Session7 Linear Models and Regression

I. Ali, M. Wells, A. Gamble

2023-03-06

# Introduction to Linear Regression

In session 6, we started our deep dive into the “when” and the “how” to apply statistical tests in your research. We offered examples of common statistical tests such as t-testing, ANOVA, and multiple testing. We also have gotten a chance to practice using data visualization tools which we can use in conjunction with statistical tests to better understand our data.

In this session we will expand further on how to establish and understand relationships between variables and trends. We will focus on the use of linear regression, the statistical tool used to construct linear models, which can teach us how a response occurs from one or more predicting variables.

By the end of this session you will be able to:

* Calculate a correlation coefficient using the cor() command
* Run a correlation test using cor.test() and interpret the results
* Create a linear model with lm() and interpret the results

## Loading Packages

For our lesson, we will need to do some data wrangling and data visualization in conjunction with our statistical tests. We will use the base stats package for most of the statistics we will be running

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.1 ✔ purrr 1.0.1  
## ✔ tibble 3.1.8 ✔ dplyr 1.1.0  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.3 ✔ forcats 0.5.2  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

# Background

We are going to be working with another fictional data set, similar to the one we used for data visualization in Session 4. This data set contains information on gene transcription (MeanCounts, representing measurements of mRNA counts per million reads) and two metrics of protein expression (WesternBlotSignal, representing protein signal from quantified western blots), and protein expression measured by Flow cytometry (FlowCytGFPSignal, representing the fluorescence intensity of a GFP-bound. protein).

Transcription levels are classically easier to measure (measured by RNA sequencing) than protein expression. Western blotting is notoriously difficult to use for quantifying protein expression. In contrast, using a GFP bound protein and quantifying the fluorescence intensity using a flow cytometer can be a more direct approach to quantifying the amount of protein in the cell. For this example, we will assume that translation is not a rate limiting step on gene expression for any of these genes. Here we will test the strength of the correlation between transcription levels and the different methods of measuring protein expression.

## Importing Data

# Using the read.csv() command import the GeneExpressionData.csv file  
  
ProteinExpressionData <- read.csv("../Datasets/ProteinExpressionData.csv", header = TRUE, sep = ",")

# Correlations: Analyzing data with lines.

A correlation is a statistical measure of the extent to which two or more variables move in coordination with each other. Correlations can describe simple reltionships among data. They are quantified by a value called the correlation coefficient (R) which ranges from -1 to 1. See the table below describing how the correlation coefficient values are typically interpreted.

| Correlation coefficient (R) | Correlation Type | Strength |
| --- | --- | --- |
| Greater than 0.7 | Positive Correlation | Strong |
| Between 0.4 and 0.7 | Positive Correlation | Moderate |
| Between 0 and 0.4 | Positive Correlation | Weak, or Negligible |
| 0 | No correlation | None |
| Between 0 and -0.4 | Negative Correlation | Weak, or Negligible |
| Between -0.4 and -0.7 | Negative Correlation | Moderate |
| Less than -0.7 | Negative Correlation | Strong |

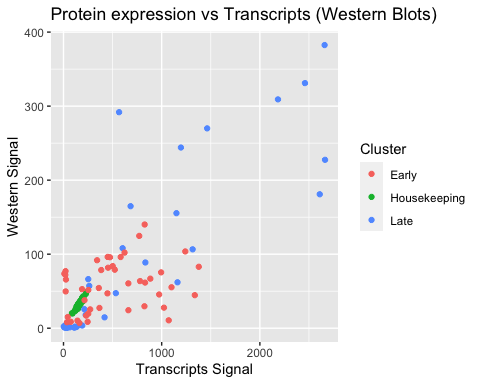
Most correlation analysis are depicted on scatter plots or line plots. Remember we used geom\_point() to create scatter plots and geom\_line() or geom\_smooth() to create line plots. The closer the points on your plot fit to a straight line, the closer the correlation coefficient (R) gets to 1. If the slope of the line is positive the R will be positive, and if the slope is negative R will be negative. Often one of the first steps to assessing a correlation is creating a scatterplot of the data to see if the data creates a line.

