Unofficial Kahn Academy R Supplement: Scatterplots, Correlations, and Regressions

The correlation command in R is, once again, quite simple. To look at the correlation matrix (or table of correlations) for the variables in the **mtcars** dataset we simply put the name of the dataset in the *cor* command.

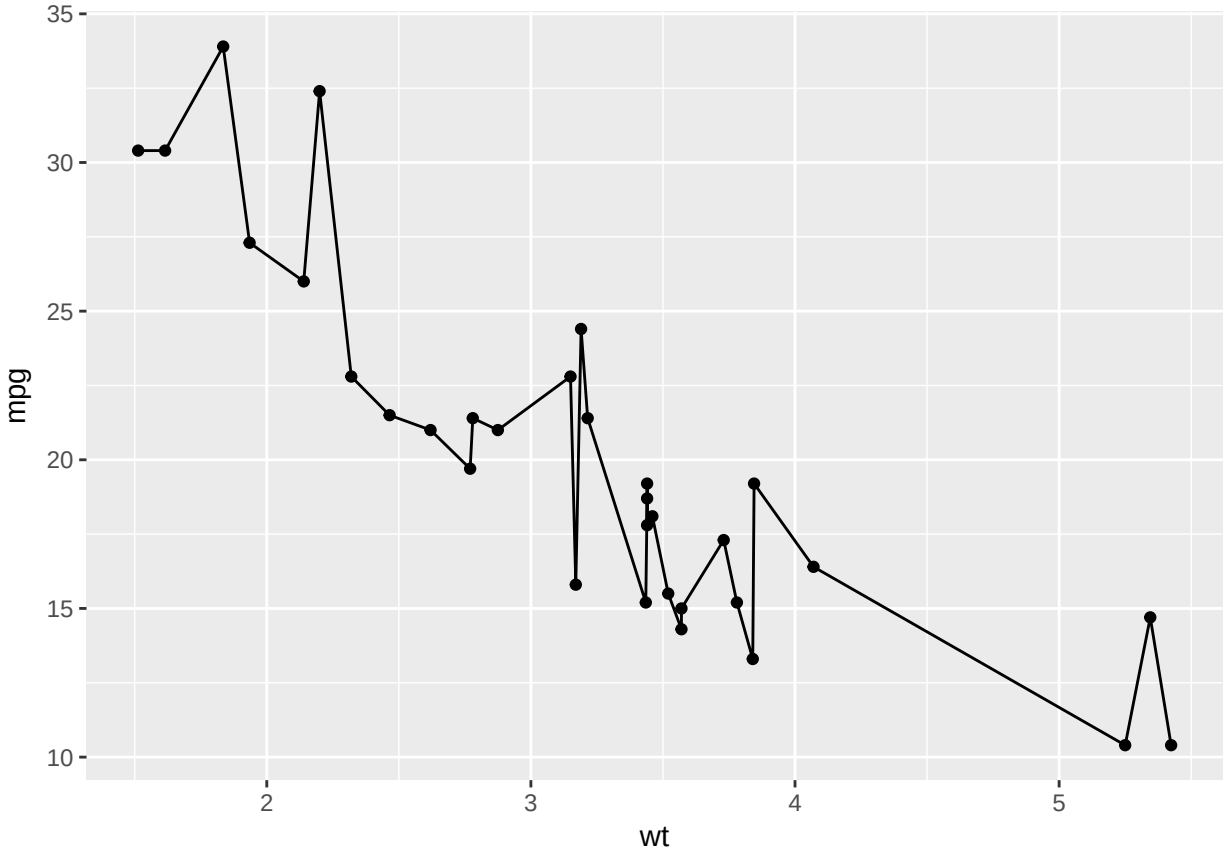
```
cor(mtcars)
```

```
##             mpg        cyl       disp         hp        drat         wt
## mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
## cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
## disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
## hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
## drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
## wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
## qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
## vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
## am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
## gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
## carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
##             qsec         vs          am       gear        carb
## mpg   0.41868403  0.6640389  0.59983243  0.4802848 -0.55092507
## cyl  -0.59124207 -0.8108118 -0.52260705 -0.4926866  0.52698829
## disp -0.43369788 -0.7104159 -0.59122704 -0.5555692  0.39497686
## hp   -0.70822339 -0.7230967 -0.24320426 -0.1257043  0.74981247
## drat  0.09120476  0.4402785  0.71271113  0.6996101 -0.09078980
## wt   -0.17471588 -0.5549157 -0.69249526 -0.5832870  0.42760594
## qsec  1.00000000  0.7445354 -0.22986086 -0.2126822 -0.65624923
## vs    0.74453544  1.0000000  0.16834512  0.2060233 -0.56960714
## am   -0.22986086  0.1683451  1.00000000  0.7940588  0.05753435
## gear -0.21268223  0.2060233  0.79405876  1.0000000  0.27407284
## carb -0.65624923 -0.5696071  0.05753435  0.2740728  1.00000000
```

