Introduction to linear regression

Jimmy Ng

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, betterpredict 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.

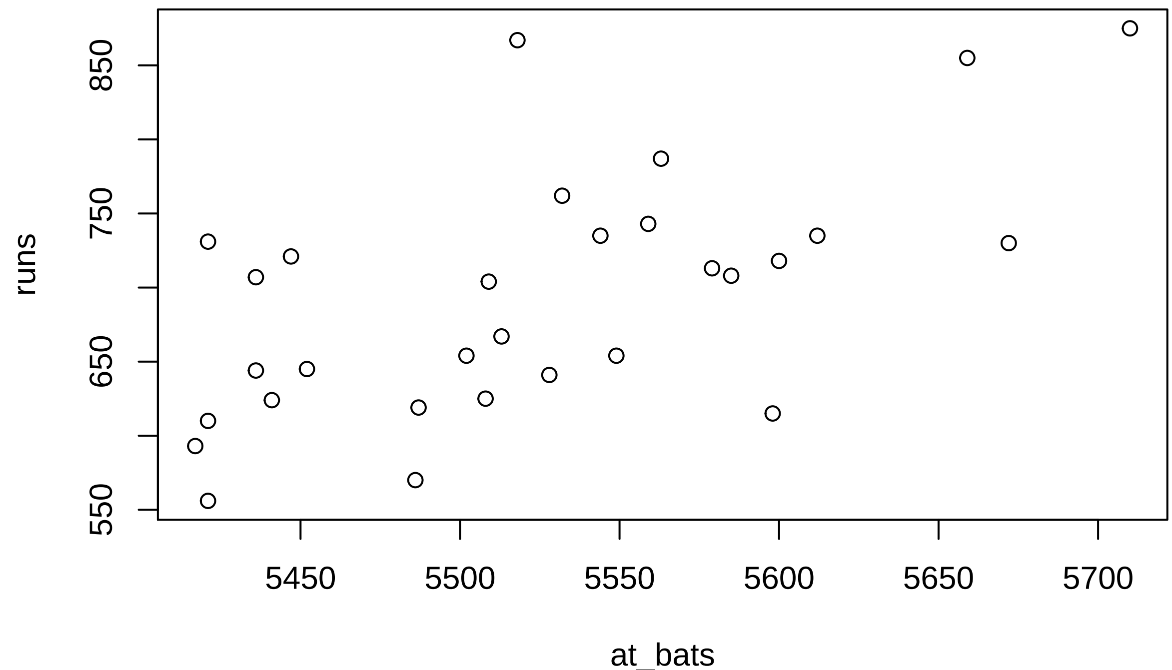
```
load("more/mlb11.RData")
```

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?

```
# I would use a scatter plot to display the relationship between "runs" and one of the other numerical
# the relationship looks linear, and I would be comfortable to use it to predict the number of runs
# the cor.test displays a statistically significant positive correlation between the two variables

plot(runs ~ at_bats, data = mlb11)
```




```r
cor.test(mlb11$runs, mlb11$at_bats)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mlb11$runs and mlb11$at_bats
## t = 4.0801, df = 28, p-value = 0.0003388
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3209675 0.7958231
## sample estimates:
##      cor
## 0.610627
```

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

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

```
## [1] 0.610627
```

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.

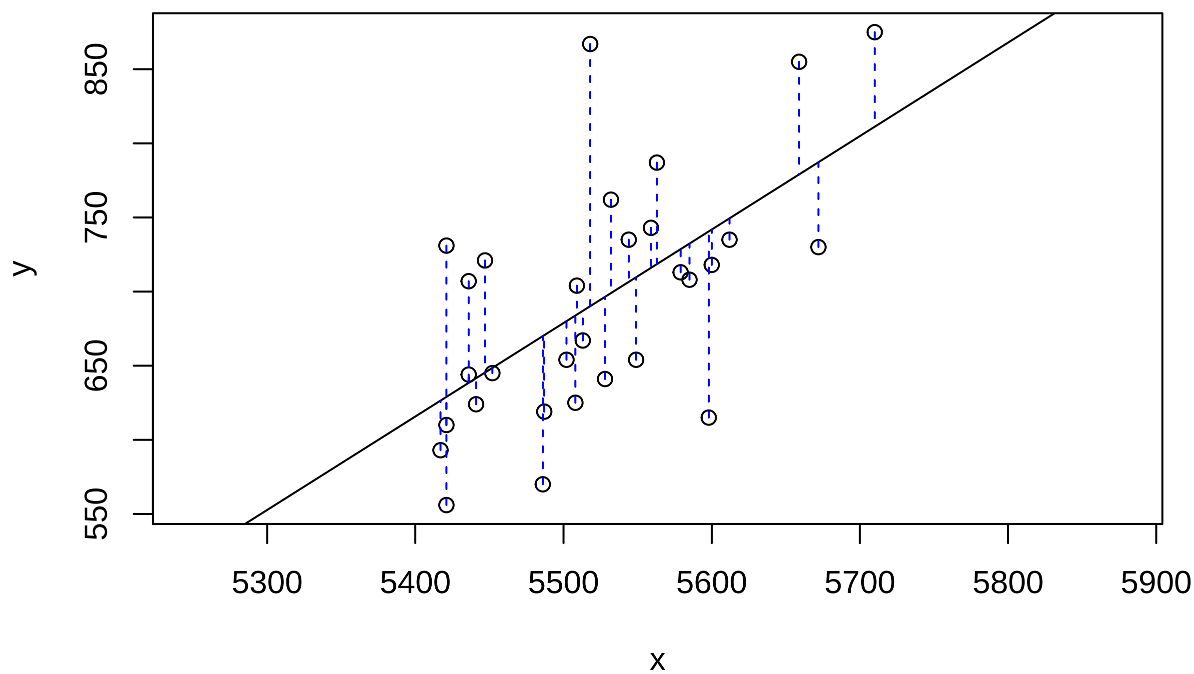
2. Looking at your plot from the previous exercise, describe the relationship between these two variables.