Of course, when we are working with biological samples there is a lot of inherent variability. Remember, just because variables correlate with each other, doesn’t mean that one may cause the other. Follow up tests need to be done to validate whether variables correlate to one another, or if one causes the other to change. Let’s start by creating some plots to view distribution of our sample genes.

## Visualizing data with a linear relationship

In our main example today we are exploring the relationship between measuring transcription expression and measuring protein expression. First, create a plot visualizing the relationship between transcript counts and western blot signal.

# Use ggplot to create a scatter plot comparing transcriptions and expression signal  
  
ggplot(ProteinExpressionData, aes(x = TranscriptCounts, y = WesternBlotSignal, color = Cluster)) +  
 geom\_point() +  
 labs(title = "Protein expression vs Transcripts (Western Blots)",  
 x = "Transcripts Signal",  
 y = "Western Signal")



We can see based on our plots that the protein expression as measured by Western Blots is quite noisy. It appears to have some sort of correlation, but the different gene clusters appear to correlate differently. Divide up the groups by cluster to get a better look at their relationships and assess the strength of the correlations for each group.

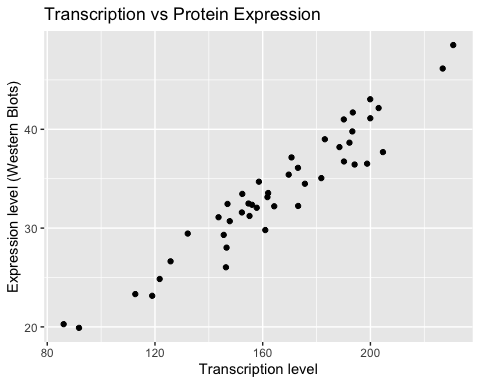
## Pearson Correlation

A pearson correlation test is both a descriptive statistic that describes characteristics of the data (like mean, median, standard deviation, etc.), but it can also be used as an inferential statistic which allows us to draw conclusions about the relationship of the variables (like a t-test or ANOVA). To calculate the correlation coefficient you can use the cor() function which typically needs three arguments described in bullets below.

* x = -A vector of data for an explanatory variable x (required)
* y = -A vector of data for an outcome variable y (required)
* method = -A method to construct the correlation (options include “pearson”, “spearman”, or “kendall”)

By default, cor() will assume that the first argument is vector of data to be plotted along the x axis and the second, the second argument a vector plotted along y. The third argument is an optional argument in case you want or need to specify the method for calculating the correlation, the default option is “pearson”.

# create a group for housekeeping genes  
Housekeeping <- ProteinExpressionData %>%  
 filter(Cluster == "Housekeeping")  
  
# plot the housekeeping genes.  
ggplot(Housekeeping, aes(x = TranscriptCounts, y = WesternBlotSignal)) +  
 geom\_point() +  
 labs(title = "Transcription vs Protein Expression",  
 x = "Transcription level",  
 y = "Expression level (Western Blots)")



# calcluate the correlation coefficient  
cor(x = Housekeeping$TranscriptCounts, y = Housekeeping$WesternBlotSignal, method = "pearson")

## [1] 0.9564108

Using our correlation test we see that the correlation coefficient is 0.956 and according to our correlation table above, this is a strong, positive correlation.

The cor() function is limited in that it only shows the correlation coefficient or the R value. A correlation test will tell allow us to test a hypothesis about the relationship between the two variables, use the cor.test() function with the same arguments.

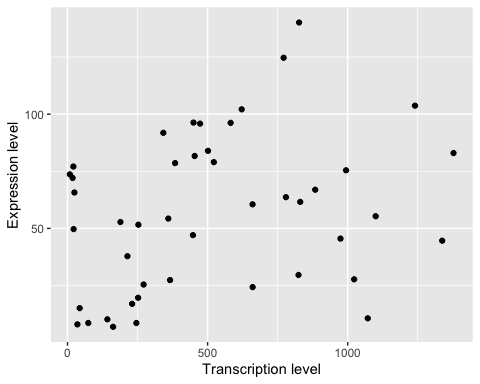
The pearson correlation test follows the hypotheses listed below:

* Null hypothesis: there is no linear relationship between the two variables.
* Alternative hypothesis: there is a linear relationship between the two variables.

