Case Study 3

Authors: Taiyo Williamson, Alexis Leherr, Alessandro Barrera

Instructions:

Get into groups of 3 and answer the following questions. Submit your knitted report as a pdf/html file and include the original .Rmd file also. Submit one report per group with all members' names on it before the end of class. You currently have the fundamental R tools to complete this exercise, but you will may still have to explore new techniques and packages. **For each question, write the name of the team member who answered/coded it.**

1. Regression Modeling & Functional Programming

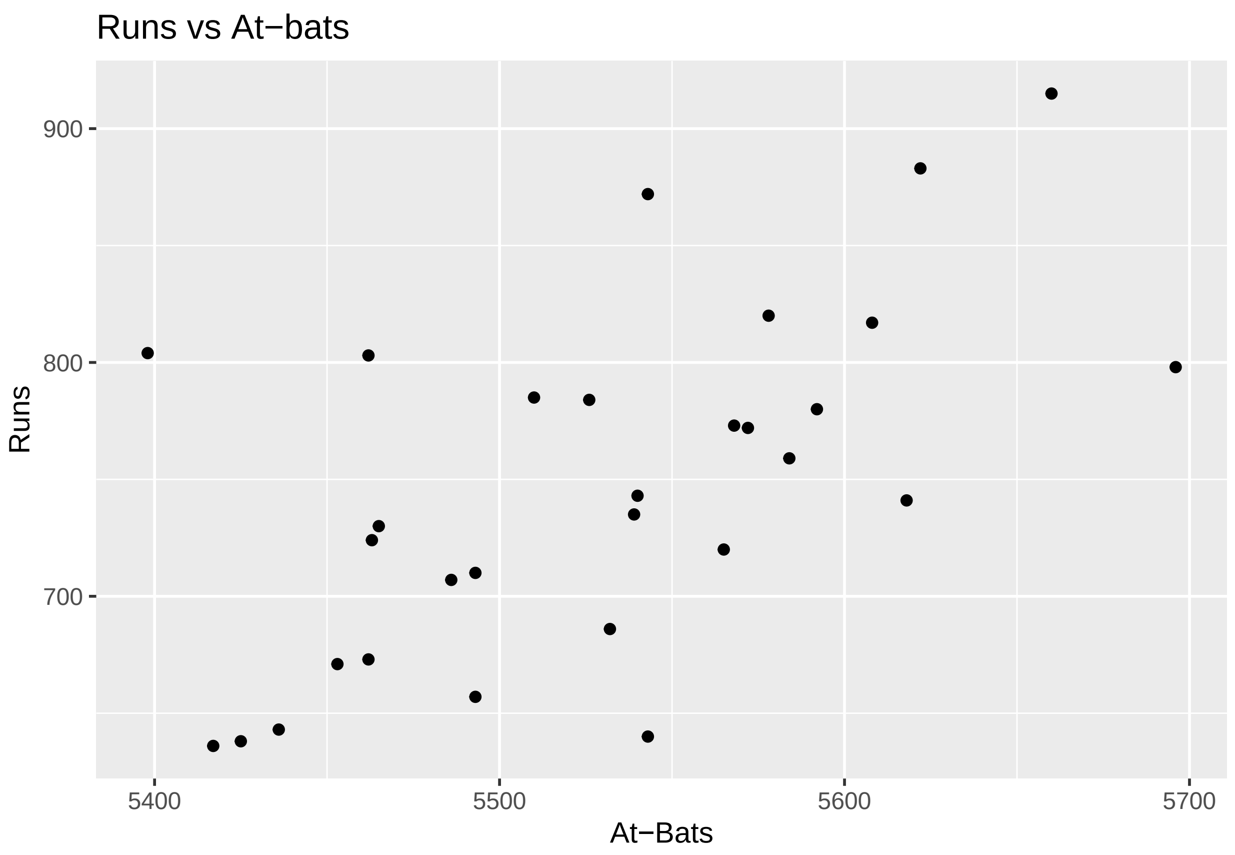
The movie Moneyball is about how proper use of statistics in baseball (called "sabermetrics") can bring unexpected success to a low-ranked, low-budget team. In it, the manager of the Oakland A's believes that (then) unpopular statistics, like a player's ability to get on base, can predict the team's ability to score runs better than traditional statistics, such as homerun counts and batting averages. By recruiting players who scored high in these underused statistics, he was able to improve the record of the team without needing to spend exorbitant amounts of money on the more mainstream players.