Make sure to discuss the form, direction, and strength of the relationship as well as any unusual observations.

JN: that's a linear relationship between the two. The two variables go in the same positive relationship, meaning both go up at the same time. Obviously, that's a moderate to strong positive correlation. There's no apparent outliner in this plot.

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
##  -2789.2429       0.6305
##
```

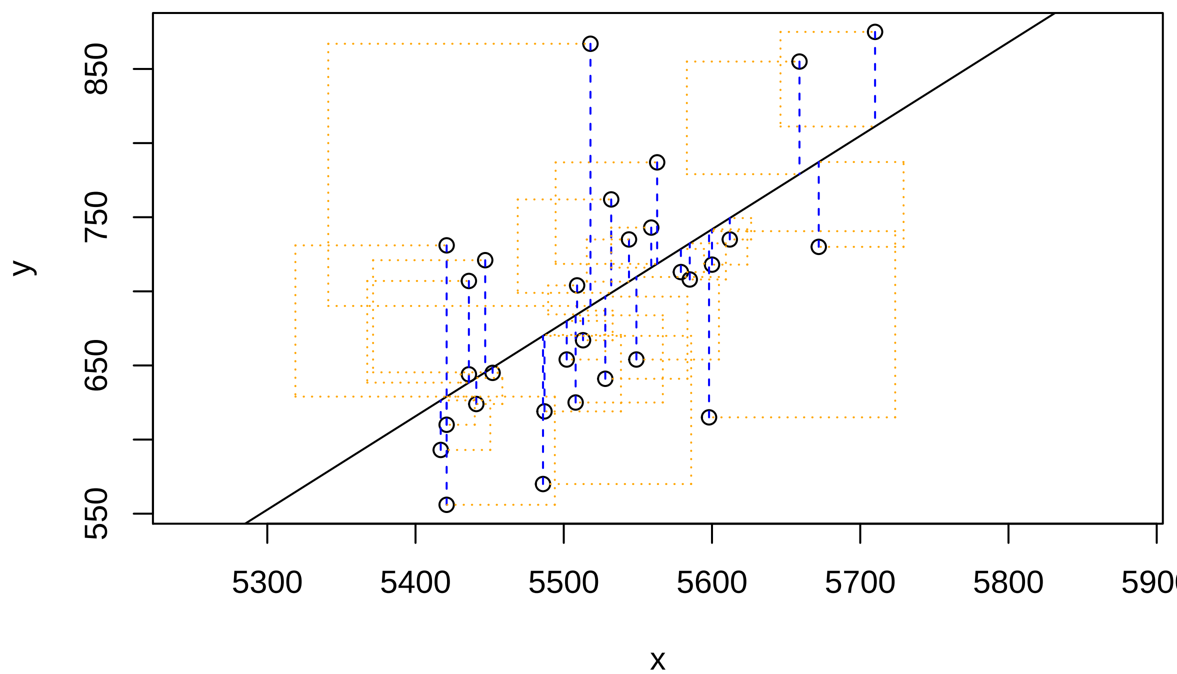
```
## Sum of Squares:  123721.9
```

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`.

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



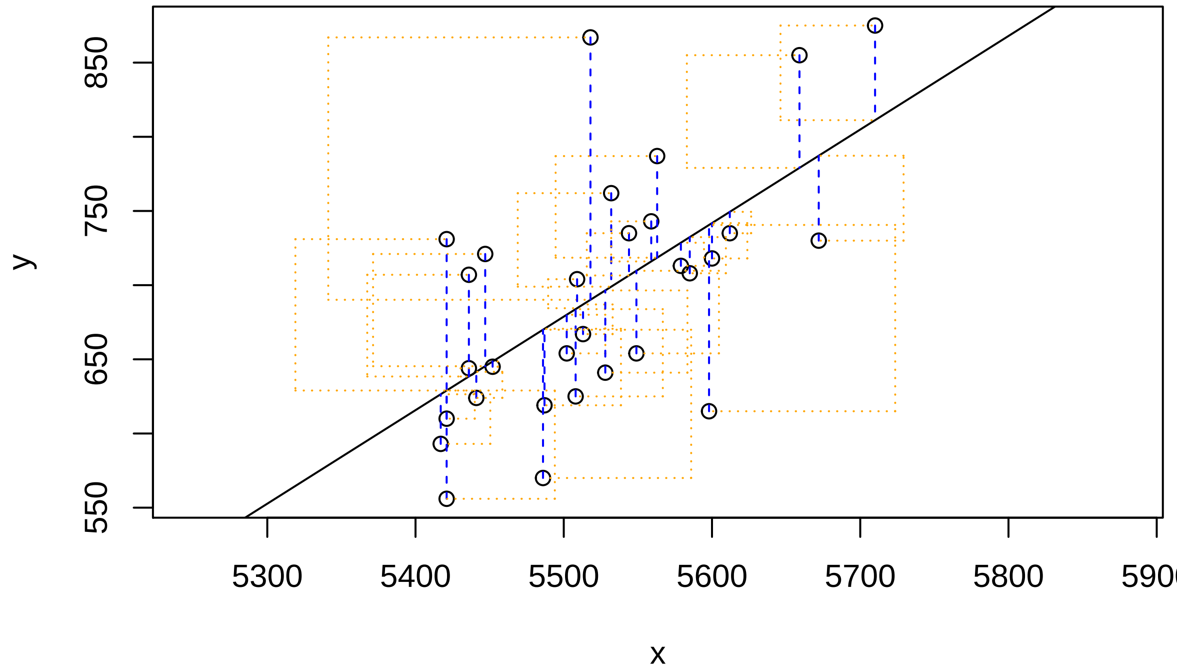