For the Pearson test to be valid, the data must follow the following assumptions:

* Data for each variable are independent
* For small samples (n<30), the two variables should be normally distributed.
* Data must not have a curved linear relationship

Early <- ProteinExpressionData %>%  
 filter(Cluster == "Early")  
  
ggplot(Early, aes(x = TranscriptCounts, y = WesternBlotSignal)) +  
 geom\_point() +  
 labs(x = "Transcription level",  
 y = "Expression level")



cor.test(Early$TranscriptCounts, Early$WesternBlotSignal, method = "pearson")

##   
## Pearson's product-moment correlation  
##   
## data: Early$TranscriptCounts and Early$WesternBlotSignal  
## t = 1.9747, df = 43, p-value = 0.05474  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.005662685 0.536476692  
## sample estimates:  
## cor   
## 0.2883505

## Interpreting cor.test() output

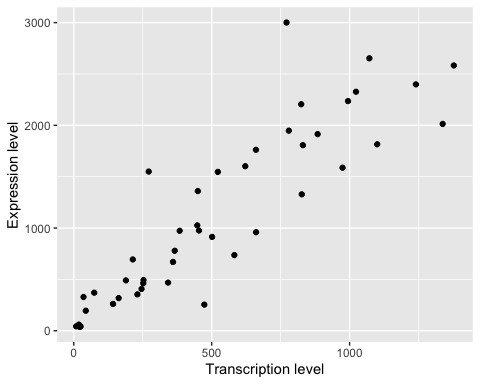
cor.test() produces a table of information similar to the t.test() output. We can focus on the first two lines and the sample estimate at the bottom. In the first line we see the vectors used for the test listing x first, then y. The second line contains the t-value t = 1.9747, the degrees of freedom (2-n), and p-value = 0.05474. At the bottom, is the correlation value 0.28835. The t-value is calculated using the following formula , the degrees of freedom is the sample size minus 2. Based on these data, we see a weak positive correlation between transcript counts and protein expression that is not statistically significant.

What are some biological or technical reasons as to why protein expression measured by western blot might not correlate strongly with the transcript counts?

## Challenge 1A

Use the cor.test() function to test the correlation between early gene transcripts and early gene protein expression quantified by flow cytometry. Create a scatterplot of the data with appropriate labels, run a correlation test and include your interpretation of the correlation.

# scatterplot  
ggplot(Early, aes(x = TranscriptCounts, y = FlowCytGFPSignal)) +  
 geom\_point() +  
 labs(x = "Transcription level",  
 y = "Expression level")



# correlation test  
cor.test(Early$TranscriptCounts, Early$FlowCytGFPSignal, method = "pearson")

##   
## Pearson's product-moment correlation  
##   
## data: Early$TranscriptCounts and Early$FlowCytGFPSignal  
## t = 12.871, df = 43, p-value = 2.407e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.8091179 0.9389764  
## sample estimates:  
## cor   
## 0.8910292

Include a brief description of how you would interpret the correlation.

* There is a strong positive correlation between transcription counts and protein expression measured by flow cytometry.

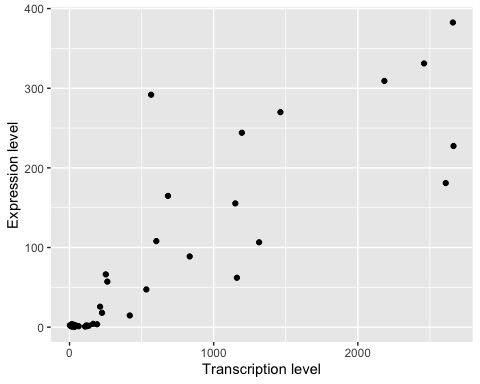
Why might protein expression measured by flow cytometry have a stronger correlation with gene transcription than expression measured by western blotting?

* To properly quantify protein expression in western blotting it requires careful titration of protein lysates, and the antibodies used to detect the proteins you are trying to visualize. Any variability in these steps might lead to a weaker correlation between gene transcript counts and protein expression.
* Using a GFP-Fusion protein directly links a visual output with the protein in question and therefore reduces the number of steps it takes to measure protein expression

## Challenge 1B

Using the approaches we learned above, perform a correlation test to see which method of measuring protein expression (western blot or flow cytometry) has a stronger correlation for the late genes.

