Lab7 - Introduction to linear regression

Michael Y.

April 14th, 2019

Batter up

The movie Moneyball focuses on the "quest for the secret of success in baseball". It follows a low-budget team, the Oakland Athletics, who believed that underused statistics, such as a player's ability to get on base, better predict the ability to score runs than typical statistics like home runs, RBIs (runs batted in), and batting average. Obtaining players who excelled in these underused statistics turned out to be much more affordable for the team.

In this lab we'll be looking at data from all 30 Major League Baseball teams and examining the linear relationship between runs scored in a season and a number of other player statistics. Our aim will be to summarize these relationships both graphically and numerically in order to find which variable, if any, helps us best predict a team's runs scored in a season.

The data

Let's load up the data for the 2011 season.

```
load("more/mlb11.RData")
```

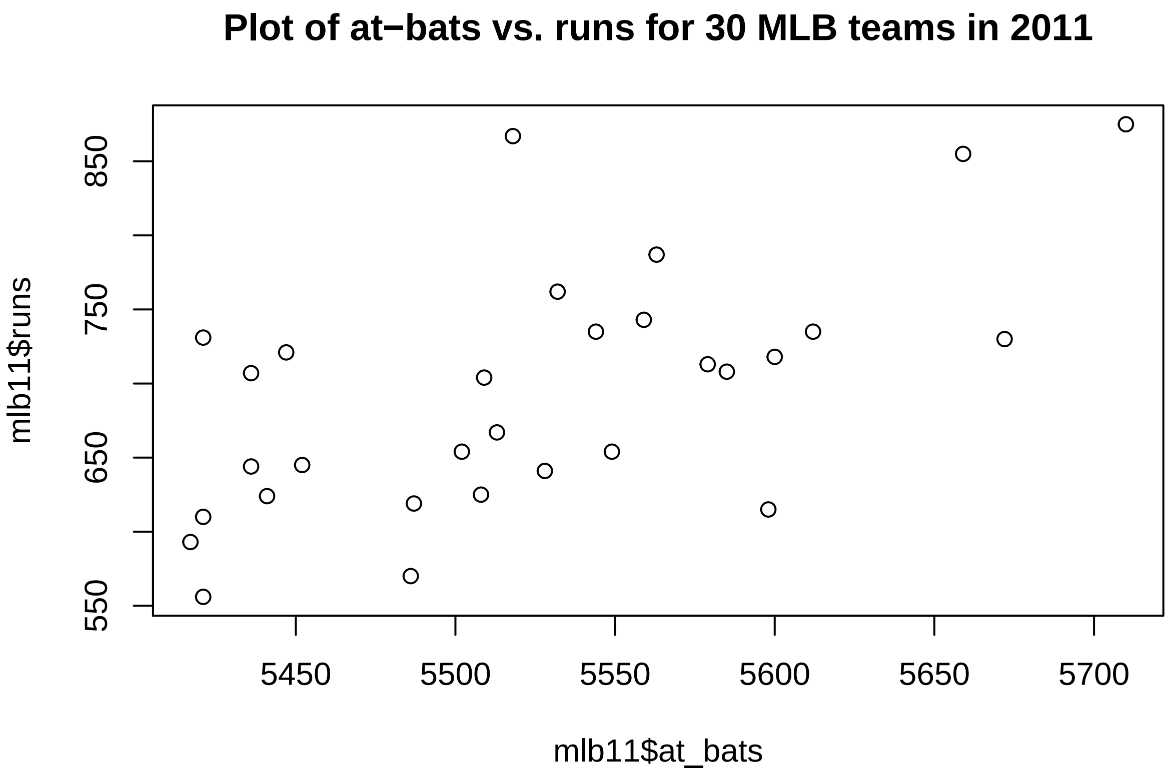
In addition to runs scored, there are seven traditionally used variables in the data set: at-bats, hits, home runs, batting average, strikeouts, stolen bases, and wins. There are also three newer variables: on-base percentage, slugging percentage, and on-base plus slugging. For the first portion of the analysis we'll consider the seven traditional variables. At the end of the lab, you'll work with the newer variables on your own.

1. What type of plot would you use to display the relationship between `runs` and one of the other numerical variables? Plot this relationship using the variable `at_bats` as the predictor. Does the relationship look linear? If you knew a team's `at_bats`, would you be comfortable using a linear model to predict the number of runs?

Scatterplot.

Plot this relationship using the variable `at_bats` as the predictor.

```
plot(x = mlb11$at_bats, y=mlb11$runs)
title(main="Plot of at-bats vs. runs for 30 MLB teams in 2011")
```



Does the relationship look linear?

It's difficult to discern visually whether or not there is a linear relationship because the fit is not very tight.

If you knew a team's `at_bats`, would you be comfortable using a linear model to predict the number of runs?

No, I don't believe that the relationship between these two variables provides a strong enough prediction to achieve the necessary level of comfort.

If the relationship looks linear, we can quantify the strength of the relationship with the correlation coefficient.

```
cor(mlb11$runs, mlb11$at_bats)
```

```
## [1] 0.61062705
```

Here the correlation is 0.61062705 which means that the goodness-of-fit, or R-Squared, is only 0.37286539 . This is not very strong.

Sum of squared residuals

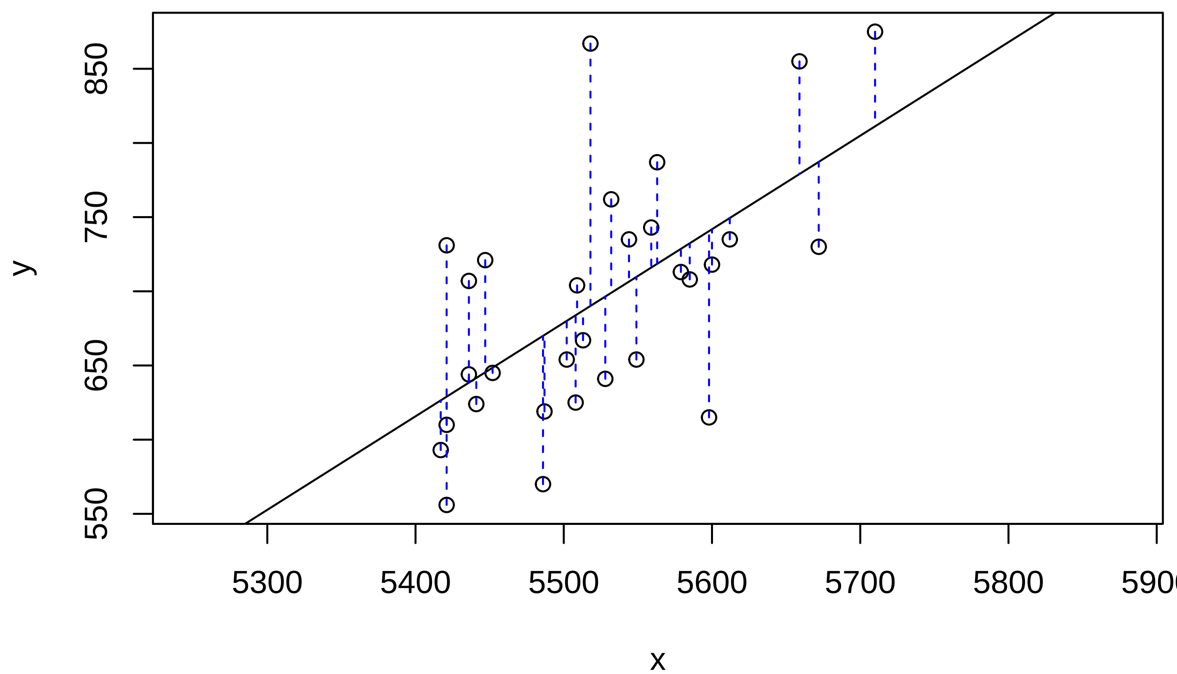
Think back to the way that we described the distribution of a single variable. Recall that we discussed characteristics such as center, spread, and shape. It's also useful to be able to describe the relationship of two numerical variables, such as `runs` and `at_bats` above.

2. Looking at your plot from the previous exercise, describe the relationship between these two variables. Make sure to discuss the form, direction, and strength of the relationship as well as any unusual observations.

The relationship appears to be somewhat linear, with an increasing relationship between at-bats and runs, but the relationship is not extremely strong.

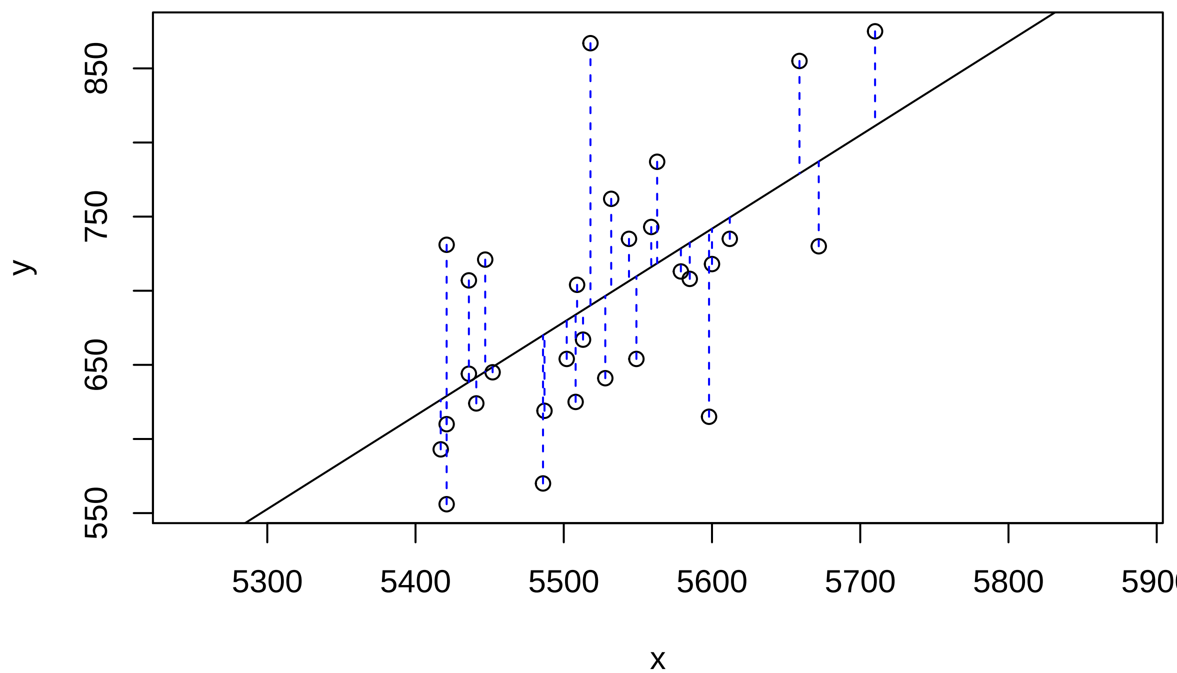
Just as we used the mean and standard deviation to summarize a single variable, we can summarize the relationship between these two variables by finding the line that best follows their association. Use the following interactive function to select the line that you think does the best job of going through the cloud of points.

```
plot_ss(x = mlb11$at_bats, y = mlb11$runs)
```




```
## Click two points to make a line.

## Call:
## lm(formula = y ~ x, data = pts)
##
## Coefficients:
## (Intercept)            x
## -2789.24289      0.63055
##
## Sum of Squares:  123721.87
```

```r
plot_ss(x = mlb11$at_bats, y = mlb11$runs
#        x1 = mean(mlb11$at_bats), y1=mean(mlb11$runs),
#        x2 = mean(mlb11$at_bats)+100, y2=mean(mlb11$runs)+100*.63055
        )
```



