R analysis

Jesse Hammond

9/20/2017

## Before we start: setting up the workspace

Before working in R, it is necessary to set up the "workspace": the virtual environment in which you can load, manipulate, and analyze data. The code below cleans the workspace, erasing any previous objects or functions; sets the working directory, from which we'll load the data to analyze, and also loads a set of "packages" of useful functions that make data cleaning and analysis easier and faster.

##########################  
##  
## Setting up workspace  
##  
##########################  
  
## Clear previous workspace, if any  
rm(list=ls())  
  
## Set working directory  
os\_detect <- Sys.info()['sysname']  
if (os\_detect == 'Darwin'){  
 setwd('/Users/localadmin/Dropbox/Research/StatsChapter')  
}   
  
## Load packages for analysis  
pacman::p\_load(  
 data.table, tidyverse, ggplot2, stargazer, easynls, pscl, pander  
)  
  
##########################  
##  
## Reading in the data sets used in this chapter  
##  
##########################  
  
## Read in spring data  
spring\_data <- read\_csv('./Data/01\_correlation.csv')

## Parsed with column specification:  
## cols(  
## x = col\_integer(),  
## y = col\_double()  
## )

sigacts\_data <- read\_csv('./Data/06\_poisson\_sigacts.csv')

## Parsed with column specification:  
## cols(  
## sigacts\_2008 = col\_integer(),  
## ggi\_2008 = col\_double(),  
## literacy = col\_double(),  
## poverty = col\_double()  
## )

recovery\_data <- read\_csv('./Data/02\_exponential\_decay.csv')

## Parsed with column specification:  
## cols(  
## T = col\_integer(),  
## Y = col\_integer()  
## )

shipping\_data <- read\_csv('./Data/03\_sine\_regression\_shipping.csv')

## Parsed with column specification:  
## cols(  
## Month = col\_integer(),  
## UsageTons = col\_integer()  
## )

afghan\_data <- read\_csv('./Data/04\_sine\_regression\_casualties.csv')

## Parsed with column specification:  
## cols(  
## Year = col\_integer(),  
## Month = col\_integer(),  
## Casualties = col\_integer()  
## )

war\_data <- read\_csv('./Data/05\_bin\_logit\_conflict.csv')

## Parsed with column specification:  
## cols(  
## side\_a = col\_integer(),  
## cd\_pct = col\_double()  
## )

sigacts\_data <- read\_csv('./Data/06\_poisson\_sigacts.csv')

## Parsed with column specification:  
## cols(  
## sigacts\_2008 = col\_integer(),  
## ggi\_2008 = col\_double(),  
## literacy = col\_double(),  
## poverty = col\_double()  
## )

alliance\_data <- read\_csv('./Data/07\_bin\_logit\_alliance.csv')

## Parsed with column specification:  
## cols(  
## statea = col\_character(),  
## stateb = col\_character(),  
## alliance\_present = col\_integer(),  
## igo\_overlap = col\_integer()  
## )

## Format and subset casualties data  
afghan\_data <- mutate(  
 afghan\_data  
 , Date = as.Date(paste0(Year, '-', Month, '-', '01'), format = '%Y-%m-%d')  
 ) %>% filter(  
 Date >= as.Date('2006-01-01')  
 , Date <= as.Date('2008-12-01')  
 ) %>% mutate(  
 DateIndex = 1:36  
 )

## Section 1: Introducing linear regression

### 1.1.1: Correlation of spring data

Calculating correlation between two (or more) variables in R is simple. After loading in the spring data set as an object into R's workspace, we can first visualize the data in tabular format. This lets us be sure that the data is in the proper format and that there are no oddities (missing values, characters entered instead of numbers) that would cause problems.

## Print data as a tibble  
print(spring\_data)

## # A tibble: 11 × 2  
## x y  
## <int> <dbl>  
## 1 50 0.1000  
## 2 100 0.1875  
## 3 150 0.2750  
## 4 200 0.3250  
## 5 250 0.4375  
## 6 300 0.4875  
## 7 350 0.5675  
## 8 400 0.6500  
## 9 450 0.7250  
## 10 500 0.8000  
## 11 550 0.8750

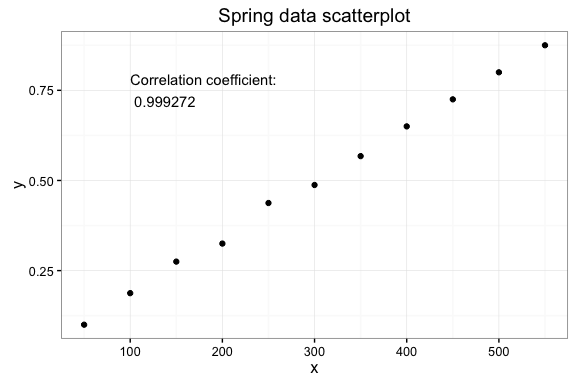
The data looks to be in order. To estimate the correlation between the two columns in this data set, we simply use the cor() command in R on the data table:

## Calculate and print correlation matrix  
print(cor(spring\_data))

## x y  
## x 1.0000000 0.9992718  
## y 0.9992718 1.0000000

The data correlate very closely. Visualizing the data makes this relationship easy to see:

## Generate a plot visualizing the data  
spring\_cor\_plot <- ggplot(  
 aes(x = x, y = y)  
 , data = spring\_data) +   
 geom\_point() +   
 annotate(  
 'text'  
 , x = 100  
 , y = 0.75  
 , label = 'Correlation coefficient:\n 0.999272'  
 , hjust = 0) +   
 ggtitle('Spring data scatterplot') +   
 theme\_bw()  
  
## Print the plot to console  
plot(spring\_cor\_plot)



### 1.1.2: Linear regression of spring data

Fitting an ordinary least-squares (OLS) model with form to the spring data in R is quite simple. Using the lm() command (short for "linear model") fits the linear model and saves the result as another object in R's workspace.