We will examine the data from the 30 MLB teams during the 2009 season. We will search for linear relationships between potential explanatory variables and the response variable: the number of runs scored in a season, which we treat as a measure of "success" for this data analysis. You don't need to know the rules of baseball to understand this question, but if you would like a refresher you can check out Wikipedia: https://en.wikipedia.org/wiki/Baseball_rules#Gameplay

In addition to runs scored, there are seven traditionally-used variables in the data set: at-bats, hits, homeruns, batting average, strikeouts, walks and stolen bases. The last three variables in the data set are "nontraditional": on-base percentage, slugging percentage, and on base plus slugging.

Done by Taiyo (a) Import the 2009 MLB dataset into R Studio. The dataset can be found on Canvas in the file "mlb09.csv". Using `ggplot`, plot `at_bats` on the x-axis and `runs` on the y-axis. Describe the relationship between the two variables in terms of direction (positively or negatively correlated), shape (linear? quadratic? exponential?) and strength. How confident would you rate your ability to predict a team's season runs scored, if you just knew the team's at-bats?

```
ggplot(data, aes(at_bats, runs)) +
  geom_point() +
  ggtitle("Runs vs At-bats") +
  xlab ("At-Bats") +
  ylab ("Runs")
```



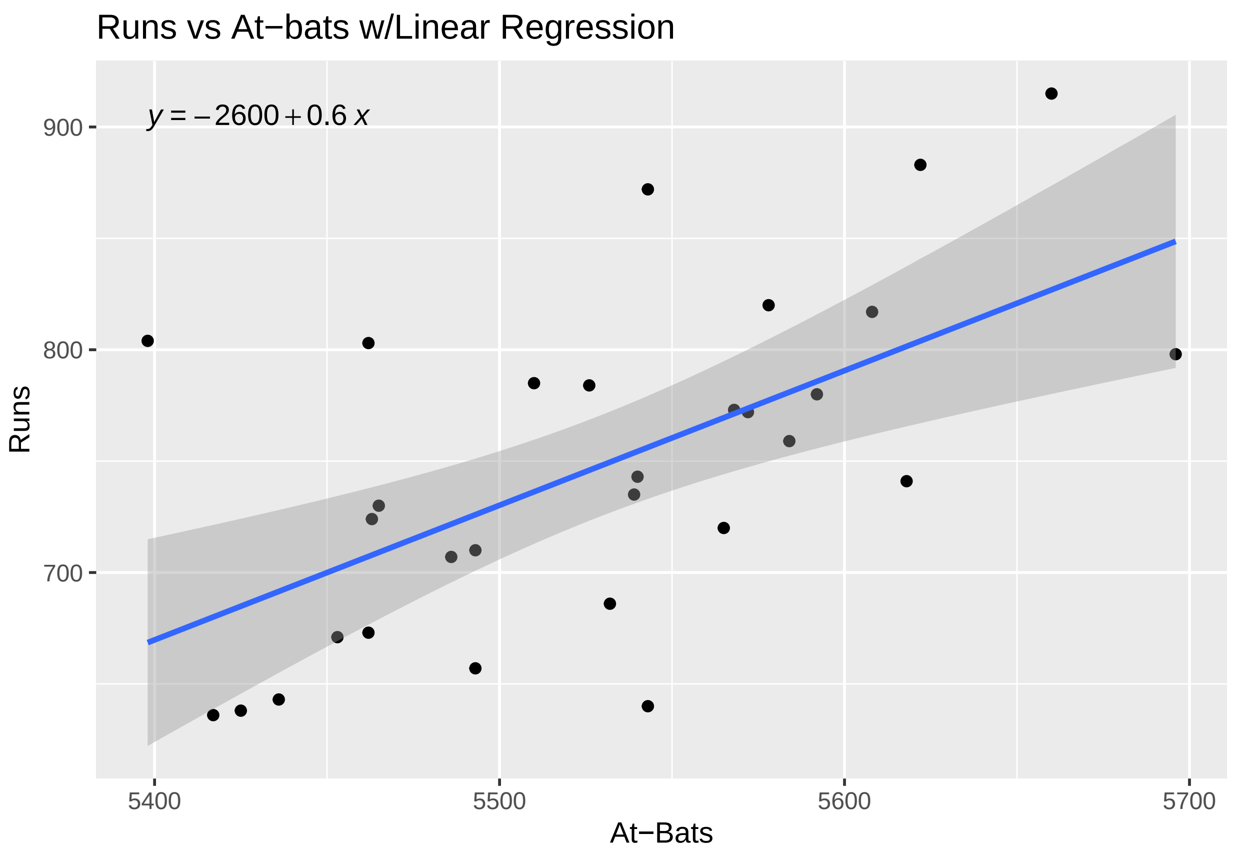
The relationship between `at_bats` and `runs` is positively correlated with a somewhat-linear shape. There is a relatively neutral strength (y=ax + b, x =~ 1). I would say I am about 90% confident I could predict a team's season runs scored, only knowing the team's at-bats.