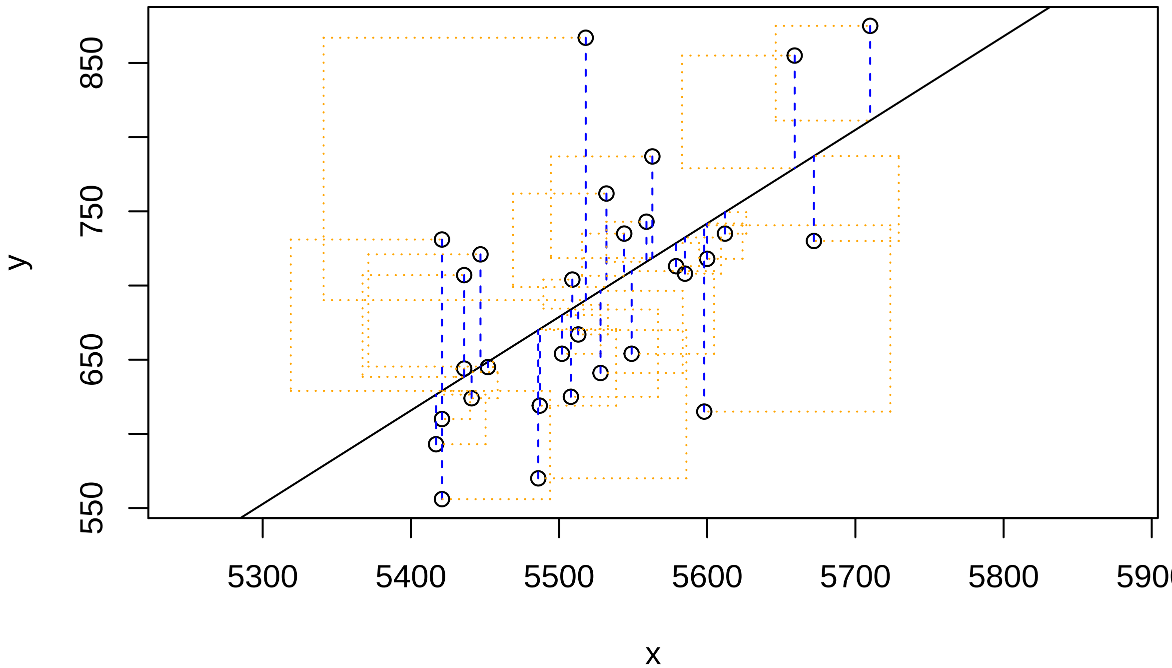

```
## Click two points to make a line.

## Call:
## lm(formula = y ~ x, data = pts)
##
## Coefficients:
## (Intercept)            x
## -2789.24289      0.63055
##
## Sum of Squares:  123721.87
```

After running this command, you'll be prompted to click two points on the plot to define a line. Once you've done that, the line you specified will be shown in black and the residuals in blue. Note that there are 30 residuals, one for each of the 30 observations. Recall that the residuals are the difference between the observed values and the values predicted by the line:

$$e_i = y_i - \hat{y}_i$$

The most common way to do linear regression is to select the line that minimizes the sum of squared residuals. To visualize the squared residuals, you can rerun the plot command and add the argument `showSquares = TRUE`.

```r
plot_ss(x = mlb11$at_bats, y = mlb11$runs, showSquares = TRUE)
```




```
## Click two points to make a line.

## Call:
## lm(formula = y ~ x, data = pts)
##
## Coefficients:
## (Intercept)            x
## -2789.24289      0.63055
##
## Sum of Squares:  123721.87
```

Note that the output from the `plot_ss` function provides you with the slope and intercept of your line as well as the sum of squares.

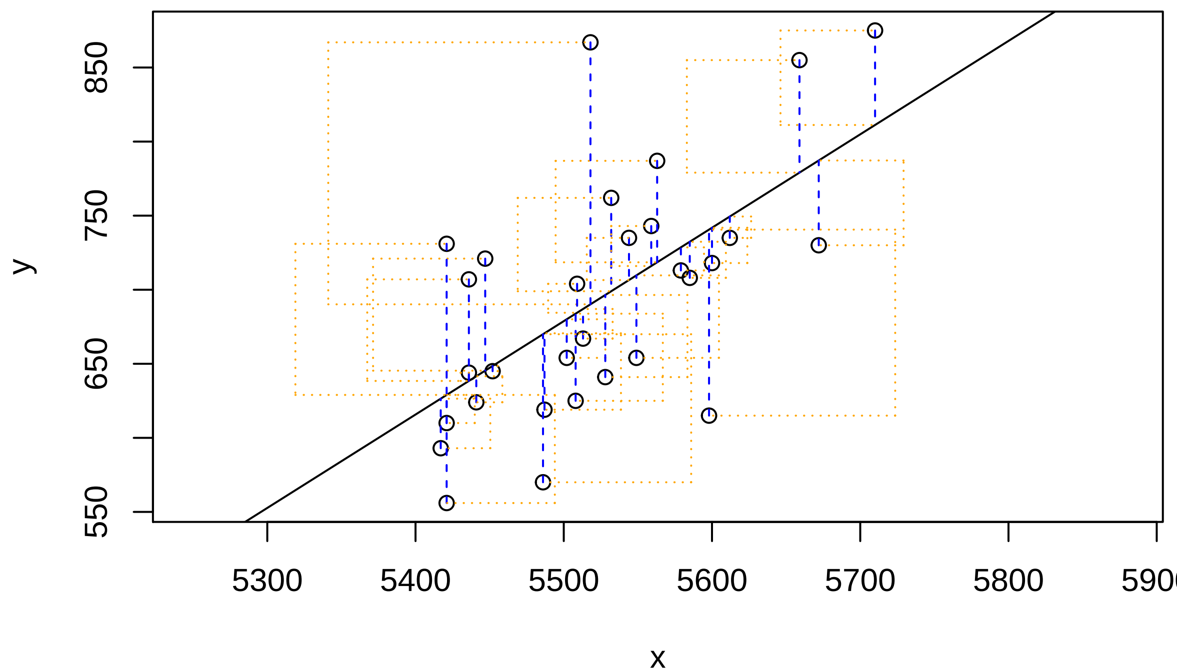
3. Using `plot_ss`, choose a line that does a good job of minimizing the sum of squares. Run the function several times. What was the smallest sum of squares that you got? How does it compare to your neighbors?

The regression line must pass through the point [mean(x), mean(y)].

Here, those figures are [5523.5, 693.6]

Knowing that the slope of the regression line is about +0.63 , we can select another point by increasing x by 100 and y by 63. This gives a residual sum of squares equal to 123721.9 . The best possible result is a tiny bit smaller:

```
plot_ss(x = mlb11$at_bats, y = mlb11$runs, showSquares = TRUE, leastSquares = T)
```




```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
## -2789.24289      0.63055
##
## Sum of Squares:  123721.87
```

The linear model

It is rather cumbersome to try to get the correct least squares line, i.e. the line that minimizes the sum of squared residuals, through trial and error. Instead we can use the `lm` function in R to fit the linear model (a.k.a. regression line).

```
m1 <- lm(runs ~ at_bats, data = mlb11)
```

The first argument in the function `lm` is a formula that takes the form `y ~ x`. Here it can be read that we want to make a linear model of `runs` as a function of `at_bats`. The second argument specifies that R should look in the mlb11 data frame to find the runs and at_bats variables.

The output of lm is an object that contains all of the information we need about the linear model that was just fit. We can access this information using the summary function.

```r
summary(m1)
```

```
##
## Call:
## lm(formula = runs ~ at_bats, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.576  -47.050  -16.588   54.399  176.868
##
## Coefficients:
##                  Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) -2789.24289   853.69572 -3.2673 0.0028706 **
## at_bats         0.63055     0.15454  4.0801 0.0003388 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66.473 on 28 degrees of freedom
## Multiple R-squared:  0.37287,    Adjusted R-squared:  0.35047
## F-statistic: 16.648 on 1 and 28 DF,  p-value: 0.00033884
```

Let's consider this output piece by piece. First, the formula used to describe the model is shown at the top. After the formula you find the five-number summary of the residuals. The "Coefficients" table shown next is key; its first column displays the linear model's y-intercept and the coefficient of at_bats. With this table, we can write down the least squares regression line for the linear model:

$$\hat{y} = -2789.2429 + 0.6305 * atbats$$

One last piece of information we will discuss from the summary output is the Multiple R-squared, or more simply, R^2. The R^2 value represents the proportion of variability in the response variable that is explained by the explanatory variable. For this model, 37.3% of the variability in runs is explained by at-bats.

4. Fit a new model that uses homeruns to predict runs. Using the estimates from the R output, write the equation of the regression line. What does the slope tell us in the context of the relationship between success of a team and its home runs?

```r
m2 <- lm(formula = runs ~ homeruns, data=mlb11)
summary(m2)
```

```
##
## Call:
## lm(formula = runs ~ homeruns, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -91.6146 -33.4096   3.2308  24.2925 104.6306
##
## Coefficients:
##               Estimate Std. Error t value        Pr(>|t|)
## (Intercept) 415.23888   41.67789  9.9630 0.0000000001044 ***
## homeruns      1.83454    0.26765  6.8541 0.0000001900086 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.295 on 28 degrees of freedom
## Multiple R-squared:  0.62656,    Adjusted R-squared:  0.61323
## F-statistic: 46.979 on 1 and 28 DF,  p-value: 0.00000019001
```

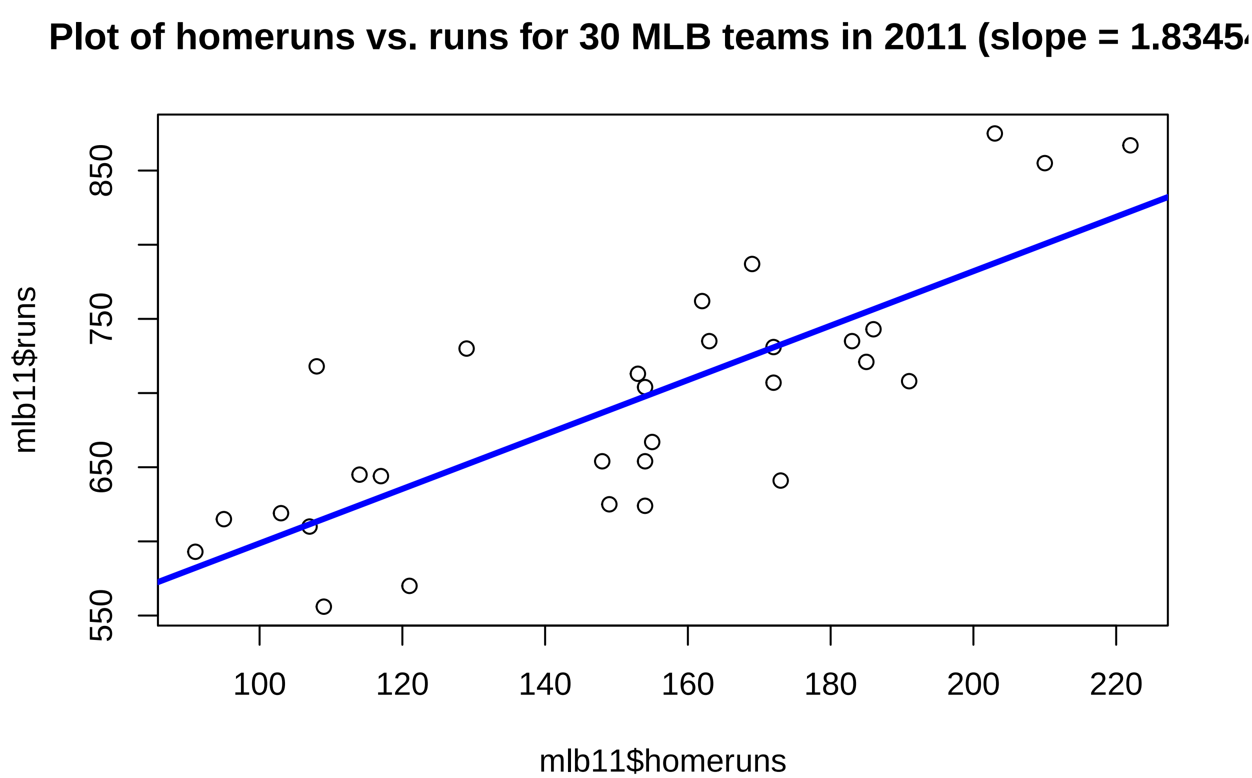
Using the estimates from the R output, write the equation of the regression line.

$$\hat{runs} = 415.23888 + 1.83454 * homeruns$$

What does the slope tell us in the context of the relationship between success of a team and its home runs?