## Fit OLS model to the data  
spring\_model <- lm(  
 y ~ x  
 , data = spring\_data  
 )

We can then operate on this object to produce tables presenting coefficient estimates and a range of diagnostic statistics to evaluate how well the model fits the data provided.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| **x** | 0.001537 | 1.957e-05 | 78.57 | 4.437e-14 |
| **(Intercept)** | 0.03245 | 0.006635 | 4.891 | 0.0008579 |

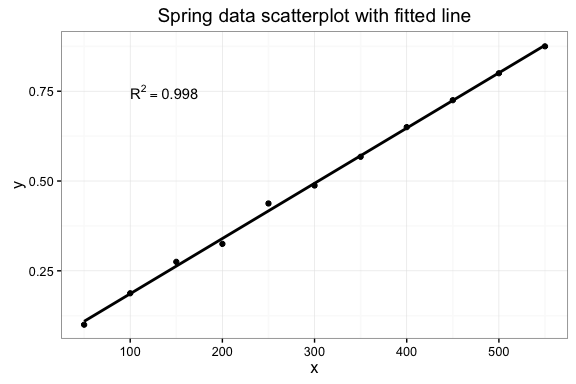
Fitting linear model: y ~ x

|  |  |  |  |
| --- | --- | --- | --- |
| Observations | Residual Std. Error |  | Adjusted |
| 11 | 0.01026 | 0.9985 | 0.9984 |

Analysis of Variance Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| **x** | 1 | 0.6499 | 0.6499 | 6173 | 4.437e-14 |
| **Residuals** | 9 | 0.0009475 | 0.0001053 | NA | NA |

We can visualize this estimated relationship by overlaying the fitted line to the spring data plot. This plot shows that the trend line estimated by the linear model fits the data quite well.



### 1.1.3: Linear regression of Philippines SIGACTS

## Fit OLS model to the data  
sigacts\_model\_ols <- lm(  
 sigacts\_2008 ~ literacy  
 , data = sigacts\_data)

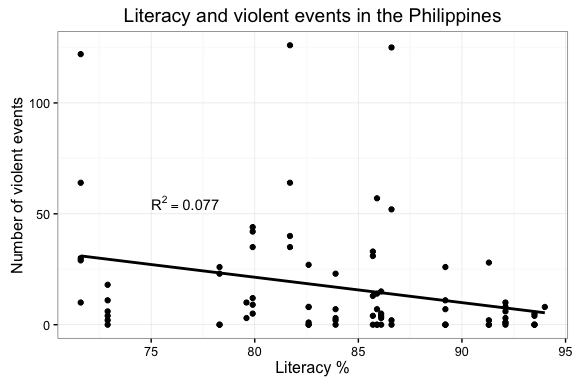
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| **literacy** | -1.145 | 0.4502 | -2.543 | 0.01297 |
| **(Intercept)** | 113 | 37.99 | 2.975 | 0.003903 |

Fitting linear model: sigacts\_2008 ~ literacy

|  |  |  |  |
| --- | --- | --- | --- |
| Observations | Residual Std. Error |  | Adjusted |
| 80 | 25.77 | 0.07656 | 0.06472 |

Analysis of Variance Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| **literacy** | 1 | 4295 | 4295 | 6.467 | 0.01297 |
| **Residuals** | 78 | 51805 | 664.2 | NA | NA |



## Section 2: Exponential decay modeling

### 1.2.1: Introducing hospital recovery data

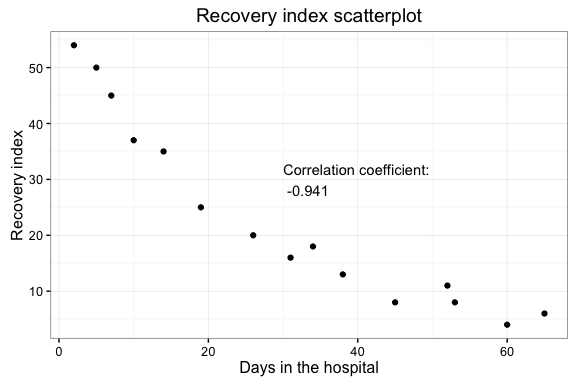
## # A tibble: 15 × 2  
## T Y  
## <int> <int>  
## 1 2 54  
## 2 5 50  
## 3 7 45  
## 4 10 37  
## 5 14 35  
## 6 19 25  
## 7 26 20  
## 8 31 16  
## 9 34 18  
## 10 38 13  
## 11 45 8  
## 12 52 11  
## 13 53 8  
## 14 60 4  
## 15 65 6

Printing the table of recovery data shows that once again, the structure of the data is amenable to statistical analysis. We have two columns, (number of days in the hospital) and (estimated recovery index) and we want to generate a model that predicts how well a patient will recover as a function of the time they spend in the hospital. Using the cor() command retrieves an initial correlation coefficient of -0.941:

## Calculate and print correlation matrix  
print(cor(recovery\_data))

## T Y  
## T 1.0000000 -0.9410528  
## Y -0.9410528 1.0000000

Once again, creating a scatter plot of the data helps us visualize how closely the estimated correlation value matches the overall trend in the data.



### 1.2.2: Linear regression of hospital recovery data

It definitely appears that there is a strong negative relationship: the longer a patient spends in the hospital, the lower their recovery index tends to be. Next, we fit an OLS model to the data to estimate the magnitude of the linear relationship.

## Fit OLS model to the data  
recovery\_model <- lm(  
 Y ~ T  
 , data = recovery\_data  
 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| **T** | -0.7525 | 0.07502 | -10.03 | 1.736e-07 |
| **(Intercept)** | 46.46 | 2.762 | 16.82 | 3.335e-10 |

Fitting linear model: Y ~ T

|  |  |  |  |
| --- | --- | --- | --- |
| Observations | Residual Std. Error |  | Adjusted |
| 15 | 5.891 | 0.8856 | 0.8768 |

Analysis of Variance Table

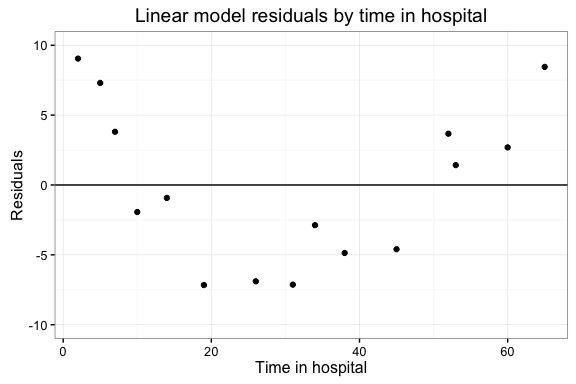
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| **T** | 1 | 3492 | 3492 | 100.6 | 1.736e-07 |
| **Residuals** | 13 | 451.2 | 34.71 | NA | NA |

OLS modeling shows that there is a negative and statistically significant relationship between time spent in the hospital and patient recovery index. However, ordinary least-squares regression may not be the best choice in this case for two reasons. First, we are dealing with real-world data: a model that can produce (for example) negative estimates of recovery index is not applicable to the underlying concepts our model is dealing with. Second, the assumption of OLS, like all linear models, is that the magnitude of the relationship between input and output variables stays constant over the entire range of values in the data. However, visualizing the data suggests that this assumption may not hold --- in fact, it appears that the magnitude of the relationship is very high for low values of and decays somewhat for patients who spend more days in the hospital.