AYCS, weight and mpg has a negative correlation of -0.8676594. This is quite high (so there is a strong relationship), and it is negative (bigger cars= lower mpg). After a while you will get a sense of how big correlations are just by looking at the scatterplots. This website is a fun, gamified way of learning how to guess correlation coefficients based on the scatterplot: http://guessthecorrelation.com

Making a scatterplot in R with ggplot2 is very similar to making a line chart with ggplot2. Matter of fact, it's basically the same code minus the *geom_line* subcommand. For example, if we do want to make a scatterplot of the same weight/mpg relationship we did a line chart for before, we would type:

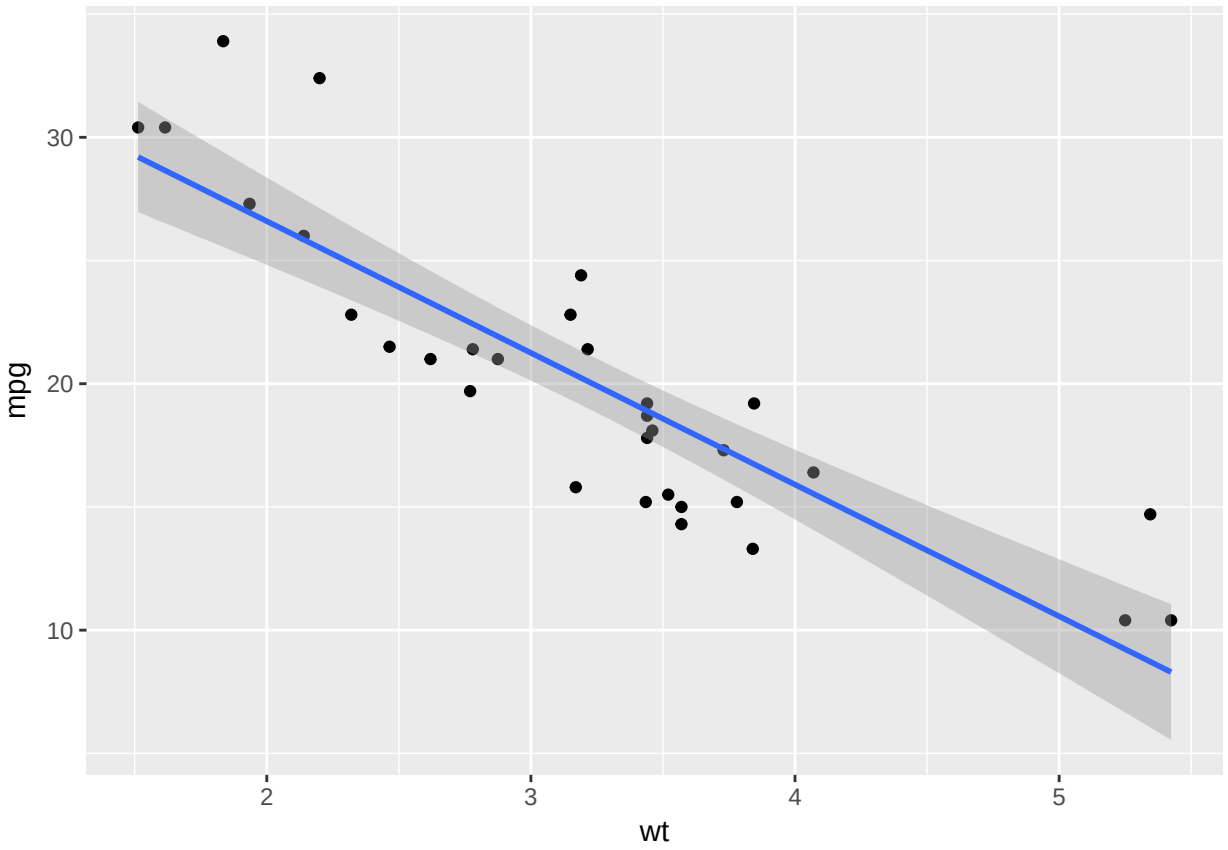
```
#install.package("ggplot2")
library(ggplot2)
ggplot(data=mtcars, aes(x=wt, y=mpg)) +
  geom_line()+
  geom_point()
```