The slope tells us that that the number of runs scored by a team increases by 1.83454 with each homerun. Of course, this coefficient must be at least 1, because a solo HR (with no base runners) would score a single run (i.e., the batter), while homeruns hit with runners on base would result in 2, 3, or in the case of a "grand slam", 4 runs.

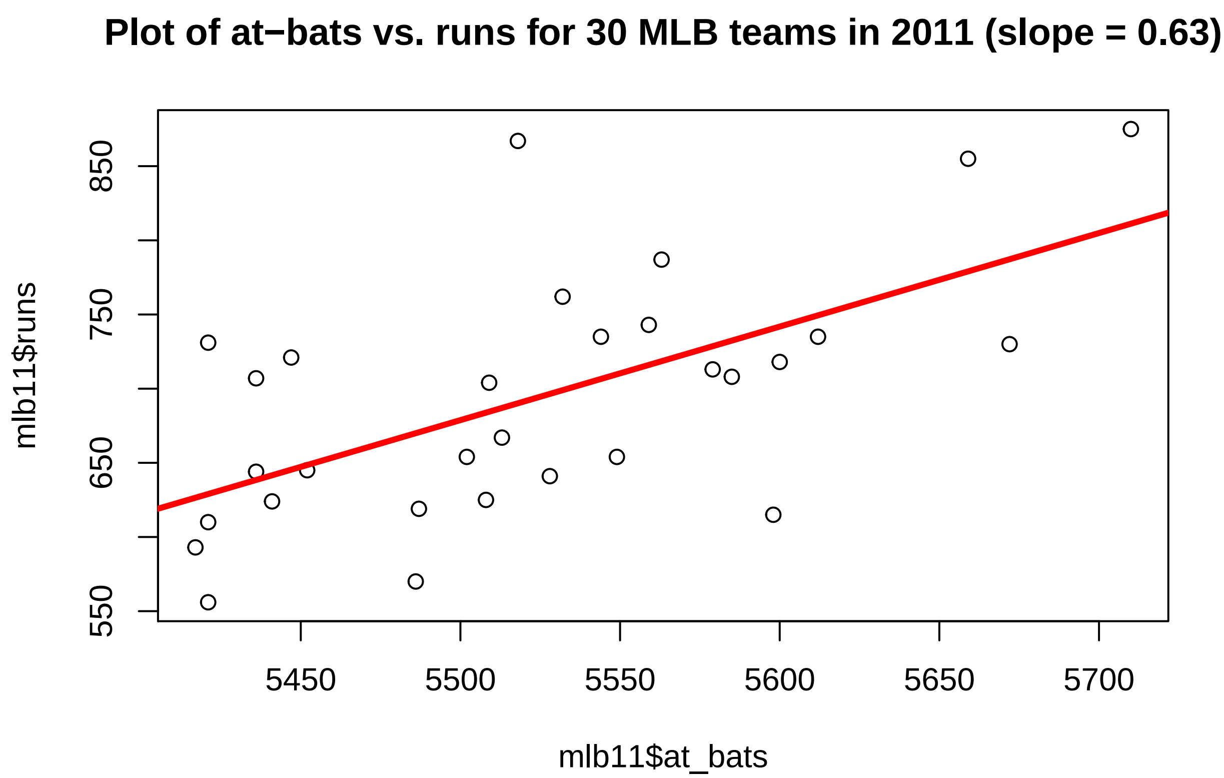
```
plot(mlb11$runs ~ mlb11$homeruns)
abline(m2, col="blue", lwd=3)
title(main="Plot of homeruns vs. runs for 30 MLB teams in 2011 (slope = 1.83454)")
```



Prediction and prediction errors

Let's create a scatterplot with the least squares line laid on top.

```
plot(mlb11$runs ~ mlb11$at_bats)
abline(m1, col="red", lwd=3)
title(main="Plot of at-bats vs. runs for 30 MLB teams in 2011 (slope = 0.63)")
```



The function `abline` plots a line based on its slope and intercept. Here, we used a shortcut by providing the model `m1`, which contains both parameter estimates. This line can be used to predict y at any value of x. When predictions are made for values of x that are beyond the range of the observed data, it is referred to as *extrapolation* and is not usually recommended. However, predictions made within the range of the data are more reliable. They're also used to compute the residuals.

5. If a team manager saw the least squares regression line and not the actual data, how many runs would he or she predict for a team with 5,578 at-bats? Is this an overestimate or an underestimate, and by how much? In other words, what is the residual for this prediction?

```
### what is the predicted number of runs for a team with 5578 at-bats?
pred_5578 <- m1$coefficients[1] + 5578 * m1$coefficients[2]
pred_5578
```

```
## (Intercept)
##   727.96497
```

```
### Is there any team with 5578 at-bats?
mlb11$at_bats==5578
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
sum(mlb11$at_bats==5578)
```

```
## [1] 0
```

```r
mlb11[mlb11$at_bats==5578,]
```

```
##  [1] team         runs         at_bats      hits         homeruns     bat_avg      strikeouts   stol
## [11] new_slug     new_obs
## <0 rows> (or 0-length row.names)
```

```r
### No, there is no such team.  Perhaps 5578 was a typo?

### What is the closest number of at-bats to 5578?
sort(mlb11$at_bats)
```

```
##  [1] 5417 5421 5421 5421 5436 5436 5441 5447 5452 5486 5487 5502 5508 5509 5513 5518 5528 5532 5544 5
## [28] 5659 5672 5710
```

```r
### OK, there is a team with 5579 at-bats. Perhaps this is the figure that was intended?
mlb11$at_bats==5579
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
### Which team is this?
mlb11[mlb11$at_bats==5579,-12]
```

```
##                     team runs at_bats hits homeruns bat_avg strikeouts stolen_bases wins new_onbase n
## 16 Philadelphia Phillies  713    5579 1409      153   0.253       1024           96  102      0.323
```

```r
### OK, Philadelphia, which is tagged as the 16th row of the dataframe:
head(mlb11[order(mlb11$at_bats,decreasing = T),-12],10)
```

```
##                     team runs at_bats hits homeruns bat_avg strikeouts stolen_bases wins new_onbase n
## 2         Boston Red Sox  875    5710 1600      203   0.280       1108          102   90      0.349
## 4     Kansas City Royals  730    5672 1560      129   0.275       1006          153   71      0.329
## 1          Texas Rangers  855    5659 1599      210   0.283        930          143   96      0.340
## 14        Cincinnati Reds  735    5612 1438      183   0.256       1250           97   79      0.326
## 6          New York Mets  718    5600 1477      108   0.264       1085          130   77      0.335
## 10        Houston Astros  615    5598 1442       95   0.258       1164          118   56      0.311
## 11     Baltimore Orioles  708    5585 1434      191   0.257       1120           81   69      0.316
## 16 Philadelphia Phillies  713    5579 1409      153   0.253       1024           96  102      0.323
## 3         Detroit Tigers  787    5563 1540      169   0.277       1143           49   95      0.340
## 20     Toronto Blue Jays  743    5559 1384      186   0.249       1184          131   81      0.317
```

```r
### What is the predicted number of runs for a team with 5579 at-bats?
pred_5579 <- m1$coefficients[1] + 5579 * m1$coefficients[2]
pred_5579
```

```
## (Intercept)
##   728.59552
```

```r
### How many runs did this team actually score?
actual_5579 <- mlb11[mlb11$at_bats==5579,'runs']
actual_5579
```

```
## [1] 713
```

```
### What is the residual?
residual_5579 <-  actual_5579 - pred_5579
residual_5579
```

```
## (Intercept)
##  -15.595525
```

The predicted number of runs for a team with 5,578 at-bats would be 727.96497461 .

However, there is no team in the data with exactly 5,578 at-bats. The closest is Philadelphia, which had 5,579.

(Perhaps 5,578 was a typo, and 5,579 was intended?)

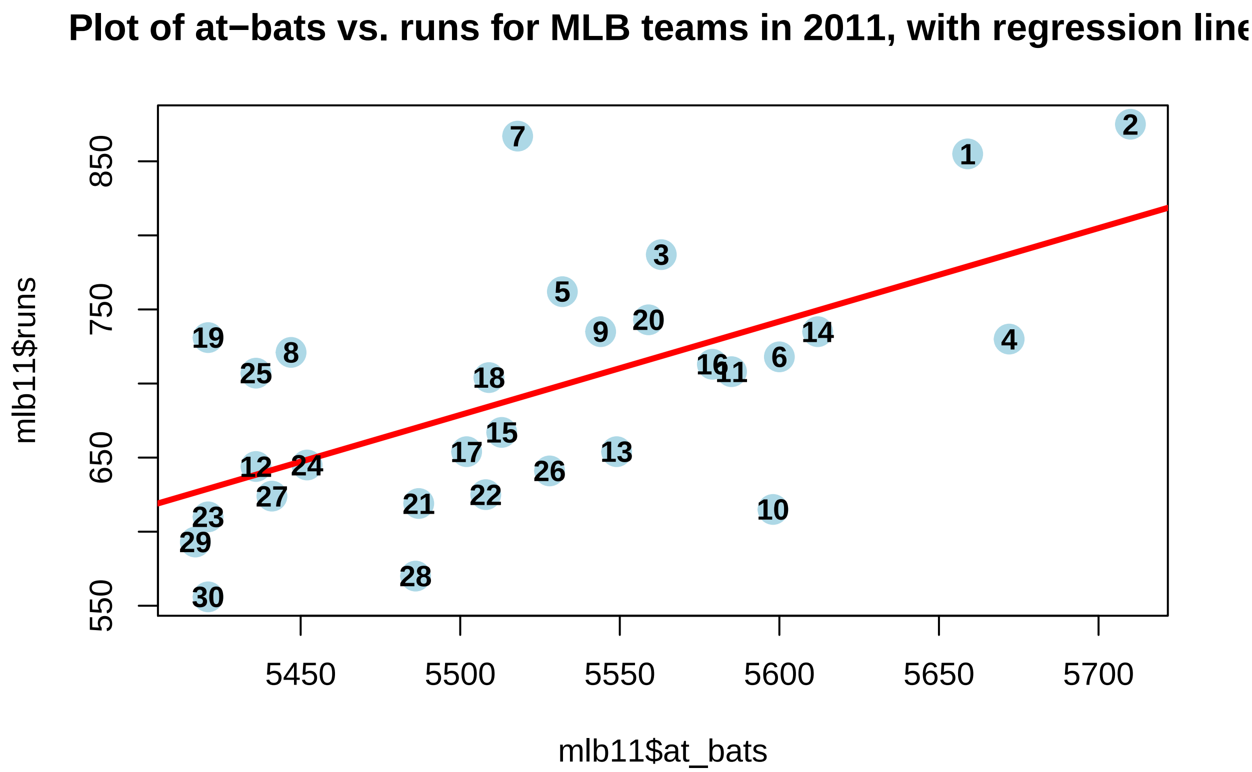
If so, the predicted number of runs for a team with 5,579 at-bats would be 728.5955246 .