To test for this phenomenon (technically referred to as ) we examine the residuals of the linear model. Residuals analysis can provide quick visual feedback about model fit and whether the relationships estimated hold over the full range of the data. We calcuate residuals as the difference between observed values and estimated values $Y&\*$, or . We then normalize residuals as percent relative error between the observed and estimated values, which helps us compare how well the model predicts each individual observation in the data set:

## # A tibble: 15 × 6  
## T Y index predicted residuals pct\_relative\_error  
## <int> <int> <int> <dbl> <dbl> <dbl>  
## 1 2 54 1 44.955397 9.0446035 16.749266  
## 2 5 50 2 42.697873 7.3021275 14.604255  
## 3 7 45 3 41.192857 3.8071435 8.460319  
## 4 10 37 4 38.935333 -1.9353325 -5.230628  
## 5 14 35 5 35.925301 -0.9253005 -2.643716  
## 6 19 25 6 32.162761 -7.1627605 -28.651042  
## 7 26 20 7 26.895205 -6.8952045 -34.476023  
## 8 31 16 8 23.132665 -7.1326645 -44.579153  
## 9 34 18 9 20.875141 -2.8751405 -15.973003  
## 10 38 13 10 17.865109 -4.8651085 -37.423912  
## 11 45 8 11 12.597553 -4.5975525 -57.469407  
## 12 52 11 12 7.329997 3.6700035 33.363668  
## 13 53 8 13 6.577489 1.4225115 17.781393  
## 14 60 4 14 1.309933 2.6900675 67.251686  
## 15 65 6 15 -2.452607 8.4526075 140.876791

These data can also be plotted to visualize how well the model fits over the range of our input variable.



The residuals plotted here show a curvilinear pattern, decreasing and then increasing in magnitude over the range of the input variable. This means that we can likely improve the fit of the model by allowing for non-linear effects. Furthermore, the current model can make predictions that are substantively nonsensical, even if they were statistically valid. For example, our model predicts that after 100 days in the hospital, a patient's estimated recovery index value would be -29.79. This has no common sense, as the recovery index variable is always positive in the real world. By allowing for non-linear terms, we can also guard against these types of nonsense predictions.

### 1.2.3: Quadratic regression of hospital recovery data

Including a quadratic term modifies the model formula: Fitting this model to the data produces separate estimates of the effect of itself as well as the effect of , the quadratic term.

## Generate model  
recovery\_model2 <- lm(Y ~ T + I(T^2), data = recovery\_data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| **T** | -1.71 | 0.1248 | -13.7 | 1.087e-08 |
| **I(T^2)** | 0.01481 | 0.001868 | 7.927 | 4.127e-06 |
| **(Intercept)** | 55.82 | 1.649 | 33.85 | 2.811e-13 |

Fitting linear model: Y ~ T + I(T^2)

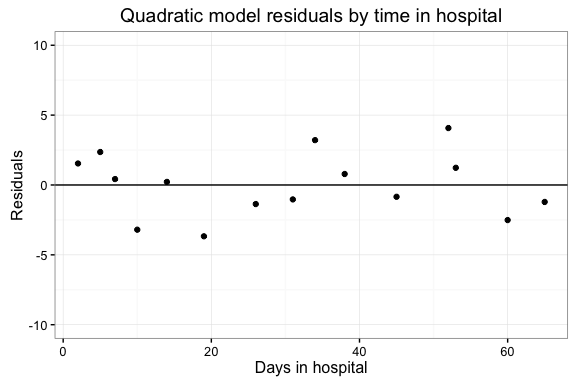
|  |  |  |  |
| --- | --- | --- | --- |
| Observations | Residual Std. Error |  | Adjusted |
| 15 | 2.455 | 0.9817 | 0.9786 |

Analysis of Variance Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| **T** | 1 | 3492 | 3492 | 579.3 | 1.59e-11 |
| **I(T^2)** | 1 | 378.9 | 378.9 | 62.84 | 4.127e-06 |
| **Residuals** | 12 | 72.34 | 6.029 | NA | NA |

Including the quadratic term improves model fit as measured by from 0.88 to 0.98 --- a sizable increase. To assess whether this new input variable deals with the curvilinear trend we saw in the residuals from the first model, we re-calculate and visualize the residuals from the quadratic model:

## # A tibble: 15 × 6  
## T Y index predicted residuals pct\_relative\_error  
## <int> <int> <int> <dbl> <dbl> <dbl>  
## 1 2 54 1 52.460836 1.5391644 2.8503045  
## 2 5 50 2 47.640993 2.3590072 4.7180144  
## 3 7 45 3 44.575834 0.4241663 0.9425917  
## 4 10 37 4 40.200199 -3.2001992 -8.6491871  
## 5 14 35 5 34.780614 0.2193857 0.6268164  
## 6 19 25 6 28.672445 -3.6724455 -14.6897820  
## 7 26 20 7 21.364792 -1.3647924 -6.8239618  
## 8 31 16 8 17.033457 -1.0334567 -6.4591042  
## 9 34 18 9 14.790022 3.2099781 17.8332119  
## 10 38 13 10 12.213370 0.7866302 6.0510012  
## 11 45 8 11 8.844363 -0.8443634 -10.5545422  
## 12 52 11 12 6.926437 4.0735627 37.0323886  
## 13 53 8 13 6.770903 1.2290967 15.3637082  
## 14 60 4 14 6.511355 -2.5113548 -62.7838691  
## 15 65 6 15 7.214379 -1.2143795 -20.2396576



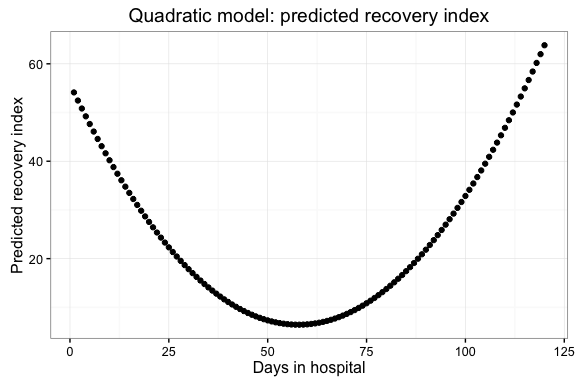
Visually evaluating the residuals from the quadratic model shows that the trend has disappeared. This means that we can assume the same relationship holds whether or . However, we are still not sure if the model produces estimates that pass the common-sense test. The simplest way to assess this is to generate predicted values of the recovery index variable using the quadratic model, and plot them to see if they make sense.