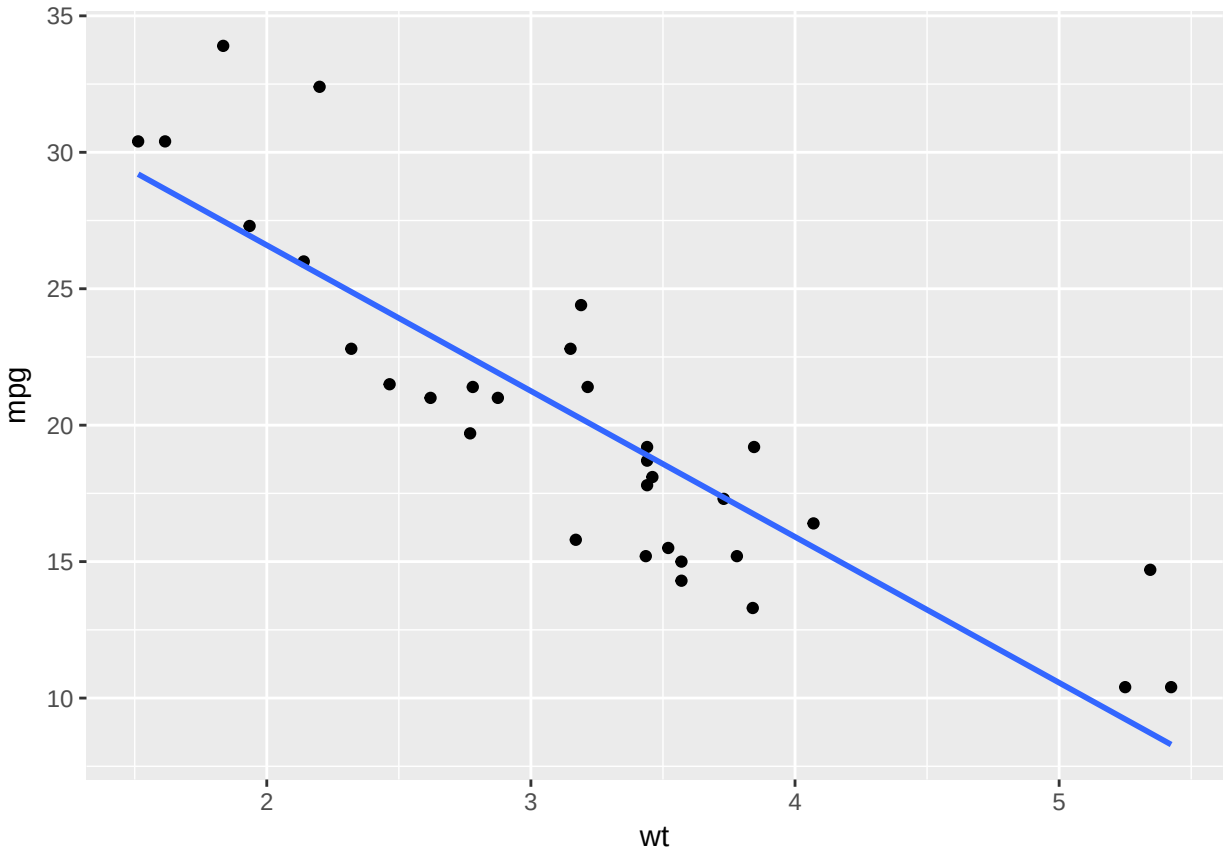
Because there's a *geom_point* subcommand and no *geom_line* subcommand like there was with our earlier line chart, R assumes that this is a dot-based scatterplot. It is worth noting that the graph that is generated changes its height and width depending on how big or small you make the box with your cursor (at least in RStudio). If you create a pdf of this graph when the box is really short, then the pdf image will be really short.