This is an *overestimate*, as the actual number of runs scored by this team was 713 .

Accordingly, the residual is negative: *-15.5955246* .

The graph below illustrates that the team numbered 16 (Philadelphia) appears just below the regression line (near the center of the graph):

```
plot(y=mlb11$runs, x=mlb11$at_bats,  col="lightblue", pch=19, cex=2)
abline(m1, col="red", lwd=3)
text(runs~at_bats, labels=rownames(mlb11),data=mlb11, cex=0.9, font=2)
title(main="Plot of at-bats vs. runs for MLB teams in 2011, with regression line")
```



Interestingly, Philadelphia won the most games during the regular season:

```
head(mlb11[order(mlb11$wins,decreasing = T),-12],10)
```

```
##                       team runs at_bats hits homeruns bat_avg strikeouts stolen_bases wins new_onbase r
## 16 Philadelphia Phillies  713    5579 1409      153   0.253       1024           96  102      0.323
## 7        New York Yankees  867    5518 1452      222   0.263       1138          147   97      0.343
## 1          Texas Rangers  855    5659 1599      210   0.283        930          143   96      0.340
## 8      Milwaukee Brewers  721    5447 1422      185   0.261       1083           94   96      0.325
## 3         Detroit Tigers  787    5563 1540      169   0.277       1143           49   95      0.340
## 19  Arizona Diamondbacks  731    5421 1357      172   0.250       1249          133   94      0.322
## 25        Tampa Bay Rays  707    5436 1324      172   0.244       1193          155   91      0.322
## 2         Boston Red Sox  875    5710 1600      203   0.280       1108          102   90      0.349
## 5    St. Louis Cardinals  762    5532 1513      162   0.273        978           57   90      0.341
## 26        Atlanta Braves  641    5528 1345      173   0.243       1260           77   89      0.308
```

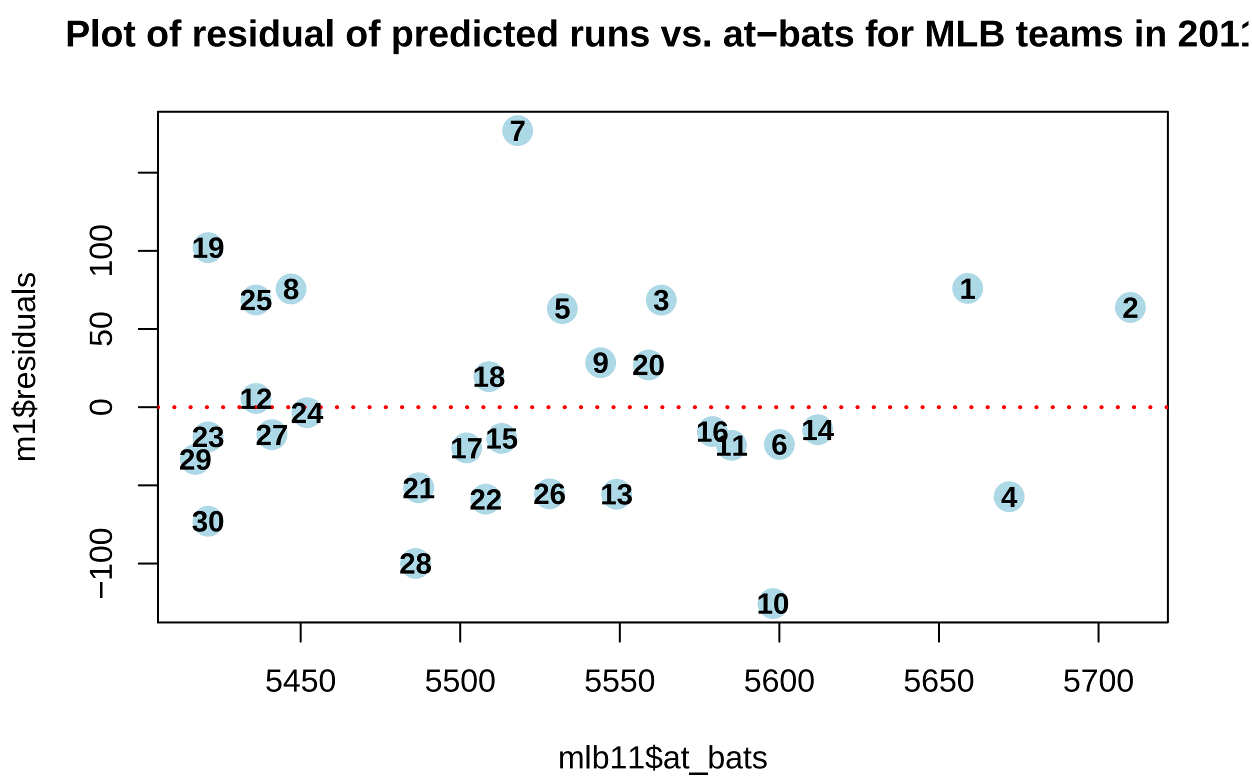
However, in postseason play, Philadelphia lost their NL Division Series to St. Louis, a Wild-Card team which would go on to win the 2011 World Series.

Model diagnostics

To assess whether the linear model is reliable, we need to check for (1) linearity, (2) nearly normal residuals, and (3) constant variability.

Linearity: You already checked if the relationship between runs and at-bats is linear using a scatterplot. We should also verify this condition with a plot of the residuals vs. at-bats. Recall that any code following a # is intended to be a comment that helps understand the code but is ignored by R.

```
plot(m1$residuals ~ mlb11$at_bats,  col="lightblue", pch=19, cex=2)
abline(h = 0, lty = 3, col="red", lwd=2)  # adds a horizontal dashed line at y = 0
text(m1$residuals~mlb11$at_bats, labels=rownames(mlb11),data=mlb11, cex=0.9, font=2)
title(main="Plot of residual of predicted runs vs. at-bats for MLB teams in 2011")
```



6. Is there any apparent pattern in the residuals plot? What does this indicate about the linearity of the relationship between runs and at-bats?

No, there does not appear to be any pattern in the residuals plot.

However, I note that there are more teams "below the line" (i.e., where runs have been over-predicted) vs. teams "above the line" (where teams have "exceeded expectatons").

Notably, success of team #7 (New York Yankees), for which their residual is so large (176.868, which is 2.66 standard deviations from the expectation) is an outlier which may have imposed unusual influence on the regression. Removal of this data point may provide more reasonable expectations for the other teams.

Using a linear model for the relationship between runs and at-bats may be appropriate.

To check for linearity, we can use the "modelAssumptions" test from the lmSupport package:

```r
require(lmSupport)    ## Note:  the "S" is capitalized in the package name
```

```
## Loading required package: lmSupport
```

```
## Warning: package 'lmSupport' was built under R version 3.5.3
```

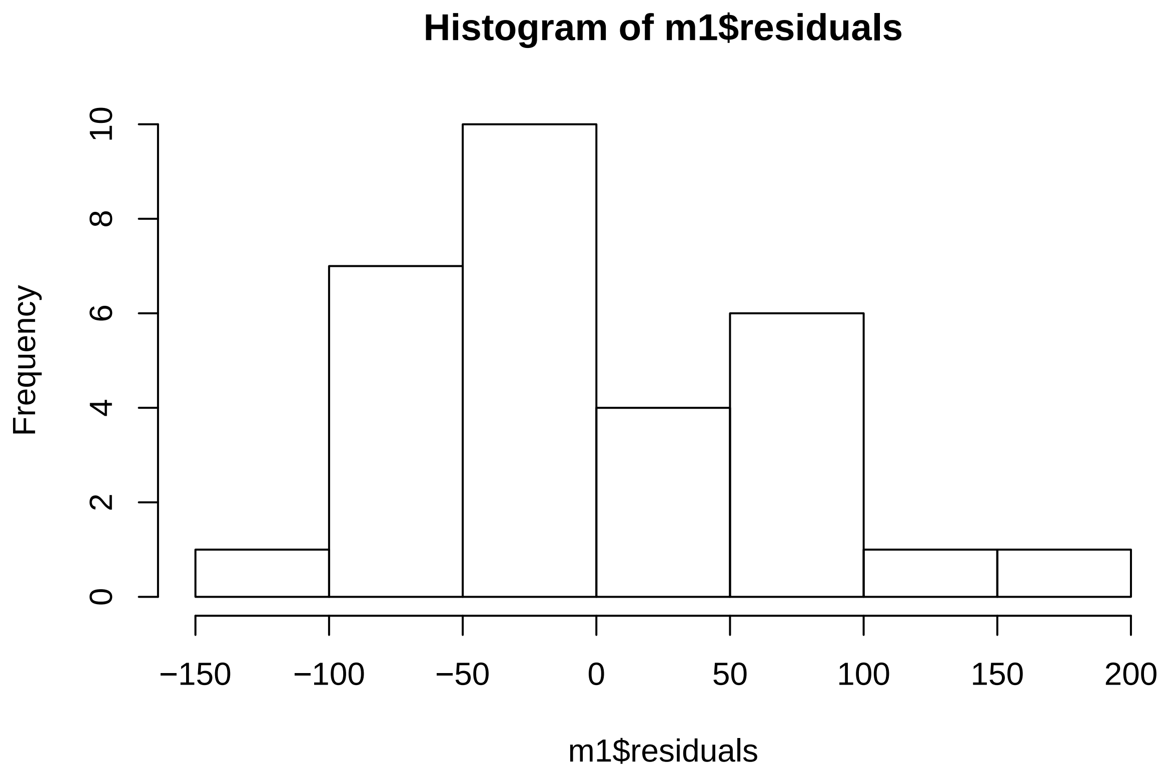
```r
modelAssumptions(m1,"LINEAR")
```

```
##
## Call:
## lm(formula = runs ~ at_bats, data = mlb11)
##
## Coefficients:
## (Intercept)      at_bats
## -2789.24289      0.63055
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = Model)
##
##                        Value p-value                Decision
## Global Stat        3.361222 0.49929 Assumptions acceptable.
## Skewness           1.479243 0.22389 Assumptions acceptable.
## Kurtosis           0.079467 0.77802 Assumptions acceptable.
## Link Function      0.508898 0.47562 Assumptions acceptable.
## Heteroscedasticity 1.293614 0.25538 Assumptions acceptable.
```

The test results are consistent with Linearity.