```
## Click two points to make a line.

## Call:
## lm(formula = y ~ x, data = pts)
##
## Coefficients:
## (Intercept)            x
##  -2789.2429       0.6305
##
## Sum of Squares:  123721.9
```

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?

```
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
##  -2789.2429       0.6305
##
## Sum of Squares:  123721.9
```

JN: the interactive feature does not work in RMarkdown. There's nothing I can choose or toggle upon.

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).

```r
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
## -125.58  -47.05  -16.59   54.40  176.87
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2789.2429   853.6957  -3.267 0.002871 **
## at_bats         0.6305     0.1545   4.080 0.000339 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66.47 on 28 degrees of freedom
## Multiple R-squared:  0.3729, Adjusted R-squared:  0.3505
## F-statistic: 16.65 on 1 and 28 DF,  p-value: 0.0003388
```

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
lm(runs ~ homeruns, data = mlb11)
```

```
##
## Call:
## lm(formula = runs ~ homeruns, data = mlb11)
##
## Coefficients:
## (Intercept)     homeruns
##     415.239        1.835
```

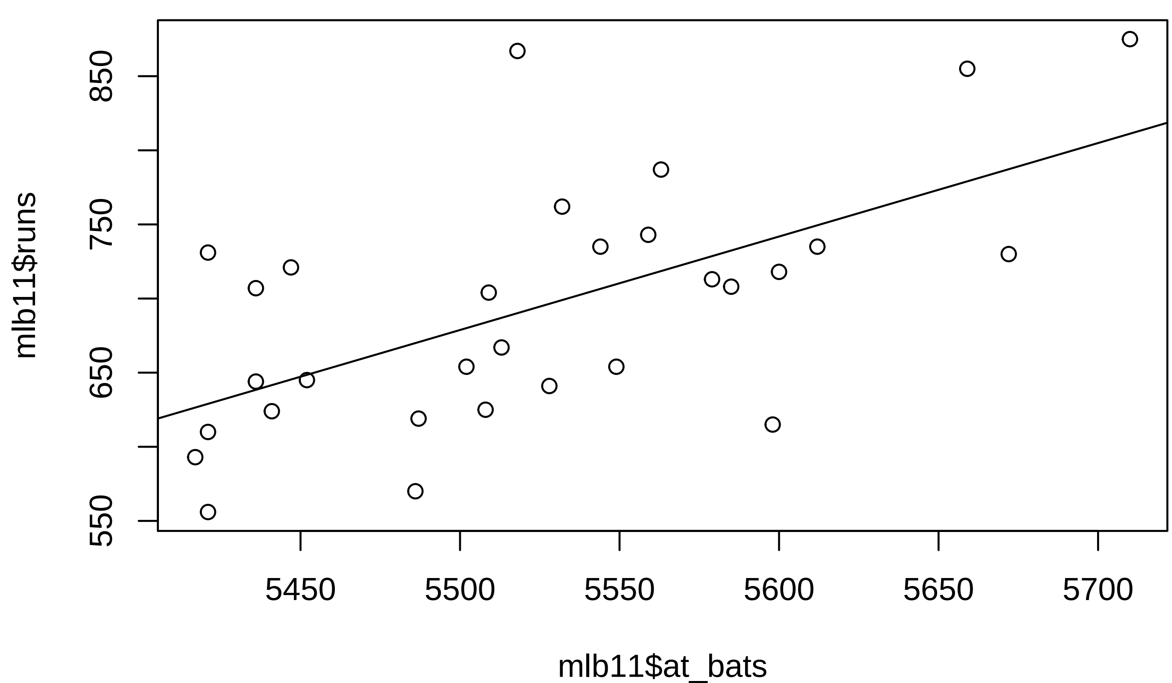
JN: The slope (1.835) indicates a positive relationship, i.e. every single home run would correspond to 1.835 increase of runs plus 415.239 (as intercept). The equation line:

$$\hat{y} = 415.239 + 1.835 * homeruns$$

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)
```



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?

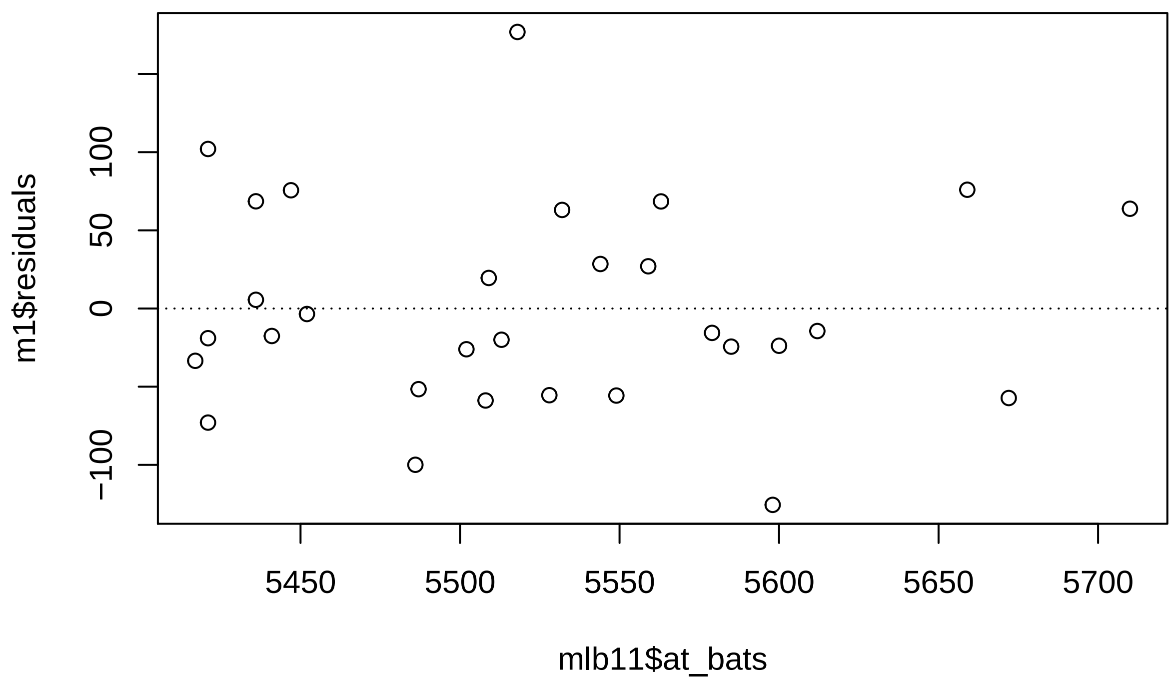
JN: he or she would have predicted between 720 and 730 runs. That would be an overestimation (as the actual data points are beneath the line) by 20 to 40 points. The residuals would be between -20 to -40.

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)
abline(h = 0, lty = 3)  # adds a horizontal dashed line at y = 0
```

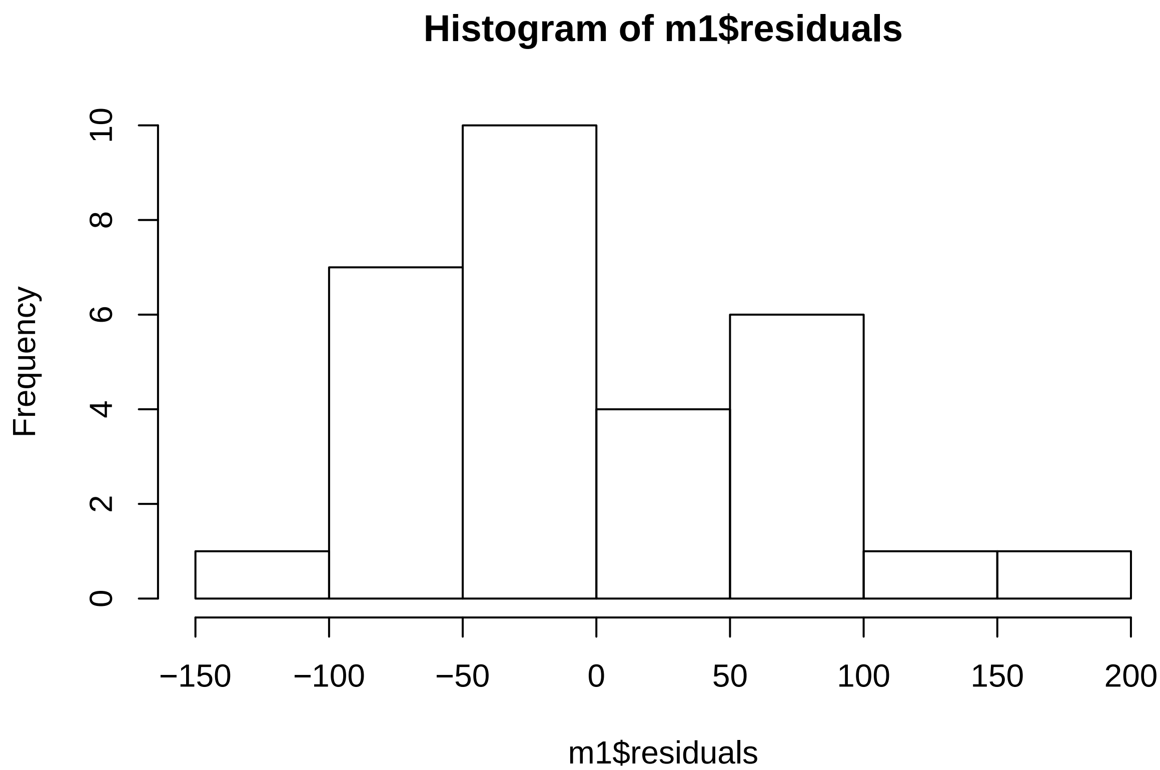


6. Is there any apparent pattern in the residuals plot? What does this indicate about the linearity of the relationship between runs and at-bats?