To generate predicted values in R, we can pass the quadratic model object to the predict() function along with a set of hypothetical input values. In other words, we can ask the model what the recovery index would look like for a set of hypothetical patients who spend anywhere from zero to 120 days in the hospital.

## Create a set of hypothetical patient observations with days in the hospital from 1 to 120  
patient\_days = tibble(T = 1:120)  
  
## Feed the new data to the model to generate predicted recovery index values  
predicted\_values = predict(  
 recovery\_model2  
 , newdata = patient\_days  
 )

We can then plot these estimates to quickly gauge whether they pass the common-sense test for real-world predictive value.

## # A tibble: 5 × 2  
## T predicted  
## <int> <dbl>  
## 1 1 54.12668  
## 2 2 52.46084  
## 3 3 50.82461  
## 4 4 49.21799  
## 5 5 47.64099



The predicted values curve up toward infinity; clearly this is a problem. The quadratic term we included in the model leads to unrealistic estimates of recovery index at larger values of . Not only is this unacceptable for the context of our model, but it is unrealistic on its face. After all, we understand that people generally spend long periods in the hospital for serious or life-threatening conditions such as severe disease or major bodily injury. As such, we can assess that someone who spends six months in the hospital probably should not have a higher recovery index than someone who was only hospitalized for a day or two.

### 1.2.4: Exponential decay modeling of hospital recovery data

We may be able to build a model that both accurately fits the data and produces estimates that pass the common-sense test by using an exponential decay model. This modeling appraoch lets us model relationships that vary over time in a non-linear fashion --- in this case, we want to accurately capture the strong correlation for lower ranges of , but allow the magnitude of this relationship to decay as increases, as the data seems to indicate.

Generating non-linear models in R is done using the non-linear least squares or NLS function, appropriately labeled nls(). This function automatically fits a wide range of non-linear models based on a functional form designated by the user. It is important to note that when fitting an NLS model in R, minimizing the sum of squares is done computationally rather than mathematicaly. That means that the choice of starting values for the optimization function is important --- the estimates produced by the model may vary considerably based on the chosen starting values! As such, it is wise to experiment when fitting these non-linear values to test how robust the resulting estimates are to the choice of starting values.

## Fit NLS model to the data  
## Generate model  
recovery\_model3 <- nls(  
 Y ~ a \* (exp(b \* T))  
 , data = recovery\_data  
 , start = c(  
 a=1  
 , b=0.05  
 )  
 , trace = T  
)

## 11198.06 : 1.00 0.05  
## 9903.899 : 10.3397660 -0.1171301  
## 5127.568 : 1.445986e+01 1.861608e-04  
## 3205.214 : 22.43186530 -0.01280346  
## 1099.753 : 37.70529163 -0.02984728  
## 68.00069 : 57.77929137 -0.04158464  
## 49.46803 : 58.62130670 -0.03954164  
## 49.4593 : 58.60566528 -0.03958494  
## 49.4593 : 58.6065363 -0.0395864

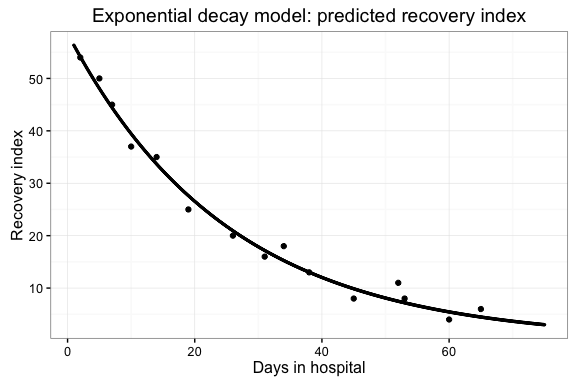
Fitting nonlinear regression model: Y ~ a \* (exp(b \* T))

Parameter Estimates

|  |  |
| --- | --- |
| a | b |
| 58.61 | -0.03959 |

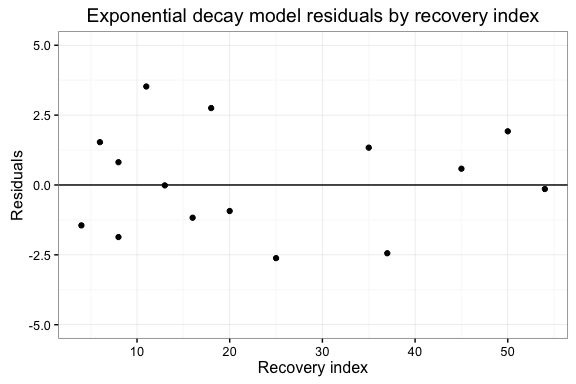
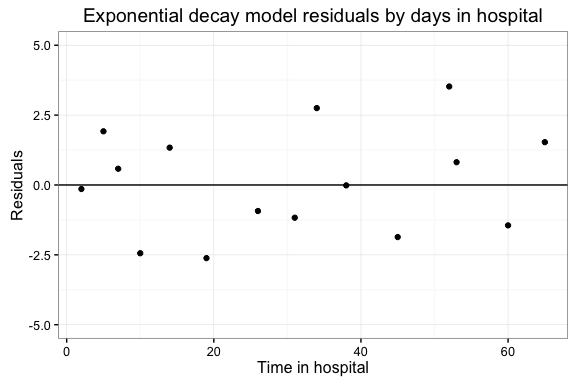
residual sum-of-squares: 1.951

Overlaying the trend produced by the model on the plot of observed values, we see that the NLS modeling approach fits the data very well

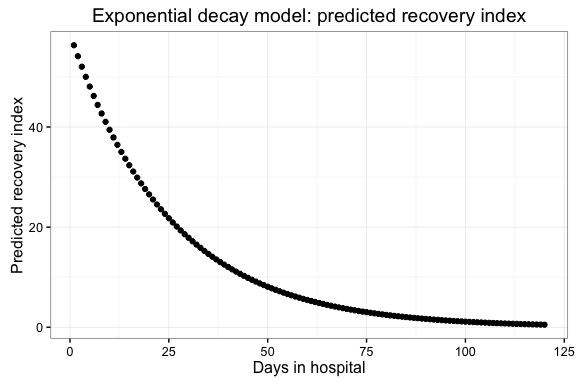


Once again, we can visually assess model fit by calculating and plotting the residuals. The figures below show the same residuals plotted along both days in the hospital and recovery index .