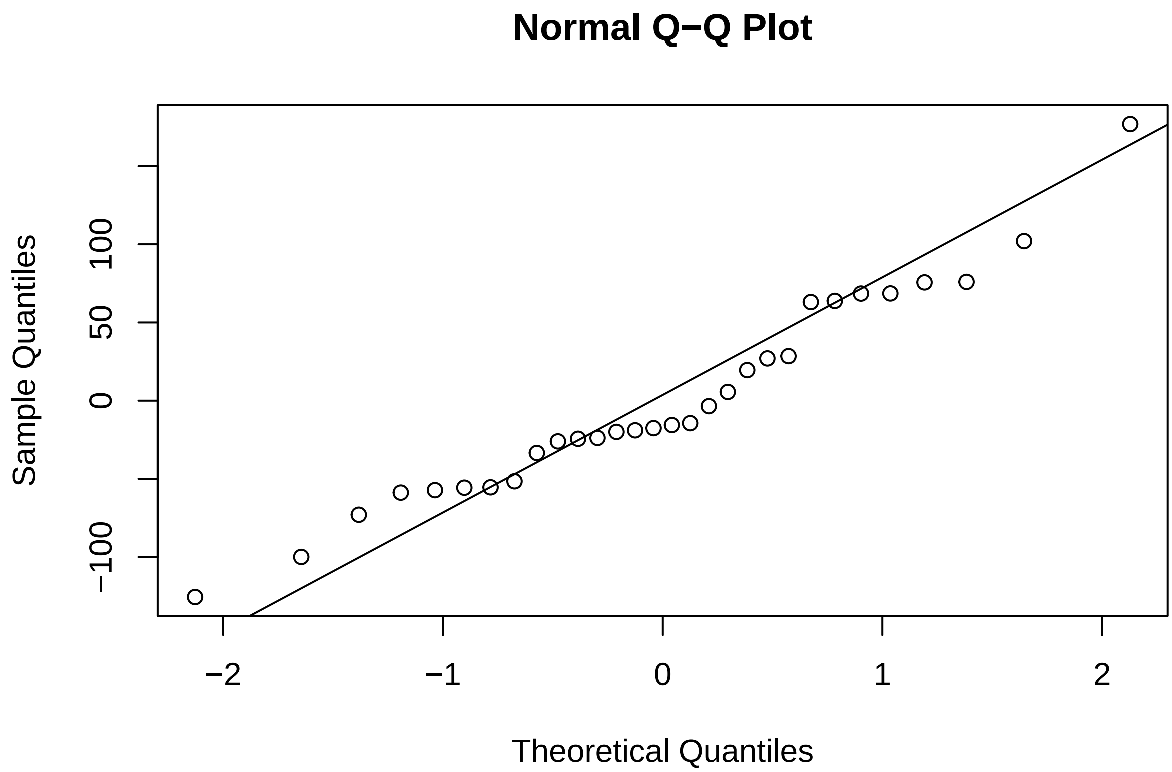
Nearly normal residuals: To check this condition, we can look at a histogram:

```r
hist(m1$residuals)
```



or a normal probability plot of the residuals:

```
qqnorm(m1$residuals)
qqline(m1$residuals)  # adds diagonal line to the normal prob plot
```



7. Based on the histogram and the normal probability plot, does the nearly normal residuals condition appear to be met?

The histogram does not appear to be "nearly normal" as the number of observations below zero (18) is greater than the number above zero (12). Likewise, the results on Q-Q-plot indicate that the empirical quantiles are not as close to the normal quantiles as would be expected.

These results would suggest that the "Nearly-normal residuals" condition may not appear to be met. However, it would be better to perform an actual test for normality, such as Shapiro-Wilks:

```
shapiro.test(m1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  m1$residuals
## W = 0.961442, p-value = 0.33703
```

Because the p-value is large, we *fail to reject* the Null Hypothesis, which is that the residuals *ARE* normal.

Another useful test for normality is Kolmogorov-Smirnov. Here we test whether the residuals are consistent with a Normal distribution with mean 0 and with standard deviation matching

that of the residuals:

```
ks.test(m1$residuals,"pnorm",0,sd(m1$residuals))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  m1$residuals
## D = 0.153934, p-value = 0.43254
## alternative hypothesis: two-sided
```

Here as well, the large p-value indicates that we *fail to reject* the null hypothesis, which is that the residuals are normal.

Another useful test of normality is the Anderson Darling test:

```
require(nortest)
```

```
## Loading required package: nortest
```

```
ad.test(m1$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  m1$residuals
## A = 0.515229, p-value = 0.17651
```

Here again, the high p-value indicates that we *fail to reject* the null hypothesis, which is that the residuals are normal.

Yet another useful test for normality is the Jarque-Bera test:

```
require(tseries)
```

```
## Loading required package: tseries
```

```
## Warning: package 'tseries' was built under R version 3.5.3
```

```
jarque.bera.test(m1$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  m1$residuals
## X-squared = 1.55871, df = 2, p-value = 0.4587
```

Here again, the high p-value indicates that we *fail to reject* the null hypothesis, which is that the residuals are normal.

Despite the questionable nature of the histogram and the QQ-plot, these numerical tests of the residuals fail to reject normality. This provides evidence that the residuals *are* normal.

Constant variability:

8. Based on the plot in (1), does the constant variability condition appear to be met?

Yes, the "constant variability" (i.e., homoscedasticity) condition does appear to be met.

A useful numeric test for constant variance is Breusch-Pagan. As it assumes that the data are normally distributed, the above tests need to have passed before we can use it.

```r
require(olsrr)
```

```
## Loading required package: olsrr

## Warning: package 'olsrr' was built under R version 3.5.3

##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
ols_test_breusch_pagan(m1)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##               Data
##  --------------------------------
##  Response : runs
##  Variables: fitted values of runs
##
##          Test Summary
##  ------------------------------
##  DF            =    1
##  Chi2          =    0.014290331
##  Prob > Chi2   =    0.90484583
```

The high p-value indicates that we fail to reject H0, which is that the variance is constant.

```r
allnames = names(mlb11)

tradvars = allnames[3:9]
numtradvars = length(tradvars)  # 7
```

On Your Own

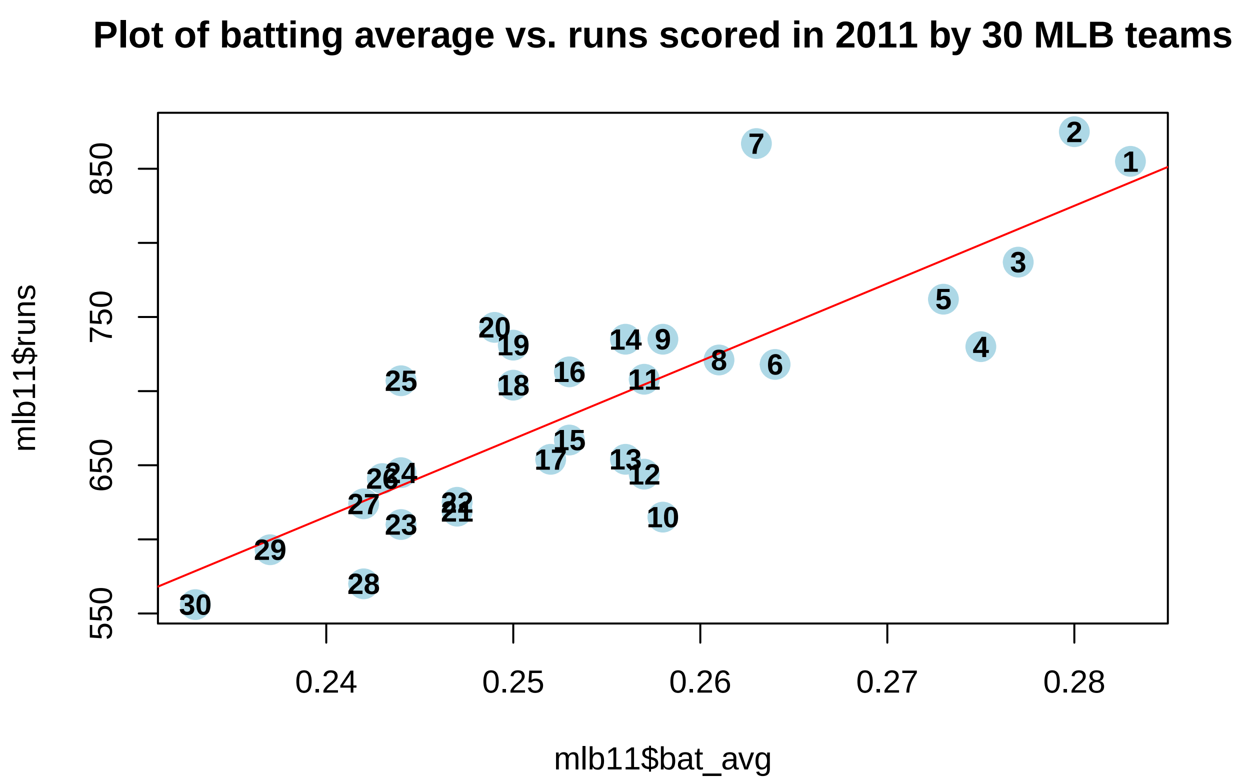
9. Choose any traditional variable from `mlb11` that you think might be a good predictor of `runs`. Produce a scatterplot of the two variables and fit a linear model. At a glance, does there seem to be a linear relationship?

The "traditional" variables include the following:

[1] "at_bats"
[2] "hits"
[3] "homeruns"
[4] "bat_avg"
[5] "strikeouts"
[6] "stolen_bases"
[7] "wins"

I choose "bat_avg" – the team batting average.

```
plot(x = mlb11$bat_avg , y = mlb11$runs, col="lightblue", pch=19, cex=2)
batavgmod <- lm(runs ~ bat_avg, data=mlb11)
abline(batavgmod, col="red")
text(runs~bat_avg, labels=rownames(mlb11),data=mlb11, cex=0.9, font=2)
title(main = "Plot of batting average vs. runs scored in 2011 by 30 MLB teams")
```



10. How does this relationship compare to the relationship between `runs` and `at_bats`? Use the R^2 values from the two model summaries to compare. Does your variable seem to predict `runs` better than `at_bats`? How can you tell?

```
print("\n\n***************BATTING AVERAGE MODEL: ")
```

```
## [1] "\n\n***************BATTING AVERAGE MODEL: "
```

```
batavgmodsumm = summary(batavgmod)
print(batavgmodsumm)
```

```
##
## Call:
## lm(formula = runs ~ bat_avg, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -94.6762 -26.3034  -5.4961  28.4822 131.1127
##
## Coefficients:
##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept)  -642.82     183.08 -3.5111      0.001531 **
## bat_avg      5242.23     717.28  7.3085 0.00000005877 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.226 on 28 degrees of freedom
## Multiple R-squared:  0.65608,    Adjusted R-squared:  0.64379
## F-statistic: 53.414 on 1 and 28 DF,  p-value: 0.00000005877
```

```
batavg_R2 <- batavgmodsumm$r.squared
batavg_AdjR2 <- batavgmodsumm$adj.r.squared

print("\n\n***************AT-BATS MODEL: ")
```

```
## [1] "\n\n***************AT-BATS MODEL: "
```

```
atbatsmodsumm = summary(m1)
print(atbatsmodsumm)
```

```
##
## Call:
## lm(formula = runs ~ at_bats, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.576  -47.050  -16.588   54.399  176.868
##
## Coefficients:
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) -2789.24289   853.69572 -3.2673 0.0028706 **
## at_bats         0.63055     0.15454  4.0801 0.0003388 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66.473 on 28 degrees of freedom
## Multiple R-squared:  0.37287,    Adjusted R-squared:  0.35047
## F-statistic: 16.648 on 1 and 28 DF,  p-value: 0.00033884
```

```
atbats_R2 <- atbatsmodsumm$r.squared
atbats_AdjR2 <- atbatsmodsumm$adj.r.squared


print(paste("Batting Average Model as predictor of runs: ", "R2: ", batavg_R2, "AdjR2: ", batavg_AdjR2
```

```
## [1] "Batting Average Model as predictor of runs:  R2:  0.656077134646863 AdjR2:  0.643794175169965"
```

```r
print(paste("At-Bats Model        as predictor of runs: ", "R2: ", atbats_R2, "AdjR2: ", atbats_AdjR2 ))
```

```
## [1] "At-Bats Model        as predictor of runs:  R2:  0.372865390186805 AdjR2:  0.350467725550619"
```

How does this relationship compare to the relationship between runs and `at_bats`?