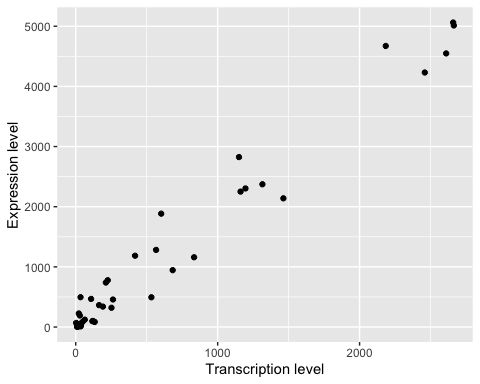
Late <- ProteinExpressionData %>%  
 filter(Cluster == "Late")  
  
ggplot(Late, aes(x = TranscriptCounts, y = WesternBlotSignal)) +  
 geom\_point() +  
 labs(x = "Transcription level",  
 y = "Expression level")



cor.test(Late$TranscriptCounts, Late$WesternBlotSignal, method = "pearson")

##   
## Pearson's product-moment correlation  
##   
## data: Late$TranscriptCounts and Late$WesternBlotSignal  
## t = 11.797, df = 43, p-value = 4.548e-15  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.7808418 0.9291837  
## sample estimates:  
## cor   
## 0.8740421

ggplot(Late, aes(x = TranscriptCounts, y = FlowCytGFPSignal)) +  
 geom\_point() +  
 labs(x = "Transcription level",  
 y = "Expression level")



cor.test(Late$TranscriptCounts, Late$FlowCytGFPSignal, method = "pearson")

##   
## Pearson's product-moment correlation  
##   
## data: Late$TranscriptCounts and Late$FlowCytGFPSignal  
## t = 35.475, df = 43, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.9697086 0.9908676  
## sample estimates:  
## cor   
## 0.9833418

# Both methods for measuring protein expression show a strong and significant correlation between protein expression and transcription.

# Spearman’s rank correlation

Spearman’s rank correlation is a non-parametric equivalent to Pearson’s correlation. It is used to measure the strength and direction of association between two ranked quantitative variables. Spearman’s correlation assesses **monotonic relationships (one variable goes up, the other goes up)** whether or not they form a straight line. See Lecture 7 for examples. Spearman’s rank correlation coefficient can also be obtained using the cor() function or the cor.test() function by setting to method to spearman.

Our sample data does does not show a non-linear relationship, but we can alter the method of our correlation test to adjust the test. Using method = "spearman" will create a rank order for the variables and test to see how the rank of the dependent variable changes relative to the rank of the independent variable. For non-linear relationships the “spearman” test is more appropriate.

# cor.test() between transcription and western blot signal using pearson method  
cor.test(Late$TranscriptCounts, Late$WesternBlotSignal, method = "pearson")

##   
## Pearson's product-moment correlation  
##   
## data: Late$TranscriptCounts and Late$WesternBlotSignal  
## t = 11.797, df = 43, p-value = 4.548e-15  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.7808418 0.9291837  
## sample estimates:  
## cor   
## 0.8740421

# cor.test() function to calculate the correlation coefficient using spearman method  
cor.test(Late$TranscriptCounts, Late$WesternBlotSignal, method = "spearman")

##   
## Spearman's rank correlation rho  
##   
## data: Late$TranscriptCounts and Late$WesternBlotSignal  
## S = 2644, p-value < 2.2e-16  
## alternative hypothesis: true rho is not equal to 0  
## sample estimates:  
## rho   
## 0.8258235

## Interpreting the result

Note that, as Spearman’s method is based on ranks, it cannot compute an exact p-value when there are tied values (i.e., when two are more observations are equal). Instead it will produce an estimate p-value < 2.2e-16. It also does not calculate a t-value. Instead it produces the Spearman test statistic S which is the sum of all the squared rank differences. This value does not tell us much about the data since it is highly dependent on our sample size. Instead we must look to the rho value 0.8256 which is the Spearman’s test version of the R Value. A value closer to 0 has weak or no correlation, and the closer you get to -1 or 1, the closer you get to a strong negative, or strong positive correlation, respectively. For this data, we can say there is a strong positive correlation between the ranked variables that is statistically significant.