## # A tibble: 15 × 5  
## T Y predicted residuals pct\_relative\_error  
## <int> <int> <dbl> <dbl> <dbl>  
## 1 2 54 54.143916 -0.14391597 -0.2665110  
## 2 5 50 48.079006 1.92099420 3.8419884  
## 3 7 45 44.418036 0.58196361 1.2932525  
## 4 10 37 39.442567 -2.44256693 -6.6015322  
## 5 14 35 33.664559 1.33544135 3.8155467  
## 6 19 25 27.617389 -2.61738904 -10.4695562  
## 7 26 20 20.931299 -0.93129928 -4.6564964  
## 8 31 16 17.171407 -1.17140692 -7.3212932  
## 9 34 18 15.247958 2.75204170 15.2891206  
## 10 38 13 13.014259 -0.01425912 -0.1096856  
## 11 45 8 9.863545 -1.86354475 -23.2943094  
## 12 52 11 7.475609 3.52439082 32.0399165  
## 13 53 8 7.185360 0.81464008 10.1830010  
## 14 60 4 5.445805 -1.44580513 -36.1451283  
## 15 65 6 4.467574 1.53242564 25.5404274



In both cases, we see that there is no easily distinguishable pattern in residuals. Finally, we apply the common-sense check by generating and plotting estimated recovery index values for a set of values of from 1 to 120.



The predicted values generated by the exponential decay model make intuitive sense. As the number of days a patient spends in the hospital increases, the model predicts that their recovery index will decrease at a decreasing rate. This means that while the recovery index variable will continuously decrease, it will not take on negative values (as predicted by the linear model) or explosively large values (as predicted by the quadratic model). It appears that the exponential decay model not only fit the data best from a purely statistical point of view, but also generates values that pass the common-sense test to an observer.

## Section 3: Sinusoidal regression

### 1.3.1: Introducing shipping data

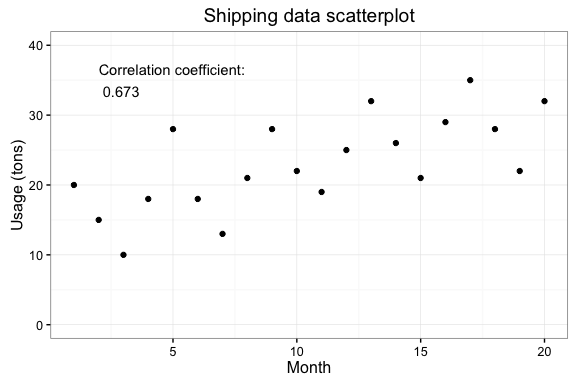
## Print data as a tibble  
print(shipping\_data)

## # A tibble: 20 × 2  
## Month UsageTons  
## <int> <int>  
## 1 1 20  
## 2 2 15  
## 3 3 10  
## 4 4 18  
## 5 5 28  
## 6 6 18  
## 7 7 13  
## 8 8 21  
## 9 9 28  
## 10 10 22  
## 11 11 19  
## 12 12 25  
## 13 13 32  
## 14 14 26  
## 15 15 21  
## 16 16 29  
## 17 17 35  
## 18 18 28  
## 19 19 22  
## 20 20 32

## Calculate and print correlation matrix  
print(cor(shipping\_data))

## Month UsageTons  
## Month 1.0000000 0.6725644  
## UsageTons 0.6725644 1.0000000

Once again, we can visualize the data in a scatter plot to assess whether this positive correlation is borne out by the overall trend.



Visualizing the data, we see that there is a clear positive trend over time in shipping usage. However, examining the data in more detail suggests that a simple linear model may not be best-suited to capturing the variation in these data. One way to plot more complex patterns in data is through the use of a trend line using polynomial or non-parametric smoothing functions.



Plotting a trend line generated via a spline function shows that there seems to be an oscillating pattern with a steady increase over time in the shipping data.

### 1.3.2: Linear regression of shipping data

As a baseline for comparison, we begin by fitting a standard OLS regression model using the lm() function in R.

## Generate model  
shipping\_model1 <- lm(UsageTons ~ Month, data = shipping\_data)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| **Month** | 0.7594 | 0.1969 | 3.856 | 0.001158 |
| **(Intercept)** | 15.13 | 2.359 | 6.411 | 4.907e-06 |

Fitting linear model: UsageTons ~ Month

|  |  |  |  |
| --- | --- | --- | --- |
| Observations | Residual Std. Error |  | Adjusted |
| 20 | 5.079 | 0.4523 | 0.4219 |

Analysis of Variance Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| **Month** | 1 | 383.5 | 383.5 | 14.87 | 0.001158 |
| **Residuals** | 18 | 464.3 | 25.79 | NA | NA |

While the linear model fits the data fairly well, the oscillation identified by the spline visualization suggests that we should apply a model that better fits the seasonal variation in the data.

### 1.3.3: Sinusoidal regression of shipping data

R treats sinusoidal regression models as part of the larger family of nonlinear least-squares (NLS) regression models. This means that we can fit a sinusoidal model using the same nls() function and syntax as we applied earlier for the exponential decay model. The functional form for the sinusoidal model we use here can be written as:

This function can be expanded out trigonometrically as:

This equation can be passed to nls() and R will computationally assess best-fit values for the and terms. It is worth stressing again the importance of selecting good starting values for this process, especially for a model like this one with many parameters to be simultaneously estimated. Here, we set starting values based on pre-analysis of the data. It is also important to note that because the underlying algorithms used to optimize these functions differ between Excel and R, the two methods produce models with different parameters but nearly identical predictive qualities. The model can be specified in R as follows.