Done by Taiyo

(b) Fit the regression of `runs` onto `at_bats`. Write the equation for the least squares regression line (LSRL, aka "best fit line") and interpret the intercept and the slope in the context of the problem.

```
ggplot(data, aes(at_bats, runs)) +
  geom_point() +
  ggtitle("Runs vs At-bats w/Linear Regression") +
  xlab ("At-Bats") +
  ylab ("Runs") +
  geom_smooth(method="lm") +
  stat_regline_equation()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
lmLine = lm(runs~at_bats, data=data)
lmLine
```

```
##
## Call:
## lm(formula = runs ~ at_bats, data = data)
##
## Coefficients:
## (Intercept)      at_bats
##  -2594.0311       0.6044
```

The slope is ~ 0.6044 and the y-intercept is about -2594.0311.

Done by Taiyo (c) Identify the coefficient of determination for this regression. Interpret R^2 in terms of the regression.

```
summary(lmLine)#0.362
```

```
##
## Call:
## lm(formula = runs ~ at_bats, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -116.19  -46.16  -13.05   33.95  135.45
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2594.0311   838.2929  -3.094 0.004441 **
## at_bats         0.6044     0.1516   3.986 0.000436 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.6 on 28 degrees of freedom
## Multiple R-squared:  0.362,  Adjusted R-squared:  0.3393
## F-statistic: 15.89 on 1 and 28 DF,  p-value: 0.000436
```

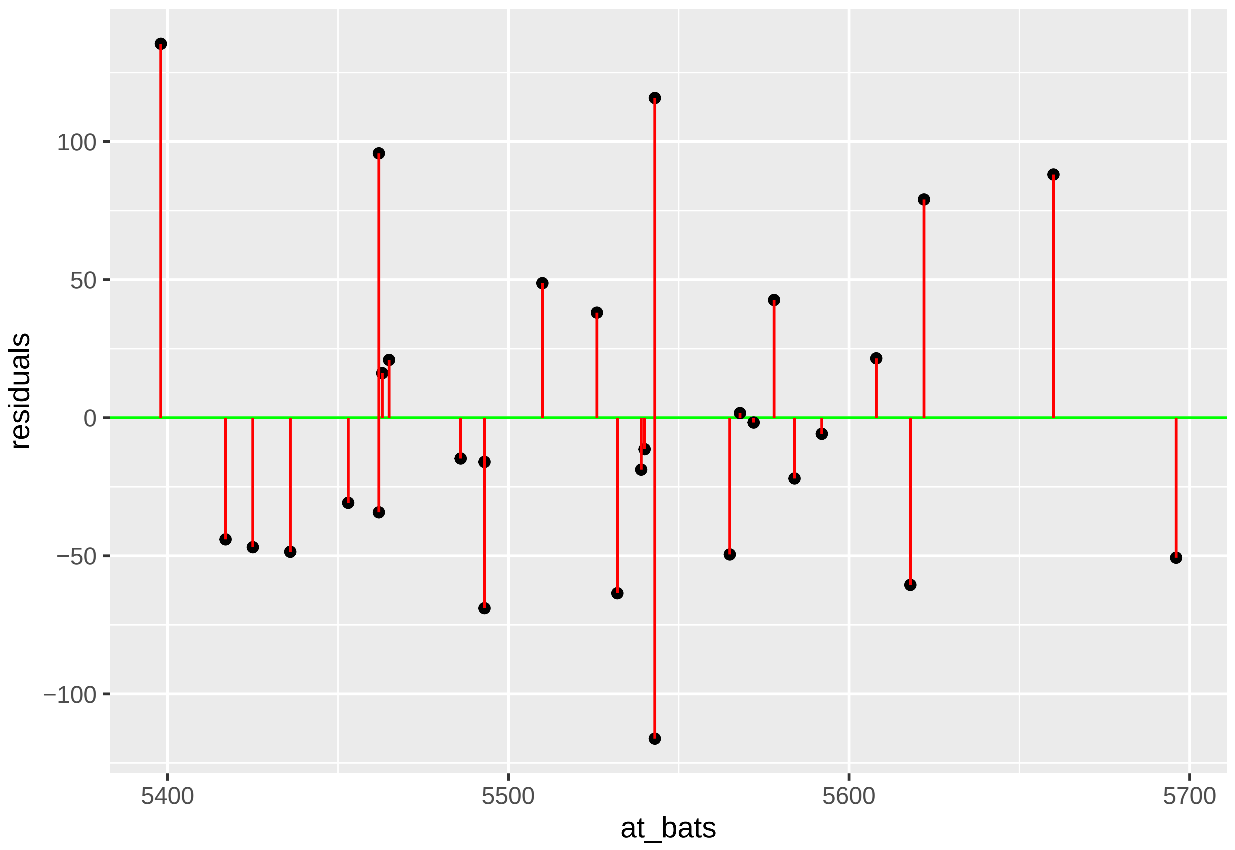
The coefficient of determination for this regression is the multiple R-squared value, which is 0.362. The R^2 statistic measures the strength of the linear relationship relative to the regression.