Notice that the R value for the Pearson correlation is higher than the value for the Spearman test.

# Linear Regression

A linear regression model describes and explains or predicts the relationship between a dependent variable (aka a response variable) and an independent variable (aka explanatory variable). Regression is different from correlation because it puts variables into equation and thus explains relationship between them. Linear regressions for instance fit the model Y = a + b \* X to the data. Linear regressions are run under the assumption that the data are normally distributed and have similar variance (also described as homoscedasticity).

Linear regressions are obtained with the lm() function, which uses the least square method to calculate a slope that best fits the data. The least squares method creates a line that has the smallest distance between any point and the slope of the line. the distance between the points and the slope are known as residuals. The linear regression follows the following hypotheses:

* Null hypothesis: There is no relationship between your independent and dependent variable.
* Alternative hypothesis: There is a relationship between your independent variable and dependent variable.

When we fit the model Y = a + b X, the lm() returns an estimate for the intercept (a) and the slope (b), with a confidence interval, as well as the p-value assessing whether these estimates as significantly different from 0. The lm() function wants at least two arguments.

* formula = y ~ x a formula where y is the dependent variable and x is the independent variable, these must each be vectors.
* data = a data frame in the global environment from which to draw the variables

Similar to ANOVA, you can assign the lm() to an object and use the summary() function to get additional details about the model. Using our protein expression data lets practice using lm(). We know that our early genes have a weak but not significant correlation between transcription and protein expression when tested by western blot. Let’s see what happens when we create a linear model for this data. In our data let’s see how well we can use protein expression to predict protein transcription.

# lm() function to fit the linear model to the data  
WesternModel <- lm(formula = TranscriptCounts ~ WesternBlotSignal, data = Early)  
  
# Use summary to obtain the estimates for the parameters a and b, as well as the p-value  
summary(WesternModel)

##   
## Call:  
## lm(formula = TranscriptCounts ~ WesternBlotSignal, data = Early)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -560.8 -235.6 -106.2 256.5 865.1   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 323.198 111.294 2.904 0.0058 \*\*  
## WesternBlotSignal 3.341 1.692 1.975 0.0547 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 378.4 on 43 degrees of freedom  
## Multiple R-squared: 0.08315, Adjusted R-squared: 0.06182   
## F-statistic: 3.9 on 1 and 43 DF, p-value: 0.05474

## Interpreting lm()

* In the first chunk we can see the call, the code that was run to create the output.
* The second chunk we see is the 5 number summary of the residuals (this is the distance between each point and the slope of the line created by the model)
* The third chunk is a table with the calculated coefficients.

1. In the (Intercept) row we see the estimate of the y intercept 43.949, standard error for the intercept 8.09, and the t-value 5.432 where the null hypothesis is that the intercept is = 0, we can see this is highly significant.
2. In the TranscriptionCounts row we have the estimate for the slope 0.02489, the standard error 0.0126, and t-value 1.975 where the null hypothesis is that the slope is = 0.

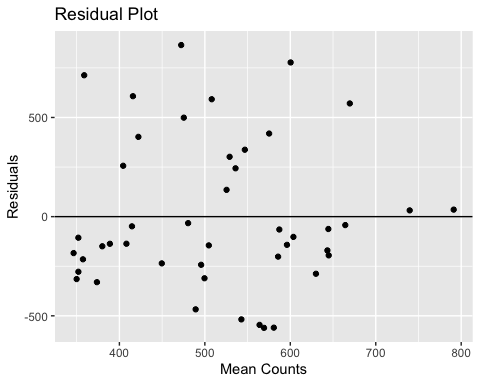
* The bottom chunk includes information about the standard error of the residuals, the R-square values, and F-statistic which is used to interpret the overall results of the model. Here we see a p-value of 0.0547 which is not low enough for us to reject the null hypothesis.

We can conclude that with this data there is no significant relationship between translation measured by western blot, and transcript counts.