## Generate model  
shipping\_model2 <- nls(  
 UsageTons ~ a \* Month + b\*sin(c\*Month) + d\*cos(c\*Month) + e  
 , data = shipping\_data  
 , start = c(  
 a=5  
 , b=10  
 , c=1  
 , d=1  
 , e=10  
 )  
 , trace = T  
)

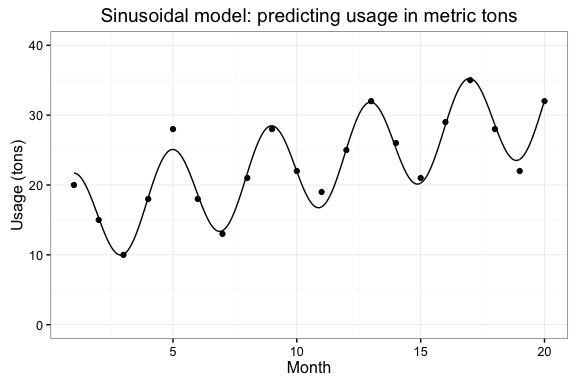
## 45042.53 : 5 10 1 1 10  
## 663.046 : 0.7736951 -1.5386559 0.9616379 4.2289392 15.3202771  
## 458.8408 : 0.7425778 -0.8555162 0.9595757 -0.1801324 15.3201414  
## 380.7506 : 0.7687895 -1.5130803 1.3777092 3.7655460 15.3260166  
## 126.2517 : 0.8345061 2.8210187 1.4873132 4.9231336 14.6378498  
## 99.34191 : 0.8624599 8.1301206 1.5831905 2.1469958 14.0661108  
## 22.29434 : 0.8478613 6.4959072 1.5747331 0.5860112 14.1975698  
## 21.80271 : 0.8479764 6.6646276 1.5733725 0.5579267 14.1866924  
## 21.80233 : 0.8479494 6.6663745 1.5735053 0.5518694 14.1865381  
## 21.80233 : 0.8479513 6.6663622 1.5735011 0.5520718 14.1865329

Fitting nonlinear regression model: UsageTons ~ a \* Month + b \* sin(c \* Month) + d \* cos(c \* Month) + e

Parameter Estimates

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | d | e |
| 0.848 | 6.666 | 1.574 | 0.5521 | 14.19 |

residual sum-of-squares: 1.206

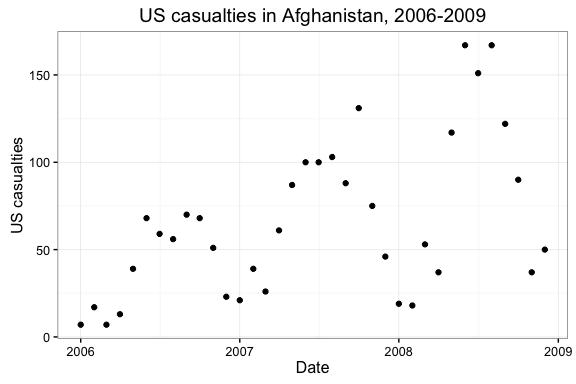


Plotting the trend line produced by the sinusoidal model shows that this modeling approach fits the data much better, accounting for both the short-term seasonal variation and the long-term increase in shipping usage.

## # A tibble: 20 × 5  
## Month UsageTons predicted residuals pct\_relative\_error  
## <int> <int> <dbl> <dbl> <dbl>  
## 1 1 20 21.69933 -1.69932877 -8.4966439  
## 2 2 15 15.29431 -0.29430990 -1.9620660  
## 3 3 10 10.06872 -0.06872371 -0.6872371  
## 4 4 18 18.20250 -0.20250034 -1.1250019  
## 5 5 28 25.08458 2.91542389 10.4122282  
## 6 6 18 18.61406 -0.61406020 -3.4114456  
## 7 7 13 13.46748 -0.46747670 -3.5959746  
## 8 8 21 21.66632 -0.66632300 -3.1729667  
## 9 9 28 28.46904 -0.46904408 -1.6751574  
## 10 10 22 21.93389 0.06611222 0.3005101  
## 11 11 19 16.86701 2.13299136 11.2262703  
## 12 12 25 25.13006 -0.13006417 -0.5202567  
## 13 13 32 31.85273 0.14726646 0.4602077  
## 14 14 26 25.25380 0.74619894 2.8699959  
## 15 15 21 20.26732 0.73268137 3.4889589  
## 16 16 29 28.59372 0.40628458 1.4009813  
## 17 17 35 35.23565 -0.23564545 -0.6732727  
## 18 18 28 28.57381 -0.57380846 -2.0493159  
## 19 19 22 23.66841 -1.66840566 -7.5836621  
## 20 20 32 32.05727 -0.05726836 -0.1789636

Analysis of model residuals bears this out, and also highlights the difference in solving method between Excel and R. The model fitted in R has different parameter estimates and slightly worse model fit (average percent relative error of 3.26% as opposed to the 3.03% from the Excel-fitted model) but the overall trend identified in the data is virtually identical.

### 1.3.4: Introducing Afghanistan casualty data



### 1.3.5: Sinusoidal regression of Afghanistan casualties

Visualizing data on casualties in Afghanistan between 2006 and 2008 shows an increasing trend overall, and significant seasonal oscillation. Once again, we want to fit a non-linear model that accounts for the oscillation present in the data. We use the same sinusoidal functional form

which as before can be expressed as

We fit the model using the nls() function once again:

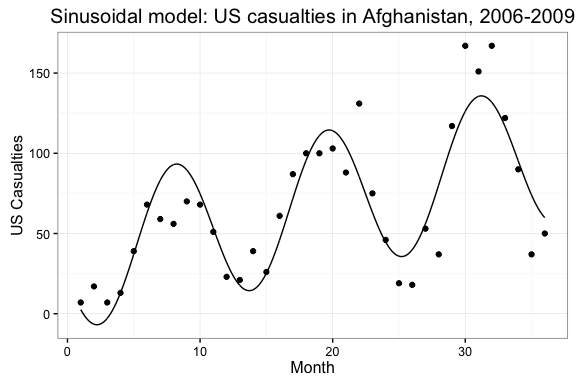
## 138985.7 : 1 10 1 10 1  
## 49982.63 : 2.258671 1.356303 1.029391 1.353397 24.619633  
## 35427.2 : 2.2068763 -9.4273932 0.6196208 15.1463459 26.0931204  
## 24267.23 : 2.1318806 -20.5140293 0.5758111 12.0966232 27.9982283  
## 18101.34 : 1.9724999 -34.2695106 0.5582244 7.4221535 31.3617905  
## 14806.36 : 1.8863900 -44.3682081 0.5584127 -0.9302422 33.3906523  
## 14498.11 : 1.8238454 -44.3050238 0.5480953 -9.2653506 34.0511485  
## 14423.67 : 1.8575261 -43.3800773 0.5492376 -10.3036492 33.5078205  
## 14411.39 : 1.8442981 -43.0865153 0.5471146 -11.8346130 33.6267615  
## 14408.94 : 1.851289 -43.005847 0.547460 -11.949002 33.525643  
## 14408.47 : 1.8484853 -42.9396313 0.5470361 -12.2282483 33.5512462  
## 14408.37 : 1.8499620 -42.9311527 0.5471265 -12.2338063 33.5310068  
## 14408.35 : 1.8493503 -42.9186154 0.5470412 -12.2859475 33.5369241  
## 14408.35 : 1.8496642 -42.9178536 0.5470632 -12.2839406 33.5327763  
## 14408.35 : 1.8495306 -42.9155162 0.5470459 -12.2938756 33.5341312  
## 14408.35 : 1.8495976 -42.9155139 0.5470511 -12.2929106 33.5332702  
## 14408.35 : 1.8495685 -42.9150732 0.5470475 -12.2948389 33.5335764  
## 14408.35 : 1.8495828 -42.9150982 0.5470487 -12.2945468 33.5333961  
## 14408.35 : 1.8495765 -42.9150134 0.5470479 -12.2949269 33.5334639