Once again, we can add additional chart parts to our basic scatterplot. For example, we can fit a line through (more or less) the middle of the points (KA talks about the "least squares" method of fitting, that is the method used here if you make the *method=lm*) by adding a *geom_smooth(method=lm)* subcommand so that we do not have to rely on eyeballing the direction the dots are headed.

```
ggplot(data=mtcars, aes(x=wt, y=mpg)) +
  geom_point()+
  geom_smooth(method=lm)
```




```
#By default the confidence intervals (we'll get to those later) are shown, so to remove #them  we add s
ggplot(data=mtcars, aes(x=wt, y=mpg)) +
  geom_point()+
  geom_smooth(method=lm, se=FALSE)
```



Now that we can see the "least squares" line, we can come up with an algebraic equation that describes it; you can do this by hand by calculating the slope and estimating where the line crosses over the Y-axis, but once again the whole reason we are using R is so that we can avoid having to do this all by ourselves, since humans are messy and we make simple mistakes that can have significant consequences. To create the least-squares line we use the *lm()* command. The *lm* command stands for least-squares model, and it fits the least squares line as described in KA. Once again, if we want to see the linear regression model for weight predicting miles per gallon, we simply put the dependent variable (in this case miles per gallon), then a ~ sign, then any independent (predictor) variables you need. In this case we only have one: weight.

```
lm(mpg~wt, data= mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

AYCS, the (intercept) number is where the line crosses the Y-axis. The -5.3 under **wt** is the slope of the line, which in this case is negative, hence a negative relationship. Therefore the equation that describes the line is Y= -5.344X + 37.285. If you go back up to the scatterplot above, you will see that this equation approximates the line drawn through the scatterplot, which crosses the Y-axis at about 37. (It may not look like that because the axis ticks stop at the floor of 1, but if you continue the trendline to 0 it crosses at about 37).

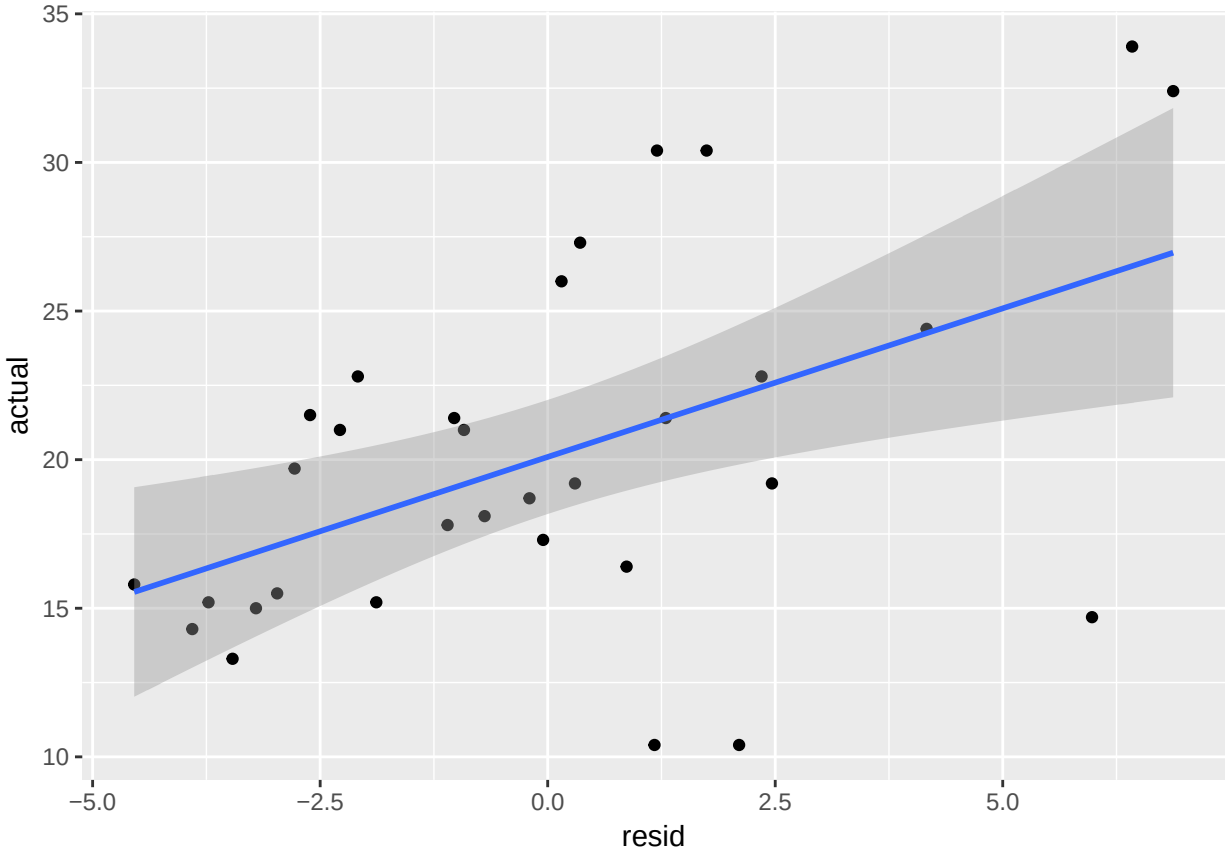
If we want to calculate the residuals, or how far from the regression line each point is, we have to make the linear model as an object, then we use the *resid* command on the linear model that we have stored as an object. This creates a series of numbers that represent the residuals for each observation. To compare the

residuals to their actual values, we need to get the residuals and actual values in the same dataset. We can then easily plot the residuals against their values. In the code below, we create a dataset of only the mpg variable, then we create a new variable in that dataset that holds the residuals.