## Residual Plots

We have mentioned residual values several times. We have explained that residuals are the distance between any point on the plot and the slope created by the linear model. You can test the quality of your model by creating a residual plot. This plots the residual values with respect to the explanatory variable. The slope of the linear model is shown here as the y-intercept, and the values are plotted relative to the slope.

# Plot the residuals to (visually) check that the assumptions are verified  
# Here we use the object created by lm(). Notice what we set as the x and y variables  
# This syntax will pull from the lm() object without requiring a data frame  
  
ggplot(WesternModel, aes(x = .fitted, y = .resid)) +  
 geom\_point()+  
 geom\_hline(yintercept = 0)+  
 labs(title = "Residual Plot",  
 x = "Mean Counts",  
 y = "Residuals")



## Interpreting the residual plot

You typically want your data points to be randomly dispersed in the residual plot. If there appears to be a pattern, there may be another factor that explains the relationship between the two variables. In this case there does not appear to be a pattern on our residual plot.

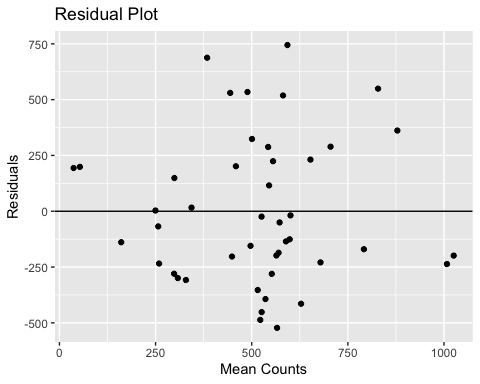
# Multiple Regression Models

Another benefit of linear regressions is that you can easily expand them to multivariate models, including several explanatory variables. In this case we can see how the transcription and time impacts protein expression. Perhaps this will create a better model for helping us predict protein expression based on the western blot signal.

# Add a gene effect  
WesternModel\_Time = lm(formula = TranscriptCounts ~ WesternBlotSignal + TimePoint, data = Early)  
  
  
# Use summary to obtain the estimates for the parameters a and b, as well as the p-value  
summary(WesternModel\_Time)

##   
## Call:  
## lm(formula = TranscriptCounts ~ WesternBlotSignal + TimePoint,   
## data = Early)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -522.48 -234.37 -67.98 224.17 744.53   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 472.953 107.845 4.385 7.61e-05 \*\*\*  
## WesternBlotSignal 6.148 1.705 3.606 0.000819 \*\*\*  
## TimePoint -77.237 22.019 -3.508 0.001091 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 336.7 on 42 degrees of freedom  
## Multiple R-squared: 0.2909, Adjusted R-squared: 0.2571   
## F-statistic: 8.615 on 2 and 42 DF, p-value: 0.0007328

# Plot the residuals to (visually) check that the assumptions are verified  
ggplot(WesternModel\_Time, aes(x = .fitted, y = .resid)) +  
 geom\_point()+  
 geom\_hline(yintercept = 0)+  
 labs(title = "Residual Plot",  
 x = "Mean Counts",  
 y = "Residuals")



When we add in the variable of time, we can see that our model improves substantially. Review the table created from WesternModel\_Time = lm(formula = TranscriptCounts ~ WesternBlotSignal + TimePoint, data = Early). The test statistics for the Intercept, WesternBlotSignal and Timepoint are less than our alpha of 0.05. We can conclude that our factoring in our western signal and timepoint we can use our western data to predict the transcript counts.

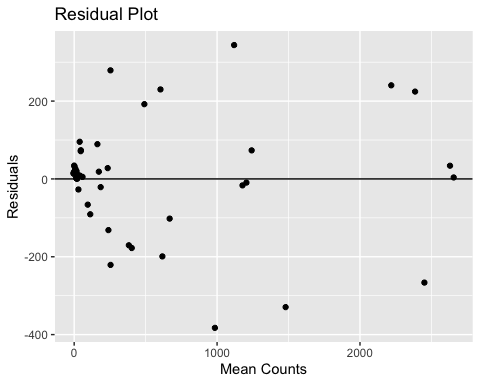
## Exercise 2A

Create a linear model investigating the relationship between fluorescence intensity measured by flow cytometry and transcript counts for late genes. Create a plot depicting the residuals.