Fitting nonlinear regression model: Casualties ~ a \* DateIndex + b \* sin(c \* DateIndex) + d \* cos(c \* DateIndex) + e

Parameter Estimates

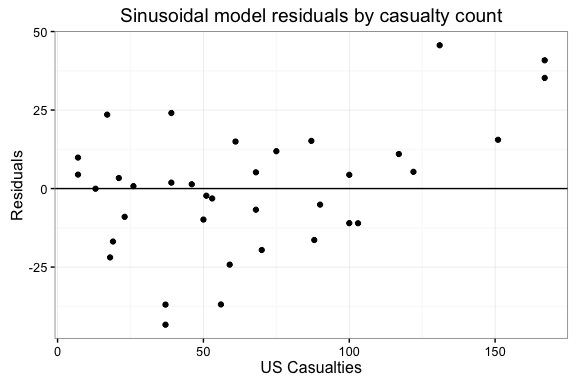
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | d | e |
| 1.85 | -42.92 | 0.547 | -12.29 | 33.53 |

residual sum-of-squares: 21.56



Plotting the trend line identified by the sinusoidal model shows again that the sinusoidal modeling approach can account for both short-term osscilation and long-term increase. We can now estimate residuals and error metrics, and assess how well the model fits over the full range of the data.

## # A tibble: 36 × 8  
## Year Month Casualties Date DateIndex predicted residuals  
## <int> <int> <int> <date> <int> <dbl> <dbl>  
## 1 2006 1 7 2006-01-01 1 2.559361 4.4406385  
## 2 2006 2 17 2006-02-01 2 -6.539489 23.5394894  
## 3 2006 3 7 2006-03-01 3 -2.862472 9.8624722  
## 4 2006 4 13 2006-04-01 4 13.057034 -0.0570337  
## 5 2006 5 39 2006-05-01 5 37.112405 1.8875953  
## 6 2006 6 68 2006-06-01 6 62.822383 5.1776168  
## 7 2006 7 59 2006-07-01 7 83.222777 -24.2227773  
## 8 2006 8 56 2006-08-01 8 92.899116 -36.8991161  
## 9 2006 9 70 2006-09-01 9 89.566984 -19.5669840  
## 10 2006 10 68 2006-10-01 10 74.738778 -6.7387785  
## # ... with 26 more rows, and 1 more variables: pct\_relative\_error <dbl>

 Again, this highlights both the importance of starting values and the difference in estimation method between R and Excel. Despite using different starting values and estimating very different parameters, each model produces very similar estimates of casualties over time: for the Excel model is 14415.2125, almost identical to the R model of 14408.35.

## Section 1.4: Logistic regression

### 1.4.1: Introducing conflict outcome data

### 1.4.2: Logistic regression of conflict outcome data

Conflict outcomes differ from the data we've examined so far in that the measure of state victory only has two values, 1 and 0. This type of data be modeled using a binomial logistic (or sometimes "logit") regression. Logistic regression estimates an underlying continuous variable usually referred to as that is then transformed into an estimate bounded below by 0 and above by 1. This means the logistic modeling approach is extremely useful for estimating binary (1/0) outcomes, as the estimated values can be easily translated into either point estimates or log-probabilities of observing a 1 versus a 0:

The logistic model in R is treated as one case of a broader range of generalized linear models (GLM), and can be accessed via the conveniently named glm() function. Note that because glm() implements a wide range of generalized linear models based on the inputs provided, it is necessary for the user to specify both the family of model (binomial) and the link function (logit).

## Generate model  
war\_model <- glm(  
 side\_a ~ cd\_pct  
 , data = war\_data  
 , family = binomial(link = 'logit')  
 )

Fitting generalized (binomial/logit) linear model: side\_a ~ cd\_pct

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| **cd\_pct** | 1.85 | 2.556 | 0.7237 | 0.4692 |
| **(Intercept)** | 0.004716 | 1.925 | 0.00245 | 0.998 |

Logistic regression shows that there is a positive correlation between civilian casualties and state victory, but that this relationship is not statistically significant at the level. This means we cannot reject the null hypothesis that no relationship exists between the input and output variables.

### 1.4.3: Introducing international alliance data

We now turn to a larger data set, measuring alliance connections between politically relevant states (powerful states and those that share a border with one another) in the international system in the year 2000. Scholars are often interested in assessing the factors that predict whether two states will form a military alliance, as these are salient and lasting forms of cooperation that signal trust (or at least, a lack of overt enmity) between governments.

Coupled with data on whether or not an alliance exists, we also have data on the level of membership overlap each pair of states shares in major intergovernmental organizations (IGOs). These IGOs include major international entities such as the United Nations, the World Trade Organization, and the International Atomic Energy Agency, as well as regional or policy-based organizations such as the Association of Southeast Asian Nations (ASEAN) or the Organization of Petroleum Exporting Countries (OPEC).

The data used for this analysis is presented in the table below. The first two columns identify the ISO-3000 code identifying each country. Alliances are recorded as being present (1) or absent (0), and the overlap of IGO membership is recorded as a count value bounded below by zero.