JN: there's a random pattern (which is good and that's what we expect in a linear model). There's a linear relationship between runs and at-bats, and the variance stays consistant throughout the predicting variable.

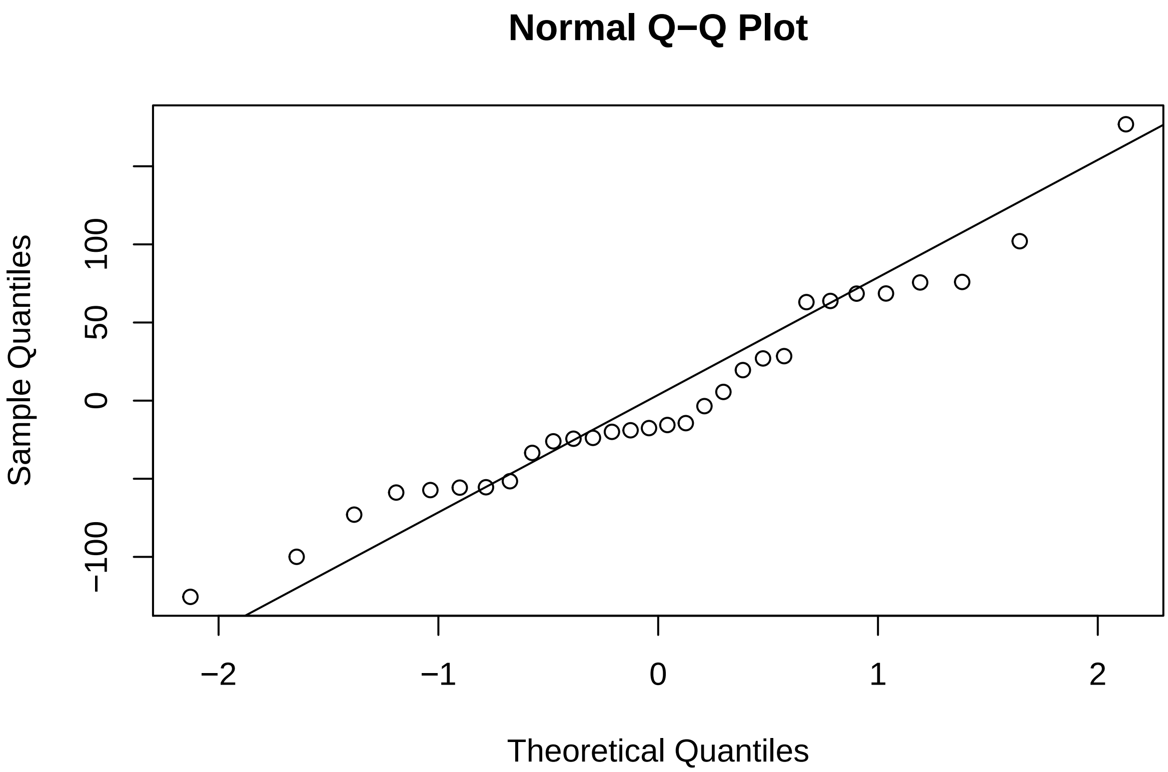
Nearly normal residuals: To check this condition, we can look at a histogram

```
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?

JN: Yes

Constant variability:

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

JN: Yes, the homoscedasticity is met.

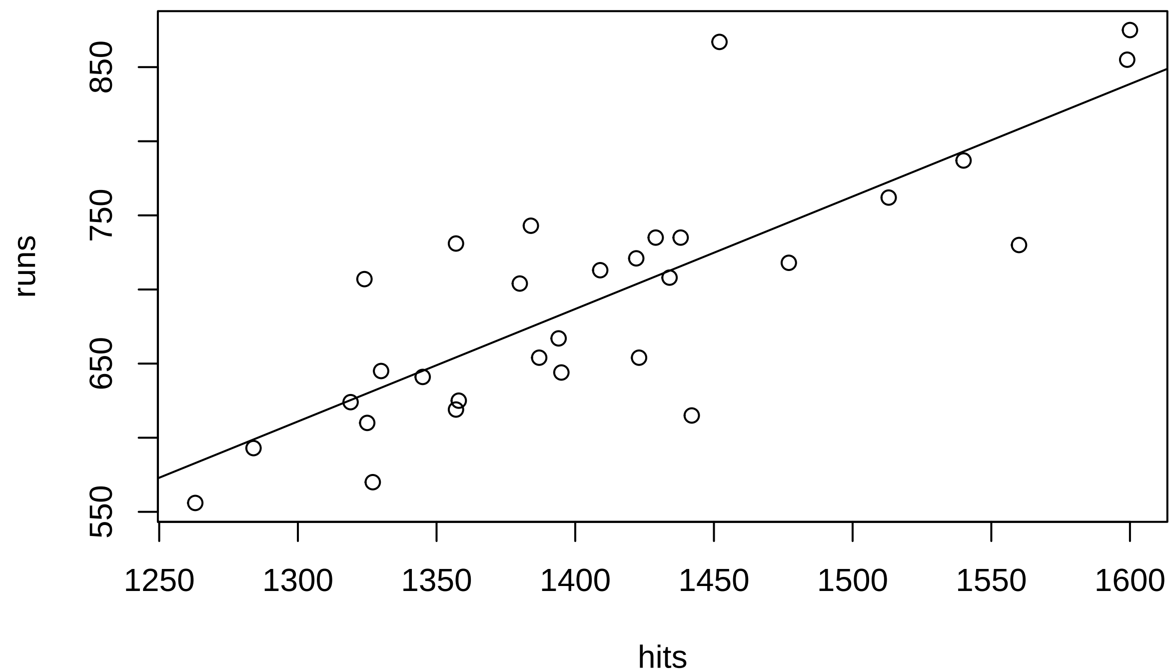
On Your Own

- Choose another 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?