# Linear model  
FlowCytModel <- lm(formula = TranscriptCounts ~ FlowCytGFPSignal, data = Late)  
  
# Print summary  
summary(FlowCytModel)

##   
## Call:  
## lm(formula = TranscriptCounts ~ FlowCytGFPSignal, data = Late)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -382.73 -27.13 11.69 34.31 343.99   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -5.94849 27.16468 -0.219 0.828   
## FlowCytGFPSignal 0.52584 0.01482 35.475 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 148.7 on 43 degrees of freedom  
## Multiple R-squared: 0.967, Adjusted R-squared: 0.9662   
## F-statistic: 1258 on 1 and 43 DF, p-value: < 2.2e-16

# Plot the residuals to (visually) check that the assumptions are verified  
ggplot(FlowCytModel, aes(x = .fitted, y = .resid)) +  
 geom\_point()+  
 geom\_hline(yintercept = 0)+  
 labs(title = "Residual Plot",  
 x = "Mean Counts",  
 y = "Residuals")



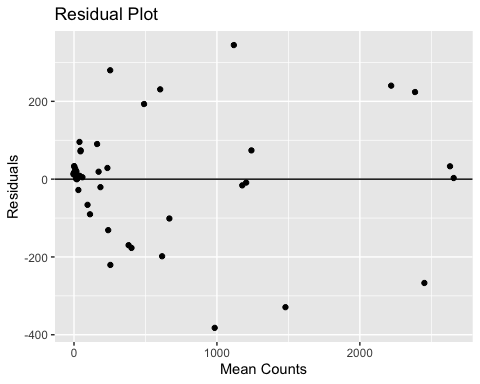
## Exercise 2B

Include time in your linear model investigating the relationship between fluorescence intensity measured by flow cytometry and transcript counts for late genes. Create a plot depicting the residuals.

# Linear model  
FlowCytModelTime <- lm(formula = TranscriptCounts ~ FlowCytGFPSignal + TimePoint, data = Late)  
  
# Print summary  
summary(FlowCytModelTime)

##   
## Call:  
## lm(formula = TranscriptCounts ~ FlowCytGFPSignal + TimePoint,   
## data = Late)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -382.32 -27.91 11.41 33.57 344.71   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -4.72590 47.16781 -0.100 0.921   
## FlowCytGFPSignal 0.52654 0.02657 19.814 <2e-16 \*\*\*  
## TimePoint -0.49093 15.39212 -0.032 0.975   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 150.5 on 42 degrees of freedom  
## Multiple R-squared: 0.967, Adjusted R-squared: 0.9654   
## F-statistic: 614.6 on 2 and 42 DF, p-value: < 2.2e-16

# Plot the residuals to (visually) check that the assumptions are verified  
ggplot(FlowCytModelTime, aes(x = .fitted, y = .resid)) +  
 geom\_point()+  
 geom\_hline(yintercept = 0)+  
 labs(title = "Residual Plot",  
 x = "Mean Counts",  
 y = "Residuals")



## Conclusion

In this section we have explored the ways in which you can visualize and test lineare relationships between quantitative variables. We did this using correlation testing and linear modeling. There are key similarities and differences between correlation analysis and linear regression modeling that are important to remember.

Similarities

* Both tell you the direction and strength of the relationship between two variables
* A negative correlation corresponds to a negative regression slope and vice versa

Differences

* For correlation, X and Y variables are interchageable whereas regression reports how x causes y to change
* Correlation generates descriptive statistics whereas regression produces an equation that can be used to predict future outcomes
* Regression attempts to discover causal relationships

In the next section we will see what happens when we have dozens, hundreds or thousands of independent variables to manage. We will learn about dimensionality reduction which we can use to identify principal components that influence overall variability in a data set.

# Sources and Extra Resources

Source: [Pearson and Spearman Correlation](https://towardsdatascience.com/clearly-explained-pearson-v-s-spearman-correlation-coefficient-ada2f473b8) Source: [Linear Modeling and Regression](https://www.mathworks.com/discovery/linear-model.html) Source: [Visualization Approaches for Correlation Analysis](https://statsandr.com/blog/correlation-coefficient-and-correlation-test-in-r/)