## # A tibble: 1,586 × 4  
## statea stateb alliance\_present igo\_overlap  
## <chr> <chr> <int> <int>  
## 1 AZE ARM 1 33  
## 2 BFA BEN 1 67  
## 3 BOL ARG 1 63  
## 4 BRA ARG 1 73  
## 5 BRA BOL 1 64  
## 6 CHE AUT 0 74  
## 7 CHL ARG 1 73  
## 8 CHL BOL 1 63  
## 9 CHN AFG 0 27  
## 10 CHN AGO 0 29  
## # ... with 1,576 more rows

### 1.4.4: Logistic regression of alliance data

States which share membership in many of the same IGOs are likely to have similar policy preferences, regional concerns, and economic status that lead to their choosing to join these organizations. If we believe that similarity breeds familiarity and lowers barriers to cooperation (similar to the 'birds of a feather' argument), then we can generate testable expectations about how shared IGO membership relates to the probability of forming an alliance between states. Specifically, we hypothesize that as shared IGO membership between a pair of states increases, the probability that these states also share a military alliance will increase as well.

We can test this hypothesis by fitting another logistic model in R using the glm() function.

alliance\_model <- glm(  
 alliance\_present ~ igo\_overlap  
 , data = alliance\_data  
 , family = binomial(link = 'logit')  
 )

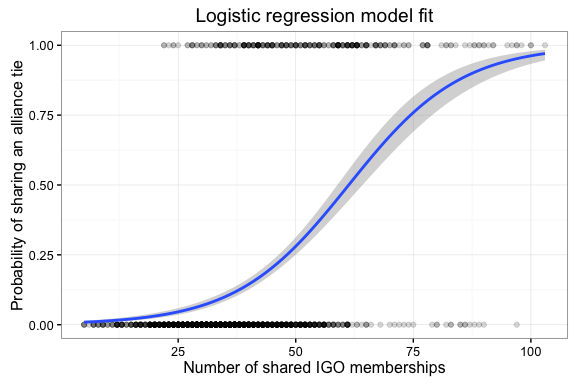
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| **igo\_overlap** | 0.08358 | 0.005461 | 15.3 | 7.2e-53 |
| **(Intercept)** | -5.121 | 0.2617 | -19.57 | 2.937e-85 |

(Dispersion parameter for binomial family taken to be 1 )

|  |  |
| --- | --- |
| Null deviance: | 1497 on 1585 degrees of freedom |
| Residual deviance: | 1156 on 1584 degrees of freedom |

The results of the logistic regression suggest that there is a positive relationship between the number of IGO memberships a pair of states share and the likelihood that they also share an alliance. This relationship is significant at the level, meaning that we can reject the null hypothesis with a high level of confidence.

Remember that logistic regression models can produce estimated probabilities of observing a 1 versus a 0 based on a given set of input values. This is a useful way of visualizing how well a model fits the observed data. Here, we produce a set of predicted probabilities (bounded between 0 and 1) that an alliance will be present between each pair of states based on their IGO membership overlap, and overlay this trend line on the scatter plot of 0 and 1 values present in the data.



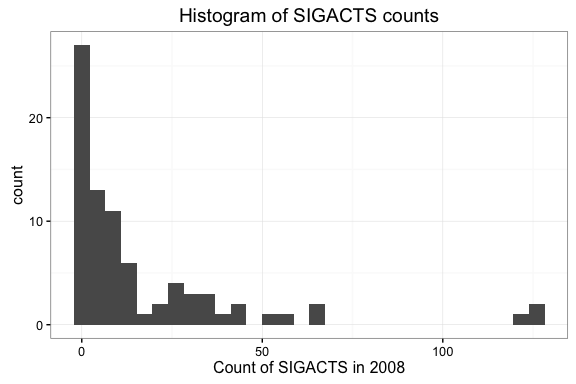
Visualizing the predicted probability estimates shows that the model does a moderately good job of separating out 0's and 1's based on the inputs used. IGO membership is certainly not the only factor that may explain how states form alliances with one another, but it provides a useful starting point for modeling.

## Section 1.5: Poisson regression

### 1.5.1: Introducing SIGACTS data

As discussed earlier in the chapter, the regional SIGACTS data recorded in the Philippines are count data, meaning they take only integer values and are bounded below by zero. Visualizing count data in a histogram is a useful way of assessing how the data are distributed.

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Visualizing the data in a histogram we observe that they appear be Poisson-distributed, which is common in count data.

### 1.5.2: Poisson regression of SIGACTS data

Poisson regression in R is also treated as a special case of GLMs, similar to the logistic regression covered in the previous section. As such, it can be implemented using the same glm() function, but now specifying the model family as 'poisson', which tells R to implement a Poisson model. The model we use here can be specified as

## Generate model  
sigacts\_model <- glm(  
 sigacts\_2008 ~ ggi\_2008 + literacy + poverty  
 , data = sigacts\_data  
 , family = poisson  
)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| **ggi\_2008** | -0.0136 | 0.001475 | -9.22 | 2.973e-20 |
| **literacy** | -0.02098 | 0.005091 | -4.12 | 3.79e-05 |
| **poverty** | 0.02297 | 0.002214 | 10.37 | 3.265e-25 |
| **(Intercept)** | 5.288 | 0.4665 | 11.34 | 8.755e-30 |

(Dispersion parameter for poisson family taken to be 1 )

|  |  |
| --- | --- |
| Null deviance: | 2358 on 79 degrees of freedom |
| Residual deviance: | 1852 on 76 degrees of freedom |

Note that Poisson models generate log-odds estimates. This means that we can readily convert coefficient estimates to odds ratios, indicating the impact that a one-unit change in a given input variable will have on the estimated number of events. When interpreting odds ratios, remember that an odds ratio above 1.0 indicates that increasing the input variable increase the estimated event count, while odds ratios lower than 1.0 indicate that increasing the input variable will lower the estimated event count.

* exp(-0.0136) = 0.986. This means that increasing the value of government satistfaction by one unit will lower the expected level of violence by about 1.4%.
* exp(-0.02098) = 0.979. This means that increasing the value of literacy by one unit will lower the expected level of violence by about 2.1%.
* exp(0.02297) = 1.023. This means that increasing the value of poverty by one unit will increase the expected level of violence by about 1.02%.

These relationships are all in the direction we would intuitively expect: higher literacy and greater satisfaction with the government should certainly be associated with lower levels of anti-government violence, while greater poverty may drive discontent and disorder, including violent acts. However, only the estimated coefficients on government satisfaction and literacy are statistically significant; for poverty, we cannot reject the null hypothesis at .