Because the R^2 and the $Adjusted R^2$ values for the Batting Average model are much greater than those of the At-Bats model, the Batting Average Model provides a much stronger fit and is preferred as a predictor.

*****Does your variable seem to predict `runs` better than `at_bats`? How can you tell?**

```r
batavganova <- anova(batavgmod)
print(batavganova)
```

```
## Analysis of Variance Table
##
## Response: runs
##           Df   Sum Sq  Mean Sq F value        Pr(>F)
## bat_avg    1 129431.7 129431.7 53.4136 0.00000005877 ***
## Residuals 28  67849.5   2423.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
batavg_ss <- batavganova$`Sum Sq`
print(batavg_ss)
```

```
## [1] 129431.684  67849.516
```

```r
batvag_Explained_Sum_of_Squares <- batavg_ss[1]
batvag_Residual_Sum_of_Squares <- batavg_ss[2]

print(paste("batvag_Explained_Sum_of_Squares: ", batvag_Explained_Sum_of_Squares ))
```

```
## [1] "batvag_Explained_Sum_of_Squares:  129431.684415694"
```

```r
print(paste("batvag_Residual_Sum_of_Squares: ", batvag_Residual_Sum_of_Squares ))
```

```
## [1] "batvag_Residual_Sum_of_Squares:  67849.5155843053"
```

```r
atbatsanova <- anova(m1)
print(atbatsanova)
```

```
## Analysis of Variance Table
##
## Response: runs
##           Df   Sum Sq Mean Sq F value     Pr(>F)
## at_bats    1  73559.3 73559.3 16.6475 0.00033884 ***
## Residuals 28 123721.9  4418.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
atbats_ss <- atbatsanova$`Sum Sq`
print(atbats_ss)
```

```
## [1]  73559.332 123721.868
```

```r
atbats_Explained_Sum_of_Squares <- atbats_ss[1]
atbats_Residual_Sum_of_Squares <- atbats_ss[2]

print(paste("atbats_Explained_Sum_of_Squares: ", atbats_Explained_Sum_of_Squares ))
```

```
## [1] "atbats_Explained_Sum_of_Squares:  73559.3316145209"
```

```r
print(paste("atbats_Residual_Sum_of_Squares: ", atbats_Residual_Sum_of_Squares ))
```

```
## [1] "atbats_Residual_Sum_of_Squares:  123721.868385479"
```

The atbats Explained Sum of Squares is quite low, while the Residual Sum of Squares is high. This indicates that the model explains less than half of the variance.

In contrast, the batting average Explained Sum of Squared is quite high, while the residual Sum of Squares is low. This indicates that this model explains much more of the variance, and thus provides predictions which have smaller errors than the atbats model.

11. Now that you can summarize the linear relationship between two variables, investigate the relationships between `runs` and each of the other five traditional variables. Which variable best predicts `runs`? Support your conclusion using the graphical and numerical methods we've discussed (for the sake of conciseness, only include output for the best variable, not all five).

Evaluating the single-variable model for each of the seven "Traditional variables:"

```r
allnames = names(mlb11)

tradvars = allnames[3:9]
numtradvars = length(tradvars)  # 7

#newvars  = allnames[10:12]
#numnewvars = length(newvars)  # 3

#allvars = allnames[3:12]
#numallvars = length(allvars)  # 10

trad_r2    = array(data = 0, dim=numtradvars, dimnames=list(tradvars))
trad_adjr2 = array(data = 0, dim=numtradvars, dimnames=list(tradvars))

for (i in 1:numtradvars) {
  name = tradvars[i]
  print(paste ( "\n\n**********************Doing ", name, "..."))
  fmula = paste("runs ~", name)
  print(paste ( "Formula: ", fmula))
  mod = lm(fmula, data=mlb11)
  print(         "Model: ")
  print(mod)
  summod = summary(mod)
  print(         "Summod: ")
  print(summod)
  trad_r2[i]=summod$r.squared
  print(paste ( "R2: ", trad_r2[i]))
  trad_adjr2[i] = summod$adj.r.squared
```

```r
  print(paste ( "AdjR2: ", trad_adjr2[i]))

  print(paste ("************ DONE WITH ", name, "***********"))
  }
```

```
## [1] "\n\n***********************Doing  at_bats ..."
## [1] "Formula:  runs ~ at_bats"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)      at_bats
## -2789.24289      0.63055
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.576  -47.050  -16.588   54.399  176.868
##
## Coefficients:
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) -2789.24289   853.69572 -3.2673 0.0028706 **
## at_bats         0.63055     0.15454  4.0801 0.0003388 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66.473 on 28 degrees of freedom
## Multiple R-squared:  0.37287,    Adjusted R-squared:  0.35047
## F-statistic: 16.648 on 1 and 28 DF,  p-value: 0.00033884
##
## [1] "R2:  0.372865390186805"
## [1] "AdjR2:  0.350467725550619"
## [1] "************ DONE WITH  at_bats ***********"
## [1] "\n\n***********************Doing  hits ..."
## [1] "Formula:  runs ~ hits"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)         hits
##  -375.55997      0.75886
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
```

```
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -103.7183  -27.1794   -5.2328   19.3224  140.6931
##
## Coefficients:
##               Estimate Std. Error t value     Pr(>|t|)
## (Intercept) -375.55997  151.18056 -2.4842      0.01924 *
## hits           0.75886    0.10711  7.0851 0.0000001043 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.228 on 28 degrees of freedom
## Multiple R-squared:  0.64194,    Adjusted R-squared:  0.62915
## F-statistic: 50.199 on 1 and 28 DF,  p-value: 0.00000010432
##
## [1] "R2:  0.641938767239419"
## [1] "AdjR2:  0.629150866069399"
## [1] "************ DONE WITH  hits ***********"
## [1] "\n\n***********************Doing  homeruns ..."
## [1] "Formula:  runs ~ homeruns"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)     homeruns
##    415.2389       1.8345
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -91.6146 -33.4096   3.2308  24.2925 104.6306
##
## Coefficients:
##              Estimate Std. Error t value        Pr(>|t|)
## (Intercept) 415.23888   41.67789  9.9630 0.0000000001044 ***
## homeruns      1.83454    0.26765  6.8541 0.0000001900086 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.295 on 28 degrees of freedom
## Multiple R-squared:  0.62656,    Adjusted R-squared:  0.61323
## F-statistic: 46.979 on 1 and 28 DF,  p-value: 0.00000019001
##
## [1] "R2:  0.626563569566283"
## [1] "AdjR2:  0.61322655419365"
## [1] "************ DONE WITH  homeruns ***********"
## [1] "\n\n***********************Doing  bat_avg ..."
```

```
## [1] "Formula:  runs ~ bat_avg"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)      bat_avg
##     -642.82      5242.23
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -94.6762 -26.3034  -5.4961  28.4822 131.1127
##
## Coefficients:
##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept)  -642.82     183.08 -3.5111      0.001531 **
## bat_avg      5242.23     717.28  7.3085 0.00000005877 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.226 on 28 degrees of freedom
## Multiple R-squared:  0.65608,    Adjusted R-squared:  0.64379
## F-statistic: 53.414 on 1 and 28 DF,  p-value: 0.00000005877
##
## [1] "R2:  0.656077134646863"
## [1] "AdjR2:  0.643794175169965"
## [1] "************ DONE WITH  bat_avg ***********"
## [1] "\n\n***********************Doing  strikeouts ..."
## [1] "Formula:  runs ~ strikeouts"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)   strikeouts
##  1054.73423     -0.31414
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -132.270  -46.948  -11.919   55.137  169.756
##
## Coefficients:
```

```
##               Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 1054.73423  151.78899  6.9487 0.0000001486 ***
## strikeouts    -0.31414    0.13148 -2.3893      0.02386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 76.502 on 28 degrees of freedom
## Multiple R-squared:  0.16936,    Adjusted R-squared:  0.13969
## F-statistic: 5.7089 on 1 and 28 DF,  p-value: 0.023856
##
## [1] "R2:  0.169357932236313"
## [1] "AdjR2:  0.139692144101895"
## [1] "************ DONE WITH  strikeouts ***********"
## [1] "\n\n***********************Doing  stolen_bases ..."
## [1] "Formula:  runs ~ stolen_bases"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
##  (Intercept)  stolen_bases
##    677.30743       0.14906
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -139.940  -62.874   10.007   38.544  182.488
##
## Coefficients:
##               Estimate Std. Error t value          Pr(>|t|)
## (Intercept)  677.30743   58.97508 11.4846 0.000000000004166 ***
## stolen_bases   0.14906    0.52109  0.2861            0.7769
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 83.817 on 28 degrees of freedom
## Multiple R-squared:  0.002914,   Adjusted R-squared:  -0.032696
## F-statistic: 0.08183 on 1 and 28 DF,  p-value: 0.77694
##
## [1] "R2:  0.00291399266657394"
## [1] "AdjR2:  -0.0326962218810485"
## [1] "************ DONE WITH  stolen_bases ***********"
## [1] "\n\n***********************Doing  wins ..."
## [1] "Formula:  runs ~ wins"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
```

```
## Coefficients:
## (Intercept)         wins
##     342.121        4.341
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -145.4498  -47.5062   -7.4819   47.3463  142.1860
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 342.1214    89.2230  3.8345 0.0006538 ***
## wins          4.3410     1.0915  3.9770 0.0004469 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 67.1 on 28 degrees of freedom
## Multiple R-squared:  0.36097,    Adjusted R-squared:  0.33815
## F-statistic: 15.816 on 1 and 28 DF,  p-value: 0.00044693
##
## [1] "R2:  0.360971179446681"
## [1] "AdjR2:  0.338148721569776"
## [1] "************ DONE WITH  wins ***********"
```

R^2 and $Adjusted - R^2$ for single-predictor models containing each of the "traditional" variables:

```r
print("R-SQUARED: ")
```

```
## [1] "R-SQUARED: "
```

```r
print(t(t(sort(trad_r2,decreasing = T))))
```

```
##                     [,1]
## bat_avg      0.6560771346
## hits         0.6419387672
## homeruns     0.6265635696
## at_bats      0.3728653902
## wins         0.3609711794
## strikeouts   0.1693579322
## stolen_bases 0.0029139927
```

```r
print("Adjusted R-SQUARED: ")
```

```
## [1] "Adjusted R-SQUARED: "
```

```r
print(t(t(sort(trad_adjr2,decreasing = T))))
```

```
##                     [,1]
## bat_avg       0.643794175
## hits          0.629150866
## homeruns      0.613226554
```

```
## at_bats       0.350467726
## wins          0.338148722
## strikeouts    0.139692144
## stolen_bases -0.032696222
```

Which variable best predicts runs?