```
# there's a strong evidence of positive correlation between runs and hits

plot(runs ~ hits, data = mlb11)
```

```r
hits.lm <- lm(runs ~ hits, data = mlb11)
abline(hits.lm)
```




```r
cor.test(mlb11$runs, mlb11$hits)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mlb11$runs and mlb11$hits
## t = 7.0851, df = 28, p-value = 1.043e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6198673 0.9013142
## sample estimates:
##       cor
## 0.8012108
```

- 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?

```r
# Using a cor.test() function to find the correlation between each of these variables with runs and we

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

```
##
##  Pearson's product-moment correlation
##
```

```
## data:  mlb11$runs and mlb11$at_bats
## t = 4.0801, df = 28, p-value = 0.0003388
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3209675 0.7958231
## sample estimates:
##      cor
## 0.610627
```

```r
cor.test(mlb11$runs, mlb11$hits)
```

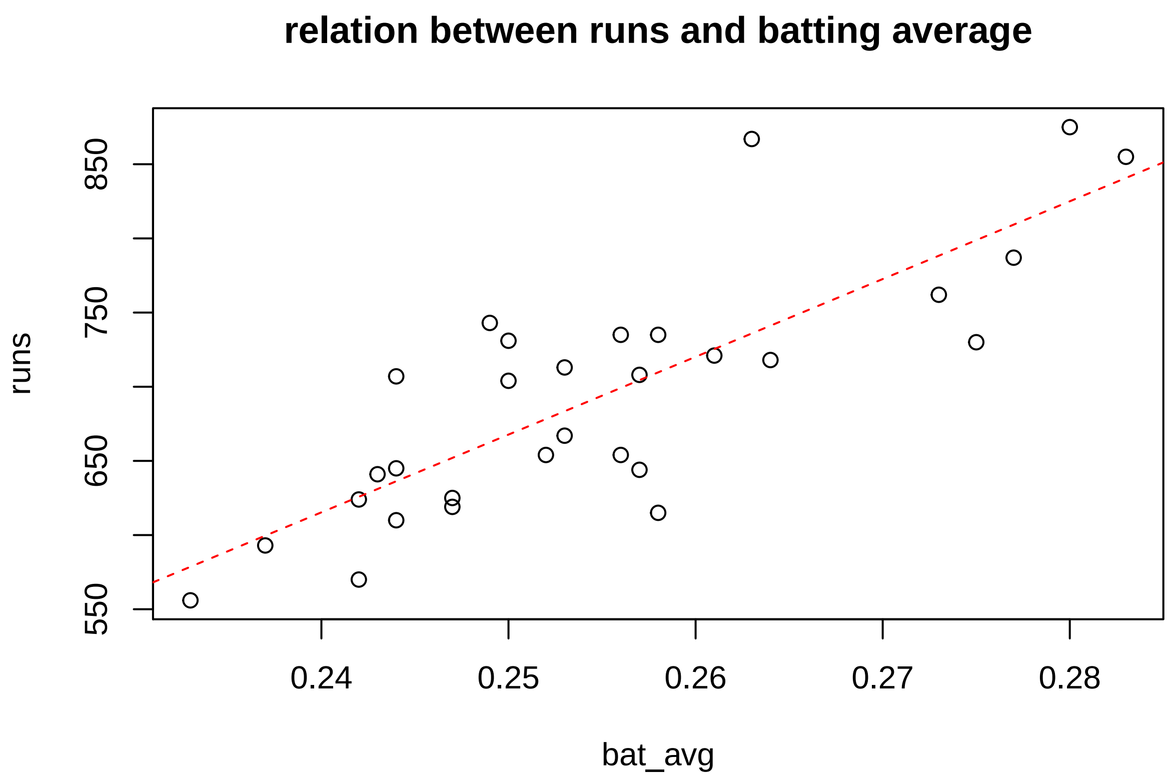
```
##
##  Pearson's product-moment correlation
##
## data:  mlb11$runs and mlb11$hits
## t = 7.0851, df = 28, p-value = 1.043e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6198673 0.9013142
## sample estimates:
##       cor
## 0.8012108
```

