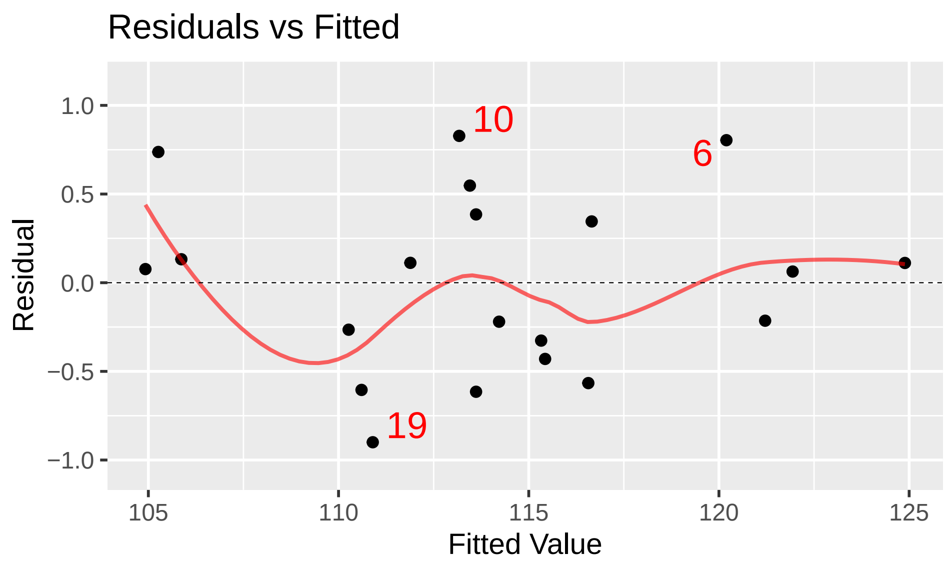
```
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -16.57937    3.00746  -5.513 3.80e-05 ***
## Weight        1.03296    0.03116  33.154  < 2e-16 ***
## Age           0.70825    0.05351  13.235 2.22e-10 ***
##
## Residual standard error: 0.5327 on 17 degrees of freedom
## Multiple R-squared:  0.9914, Adjusted R-squared:  0.9904
## F-statistic: 978.2 on 2 and 17 DF,  p-value: < 2.2e-16
```

```
car::vif(fm3)
```

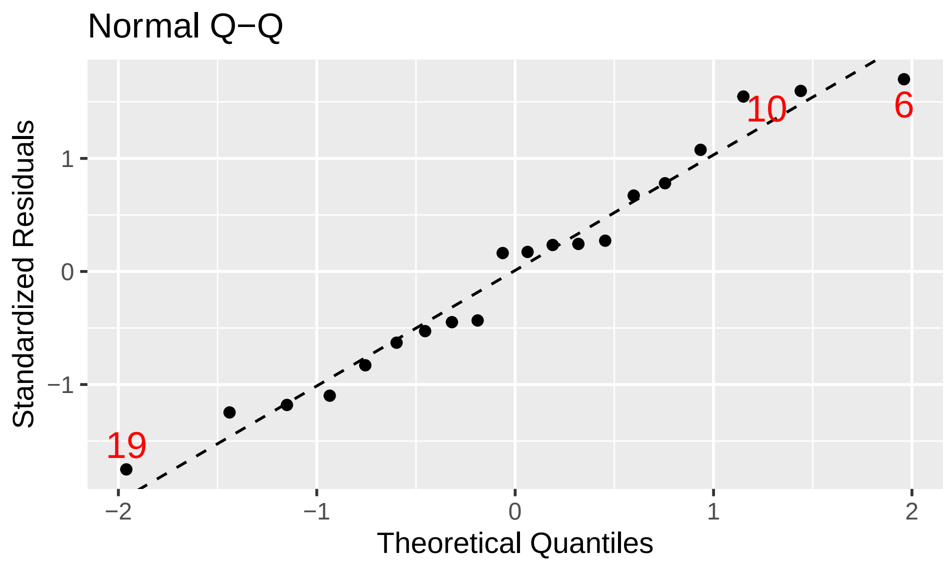
```
##   Weight      Age
## 1.198945 1.198945
```

```
mplot(fm3, which = 1)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
mplot(fm3, which = 2)
```



Which is the best model?

This model is again nearly as effective as the full model, with no evidence of multicollinearity, and more normally distributed residuals. What we have lost in R^2, we have more than made up for in simplicity and conformity to our assumptions.