Among the "Traditional" variables, Batting Average, Hits, and HomeRuns each provide an R^2 and an $Adjusted - R^2$ in the .60 range, which is rather strong, while At Bats and Wins provide results in the .30 range, which is only moderately strong. Strikeouts and Stolen Bases provide extremely weak predictions of runs.

The very best predictor is Batting Average . The graphical results for the model containing this variable are shown above in response to the question numbered (9) .

12. Now examine the three newer variables. These are the statistics used by the author of *Moneyball* to predict a teams success. In general, are they more or less effective at predicting runs that the old variables? Explain using appropriate graphical and numerical evidence. Of all ten variables we've analyzed, which seems to be the best predictor of `runs`? Using the limited (or not so limited) information you know about these baseball statistics, does your result make sense?

Evaluating the single-variable model for each of the three "New" variables:

```
allnames = names(mlb11)

#tradvars = allnames[3:9]
#numtradvars = length(tradvars)  # 7

newvars  = allnames[10:12]
numnewvars = length(newvars)  # 3

#allvars = allnames[3:12]
#numallvars = length(allvars)  # 10

new_r2    = array(data = 0, dim=numnewvars, dimnames=list(newvars))
new_adjr2 = array(data = 0, dim=numnewvars, dimnames=list(newvars))

for (i in 1:numnewvars) {
  name = newvars[i]
  print(paste ( "\n\n**********************Doing ", name, "..."))
  fmula = paste("runs ~", name)
  print(paste ( "Formula: ", fmula))
  mod = lm(fmula, data=mlb11)
  print(         "Model: ")
  print(mod)
  summod = summary(mod)
  print(         "Summod: ")
  print(summod)
  new_r2[i]=summod$r.squared
  print(paste ( "R2: ", new_r2[i]))
  new_adjr2[i] = summod$adj.r.squared
  print(paste ( "AdjR2: ", new_adjr2[i]))
```

```r
  print(paste ("************ DONE WITH ", name, "***********"))
  }
```

```
## [1] "\n\n**********************Doing  new_onbase ..."
## [1] "Formula:  runs ~ new_onbase"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)   new_onbase
##     -1118.4       5654.3
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -58.2700 -18.3348   3.2486  19.5203  69.0016
##
## Coefficients:
##             Estimate Std. Error t value           Pr(>|t|)
## (Intercept) -1118.42     144.48  -7.741 0.0000000196789595 ***
## new_onbase   5654.32     450.46  12.552 0.0000000000005116 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.606 on 28 degrees of freedom
## Multiple R-squared:  0.84911,    Adjusted R-squared:  0.84372
## F-statistic: 157.56 on 1 and 28 DF,  p-value: 0.00000000000051157
##
## [1] "R2:  0.849105251446139"
## [1] "AdjR2:  0.843716153283501"
## [1] "************ DONE WITH  new_onbase ***********"
## [1] "\n\n**********************Doing  new_slug ..."
## [1] "Formula:  runs ~ new_slug"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)     new_slug
##      -375.8       2681.3
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -45.4096 -18.6566  -0.9096  16.2935  52.2932
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) -375.804     68.708 -5.4696 0.00000769547969384 ***
## new_slug    2681.331    171.830 15.6046 0.00000000000000242 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.956 on 28 degrees of freedom
## Multiple R-squared:  0.89687,    Adjusted R-squared:  0.89319
## F-statistic:  243.5 on 1 and 28 DF,  p-value: 0.0000000000000024201
##
## [1] "R2:  0.896870368409638"
## [1] "AdjR2:  0.893187167281411"
## [1] "************ DONE WITH  new_slug ***********"
## [1] "\n\n***********************Doing  new_obs ..."
## [1] "Formula:  runs ~ new_obs"
## [1] "Model: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Coefficients:
## (Intercept)      new_obs
##     -686.61      1919.36
##
## [1] "Summod: "
##
## Call:
## lm(formula = fmula, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -43.4563 -13.6903   1.1646  13.9352  41.1559
##
## Coefficients:
##             Estimate Std. Error t value              Pr(>|t|)
## (Intercept) -686.614     68.925 -9.9617       0.0000000001047 ***
## new_obs     1919.364     95.695 20.0571 < 0.00000000000000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.412 on 28 degrees of freedom
## Multiple R-squared:  0.93493,    Adjusted R-squared:  0.9326
## F-statistic: 402.29 on 1 and 28 DF,  p-value: < 0.000000000000000222
##
## [1] "R2:  0.934927126351814"
## [1] "AdjR2:  0.932603095150093"
## [1] "************ DONE WITH  new_obs ***********"
```

R^2 and $Adjusted - R^2$ for single-predictor models containing each of the three "new" variables:

```r
print("R-SQUARED: ")
```

```
## [1] "R-SQUARED: "
```

```r
print(t(t(sort(new_r2,decreasing = T))))
```

```
##                  [,1]
## new_obs    0.93492713
## new_slug   0.89687037
## new_onbase 0.84910525
```

```r
print("Adjusted R-SQUARED: ")
```

```
## [1] "Adjusted R-SQUARED: "
```

```r
print(t(t(sort(new_adjr2,decreasing = T))))
```

```
##                  [,1]
## new_obs    0.93260310
## new_slug   0.89318717
## new_onbase 0.84371615
```

In general, are they more or less effective at predicting runs that the old variables? Explain using appropriate graphical and numerical evidence.

Because the "New" variables have much higher R^2 than the "Traditional" variables, they would be more effective at predicting runs. The R^2 and $Adjusted R^2$ values for each variable are given above.

Of all ten variables we've analyzed, which seems to be the best predictor of runs?

The variable new_obs ("on-base plus slugging") seems to be the best predictor, as it has the highest R^2 .

```r
new_obs_model <- lm (formula = runs ~ new_obs, data = mlb11)
summary(new_obs_model)
```

```
##
## Call:
## lm(formula = runs ~ new_obs, data = mlb11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -43.4563 -13.6903   1.1646  13.9352  41.1559
##
## Coefficients:
##             Estimate Std. Error t value              Pr(>|t|)
## (Intercept) -686.614     68.925 -9.9617       0.0000000001047 ***
## new_obs     1919.364     95.695 20.0571 < 0.00000000000000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.412 on 28 degrees of freedom
```

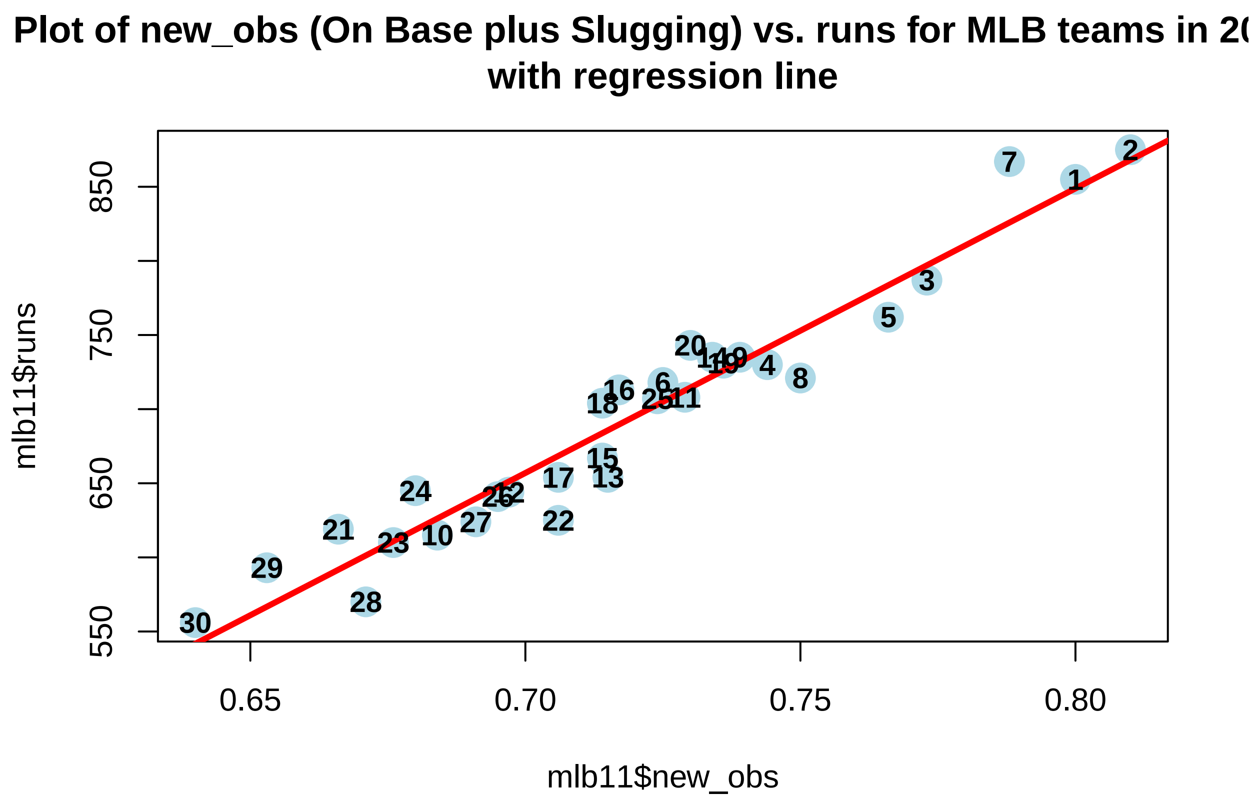
```
## Multiple R-squared:  0.93493,    Adjusted R-squared:  0.9326
## F-statistic: 402.29 on 1 and 28 DF,  p-value: < 0.0000000000000000222
```

```r
anova(new_obs_model)
```

```
## Analysis of Variance Table
##
## Response: runs
##           Df   Sum Sq  Mean Sq F value                 Pr(>F)
## new_obs    1 184443.5 184443.5 402.287 < 0.0000000000000000222 ***
## Residuals 28  12837.7    458.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As can be seen above, the R^2 results are above 90 percent, and the explained sum of squares (184443.5) far exceeds the Residual Sum of Squares (12837.7). This indicates an extremely tight fit.