- 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).

```r
# bat_avg (followed by hits) has the strongest correlation paired with runs among the traditional varia

library(dplyr)
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
attach(mlb11)
corList <- lapply(list(at_bats, hits, homeruns, bat_avg, strikeouts, stolen_bases, wins),
                  function(x) cor(runs, x))
names(corList) <- c("at_bats", "hits", "homeruns", "bat_avg", "strikeouts", "stolen_bases", "wins")
unlist(corList) %>%
        as.data.frame %>%
        rename(., coefficient = .) %>%
        mutate(variable = row.names(.)) %>%
        arrange(desc(abs(coefficient)))
##   coefficient     variable
## 1  0.80998589      bat_avg
## 2  0.80121081         hits
## 3  0.79155769     homeruns
## 4  0.61062705      at_bats
## 5  0.60080877         wins
## 6 -0.41153120   strikeouts
## 7  0.05398141 stolen_bases
```

```r
plot(runs ~ bat_avg, data = mlb11, main = "relation between runs and batting average")
hits.ba <- lm(runs ~ bat_avg, data = mlb11)
abline(hits.ba, col = "red", lty = 2)
```




```r
cor.test(runs, bat_avg)
##
##  Pearson's product-moment correlation
##
## data:  runs and bat_avg
## t = 7.3085, df = 28, p-value = 5.877e-08
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6350253 0.9059013
## sample estimates:
##       cor
## 0.8099859
detach(mlb11)
```

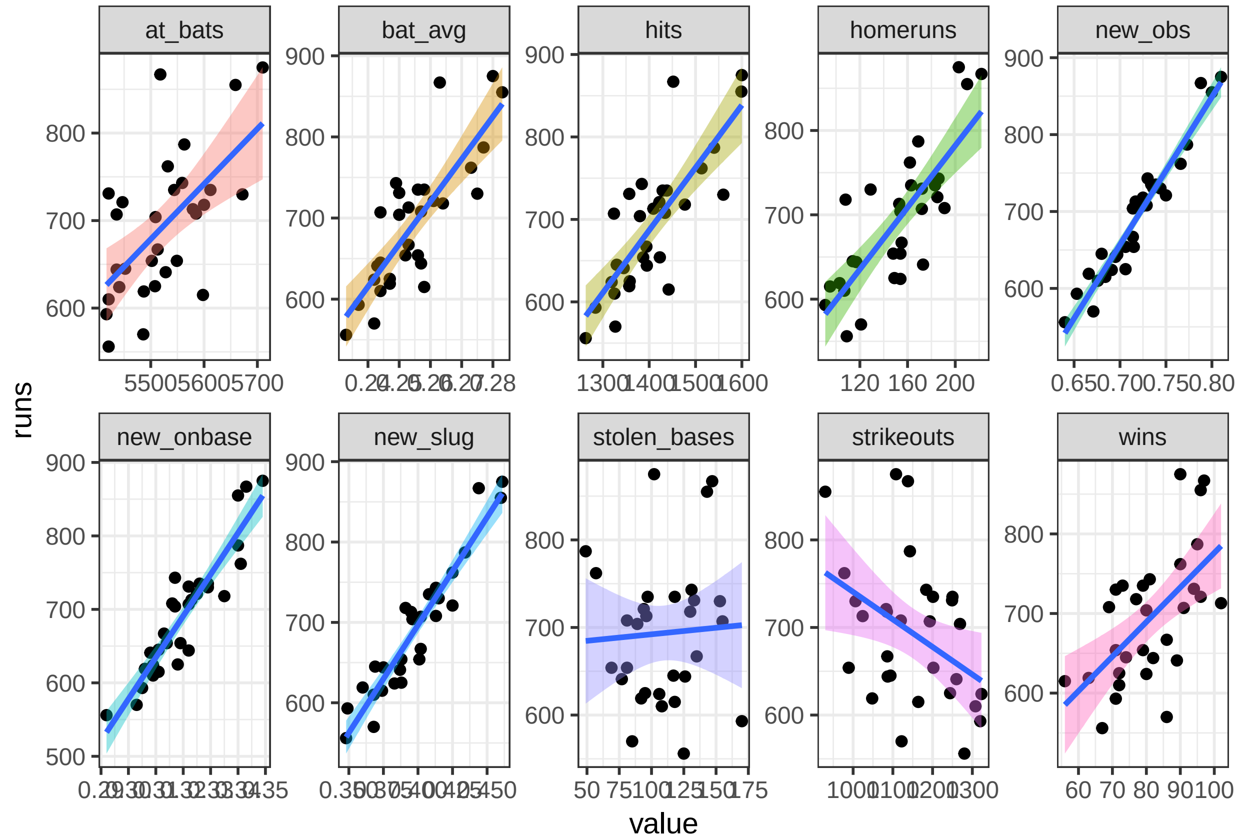
- Now examine the three newer variables. These are the statistics used by the author of *Moneyball* to predict a teams success. In general, are they more or less effective at predicting runs that the old variables? Explain using appropriate graphical and numerical evidence. Of all ten variables we've 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?

```r
# yes these new variables are more successful (or more predictive than traditional ones)
# the best (or most predictive) variable is "new_obs"
```

```r
# the correlation coefficient is 0.96691630, accounting for 93% of the variance in "runs"

library(dplyr)
attach(mlb11)
corList <- lapply(list(at_bats, hits, homeruns, bat_avg, strikeouts, stolen_bases, wins,
                       new_onbase, new_slug, new_obs),
                  function(x) cor(runs, x))
names(corList) <- c("at_bats", "hits", "homeruns", "bat_avg", "strikeouts", "stolen_bases", "wins",
                    "new_onbase", "new_slug", "new_obs")
unlist(corList) %>%
        as.data.frame %>%
        rename(., coefficient = .) %>%
        mutate(variable = row.names(.)) %>%
        arrange(desc(abs(coefficient)))
##    coefficient     variable
## 1   0.96691630      new_obs
## 2   0.94703240     new_slug
## 3   0.92146907   new_onbase
## 4   0.80998589      bat_avg
## 5   0.80121081         hits
## 6   0.79155769     homeruns
## 7   0.61062705      at_bats
## 8   0.60080877         wins
## 9  -0.41153120   strikeouts
## 10  0.05398141 stolen_bases

# plot them all
library(tidyr)
library(ggplot2)
mlb11 %>%
        select(-team) %>%
        gather(variables, value, -runs) %>%
        select(variables, runs, value) %>%
        ggplot(., aes(x = value, y = runs, fill = variables)) +
        geom_point() +
        geom_smooth(method = "lm") +
        theme_bw() +
        theme(legend.position = "none") +
        facet_wrap(~variables, scales = "free", nrow = 2)
```




```r
detach(mlb11)
```