```
#Here we assign the model itself as an object (named MTCARSModel)
MTCARSModel<-lm(mpg~wt, data=mtcars)
summary(MTCARSModel)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```
#Here we create a new data frame (i.e. dataset) of only the mpg variable
#To do that with use the as.data.frame command
mtcars_mpg<-as.data.frame(mtcars$mpg)
#Now we change the name of the variable to make it easier.
mtcars_mpg$actual=mtcars_mpg$`mtcars$mpg`
# Below we are creating a new variable in the formerly one-variable mtcars_mpg
#dataset (mtcars_mpg$resid), the new variable is the residuals from the linear model above.
mtcars_mpg$resid = resid(MTCARSModel)
#Now we are in a position to do a scatterplot with ggplot, since we have both
#variables in the same dataset.
ggplot(data=mtcars_mpg, aes(x=resid, y=actual)) +
  geom_point()+
  geom_smooth(method=lm)
```



Why would we want to plot the residuals? To make a long story short, if there is a strong trend one way or the other, then it suggests that the relationship is not linear (in other words, the line should be curving to really fit the data well), and we have to try a non-linear approach. KA discusses non-linear modelling further down the road, but hopefully at this point you understand the concept of residuals and the role they play in least squares.

Finally, if we want to look at other characteristics of the model such as R-squared, we simply use the *summary* command on the model.

```
summary(MTCARSModel)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
```

```
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

Here we get the R-squared, the F-statistic, and a number of other goodies. If you want to create a publication-quality table out of your regression, probably the best option would be to look into the *stargazer* package.