```r
plot(y=mlb11$runs, x=mlb11$new_obs, col="lightblue", pch=19, cex=2)
abline(new_obs_model, col="red", lwd=3)
text(runs~new_obs, labels=rownames(mlb11),data=mlb11, cex=0.9, font=2)
title(main="Plot of new_obs (On Base plus Slugging) vs. runs for MLB teams in 2011,\nwith regression li
```



The above plot confirms an extremely tight fit, which makes for accurate predictions.

13. Check the model diagnostics for the regression model with the variable you decided was the best predictor for runs.

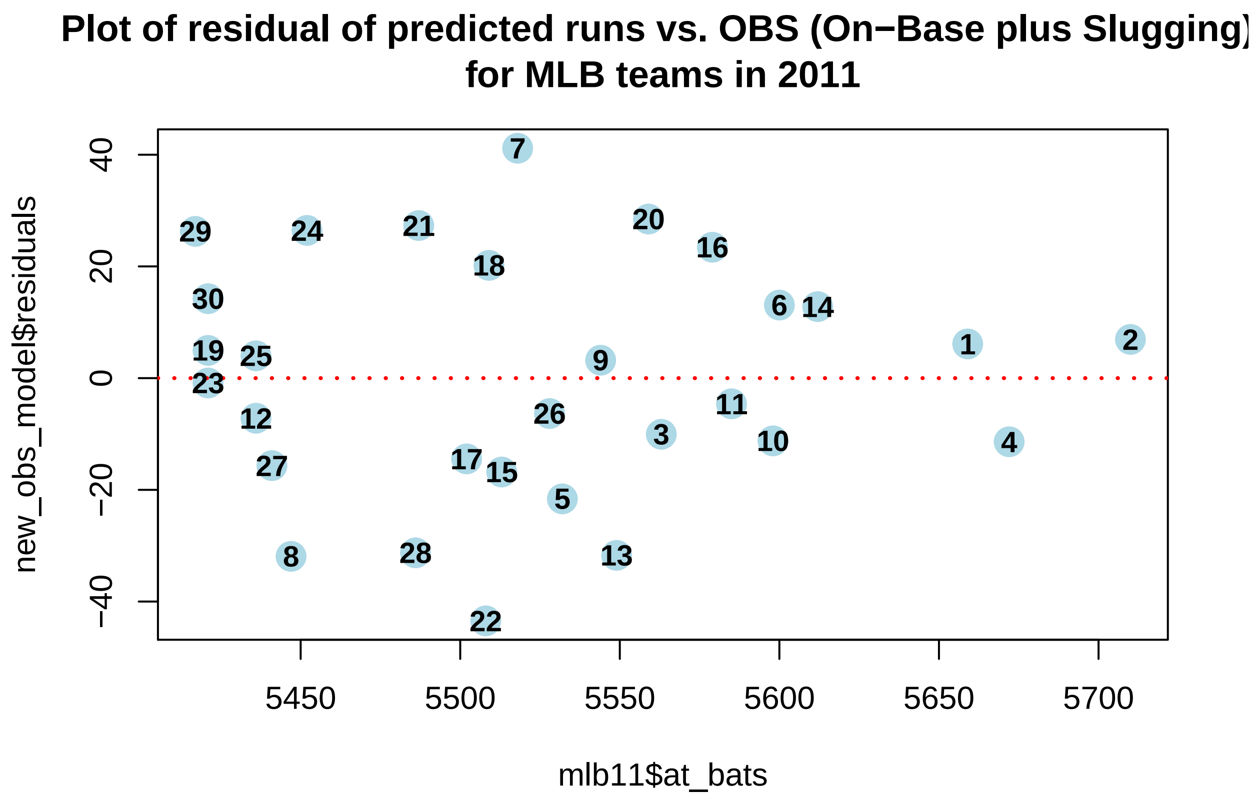
To assess whether the linear model is reliable, we need to check for

(1) linearity,

(2) nearly normal residuals, and

(3) constant variability.

```
plot(new_obs_model$residuals ~ mlb11$at_bats,  col="lightblue", pch=19, cex=2)
abline(h = 0, lty = 3, col="red", lwd=2)  # adds a horizontal dashed line at y = 0
text(new_obs_model$residuals~mlb11$at_bats, labels=rownames(mlb11),data=mlb11, cex=0.9, font=2)
title(main="Plot of residual of predicted runs vs. OBS (On-Base plus Slugging) \nfor MLB teams in 2011")
```



To check for linearity, we can use the "modelAssumptions" test from the lmSupport package:

```
require(lmSupport)    ## Note:  the "S" is capitalized in the package name
modelAssumptions(new_obs_model,"LINEAR")
```

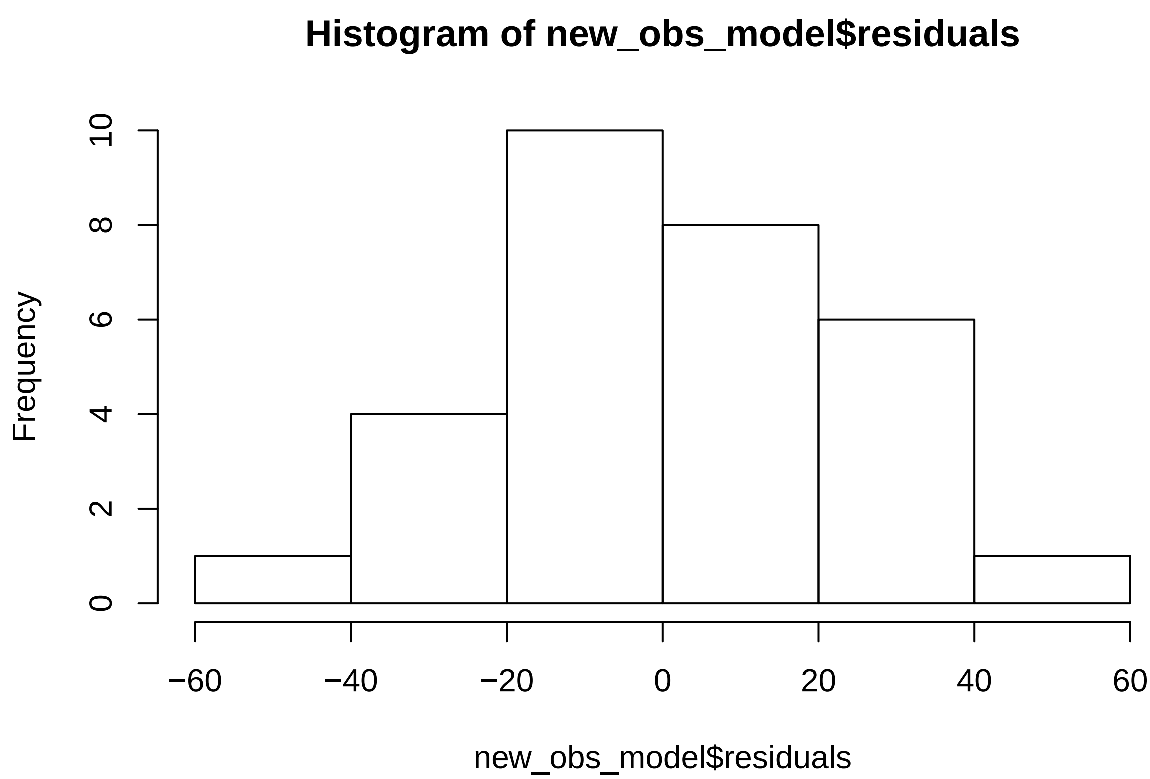
```
##
## Call:
## lm(formula = runs ~ new_obs, data = mlb11)
##
## Coefficients:
## (Intercept)      new_obs
##     -686.61      1919.36
##
##
```

```
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = Model)
##
##                       Value p-value                Decision
## Global Stat        3.711018 0.44653 Assumptions acceptable.
## Skewness           0.042733 0.83623 Assumptions acceptable.
## Kurtosis           0.610288 0.43468 Assumptions acceptable.
## Link Function      2.615006 0.10586 Assumptions acceptable.
## Heteroscedasticity 0.442990 0.50568 Assumptions acceptable.
```

The test results are consistent with Linearity.

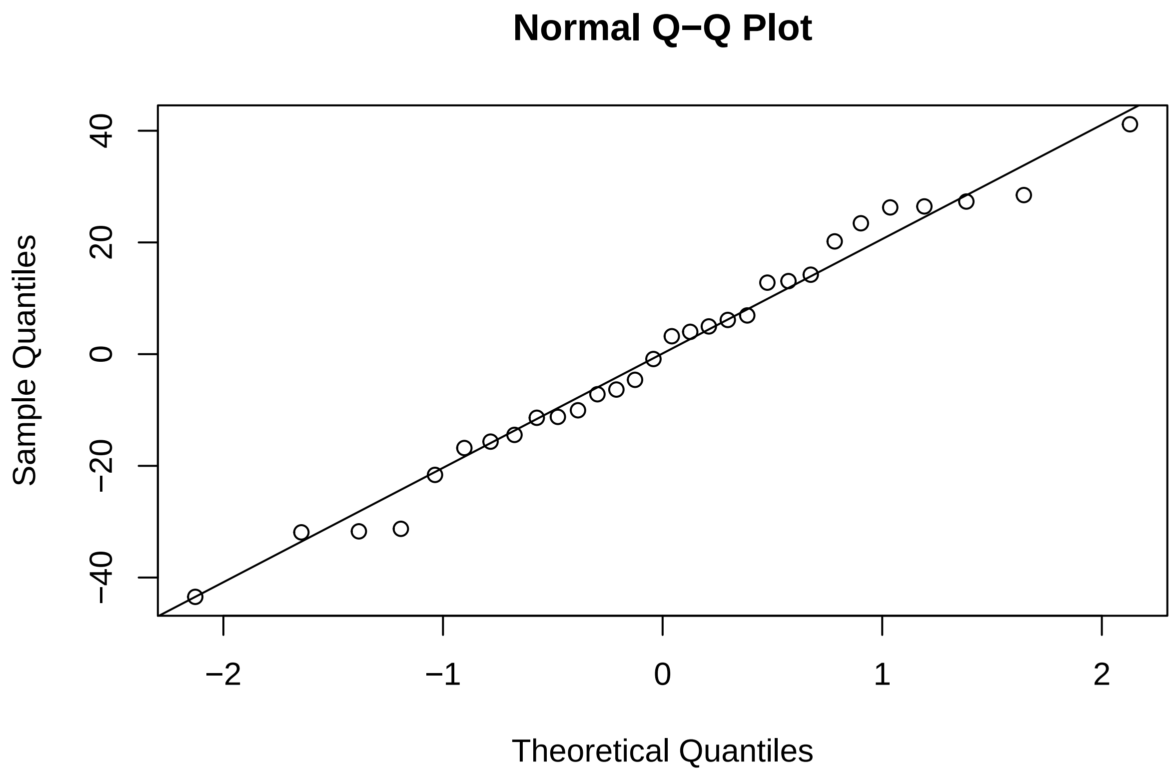
Nearly normal residuals: To check this condition, we can look at a histogram:

```
hist(new_obs_model$residuals)
```



or a normal probability plot of the residuals:

```
qqnorm(new_obs_model$residuals)
qqline(new_obs_model$residuals)   # adds diagonal line to the normal prob plot
```



The above plots confirm that the residuals from the OBS model are "nearly normal."

```
### Shapiro-Wilks test of Normality:
shapiro.test(new_obs_model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  new_obs_model$residuals
## W = 0.980683, p-value = 0.84342
```

```
### Kolmogorov-Smirnov test of Normality:
ks.test(new_obs_model$residuals,"pnorm",0,sd(new_obs_model$residuals))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  new_obs_model$residuals
## D = 0.0672809, p-value = 0.99778
## alternative hypothesis: two-sided
```

```
### Anderson-Darling test of Normality:
require(nortest)
ad.test(new_obs_model$residuals)
```

```
##
##  Anderson-Darling normality test
```

```
##
## data:  new_obs_model$residuals
## A = 0.202652, p-value = 0.86542
```

```
### Jarque-Bera test of Normality:

require(tseries)
jarque.bera.test(new_obs_model$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  new_obs_model$residuals
## X-squared = 0.653021, df = 2, p-value = 0.72144
```

The extremely high p-values for all of the above tests cause us to fail to reject the null hypothesis, which is that the residuals are Normal.

Constant variability :

```
require(olsrr)
ols_test_breusch_pagan(m1)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##               Data
##  -------------------------------
##  Response : runs
##  Variables: fitted values of runs
##
##          Test Summary
##  -----------------------------
##  DF            =    1
##  Chi2          =    0.014290331
##  Prob > Chi2   =    0.90484583
```

The high p-value provides evidence in favor of the null hypothesis, i.e., the variance is constant.