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

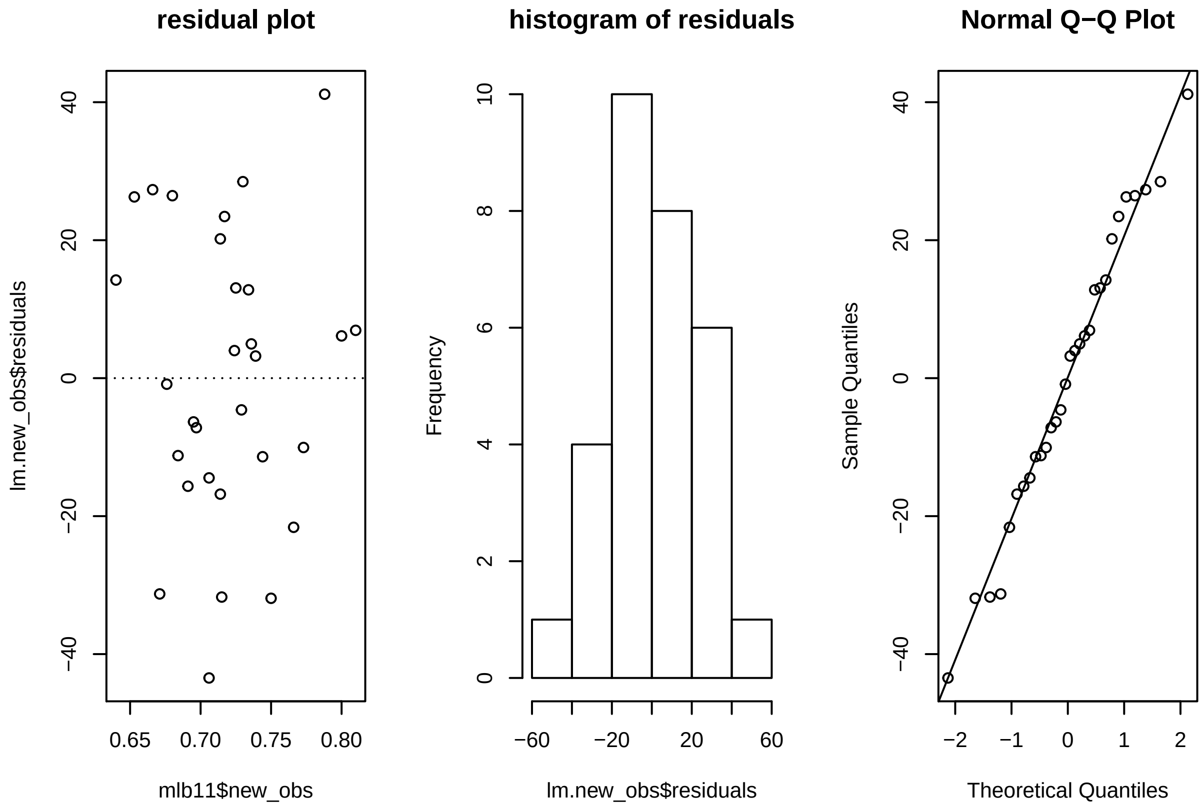
```r
# Yes, everything checks out!

par(mfrow = c(1, 3))
lm.new_obs <- lm(runs ~ new_obs, data = mlb11)

plot(lm.new_obs$residuals ~ mlb11$new_obs, main = "residual plot")
abline(h = 0, lty = 3)

hist(lm.new_obs$residuals, main = "histogram of residuals")

qqnorm(lm.new_obs$residuals)
qqline(lm.new_obs$residuals)
```


residual plot
histogram of residuals
Normal Q−Q Plot
lm.new_obs$residuals
mlb11$new_obs
Frequency
lm.new_obs$residuals
Sample Quantiles
Theoretical Quantiles