Done by Taiyo (d) Create the residual plot and discuss what it indicates. Does the plot suggest linearity or not? Do we suspect anything wrong with our model?

```
data$residuals = residuals(lmLine)
str(data)
```

```
## 'data.frame':    30 obs. of  13 variables:
##  $ team        : chr  "NY Yankees" "LA Angels" "Boston" "Philadelphia" ...
##  $ runs        : int  915 883 872 820 817 804 803 798 785 784 ...
##  $ at_bats     : int  5660 5622 5543 5578 5608 5398 5462 5696 5510 5526 ...
##  $ hits        : int  1604 1604 1495 1439 1539 1408 1434 1516 1447 1436 ...
##  $ homeruns    : int  244 173 212 224 172 190 199 209 182 224 ...
##  $ bat_avg     : num  0.28 0.29 0.27 0.26 0.27 0.26 0.26 0.27 0.26 0.26 ...
##  $ strikeouts  : int  1014 1054 1120 1155 1021 1277 1229 1028 1231 1253 ...
##  $ walks       : int  663 547 659 589 585 660 642 548 610 472 ...
##  $ stolen_bases: int  111 148 126 119 85 106 194 73 68 149 ...
##  $ on_base     : num  0.36 0.35 0.35 0.33 0.35 0.34 0.34 0.33 0.34 0.32 ...
##  $ slugging    : num  0.48 0.44 0.45 0.45 0.43 0.44 0.44 0.44 0.43 0.45 ...
##  $ ob_slg      : num  0.84 0.79 0.81 0.78 0.77 0.78 0.78 0.77 0.77 0.76 ...
##  $ residuals   : num  88.1 79.1 115.8 42.7 21.5 ...
```

```
ggplot(data, aes(at_bats, residuals)) +
  geom_hline(aes(yintercept=0), color="green") +
  geom_point() +
  geom_segment(aes(xend=at_bats, yend=0), color="red")
```



The plot suggests linearity; since the points are scattered pretty randomly around the residual line `y=0`, it means our linear model is appropriate without skewing towards one input. `geom_segment` shows that the distance from the residual line for each point is random.

Done by Taiyo (e) Suppose the manager of a team comes and asks you to predict how many runs his team will score if they get 5000 at-bats, 5500 at-bats, and 6000 at-bats. Write a function which takes these inputs as arguments and returns the predicted values according to the regression line. Implement defensive programming to ensure that you receive the expected input; write a descriptive error (or multiple, if different cases needed) to throw if this is not the case.

```r
data$predicted = predict(lmLine) #unnecessary, I'm just doing this cuz I want to

atbatpredict = function(x) {
  tryCatch(
    {
      return (0.6044*x - 2594.0311)
    },
    error = function(e) {
      message("Invalid input, ERROR")
    },
    warning = function(w) {
      warning("A warning was called")
      warning(w)
    }
    )
}
#test case for implemented defensive programming
atbatpredict("abcdefg")
```

```
## Invalid input, ERROR
```

Done by Taiyo (f) Use your function from (e) to predict how many runs his team will score if they get 5000 at-bats, 5500 at-bats, and 6000 at-bats (3 separate predictions). Be sure to caution him if you have any hesitance with your ability to perform any one of these predictions.

```r
#predicted values
atbatpredict(c(5000,5500,6000))
```

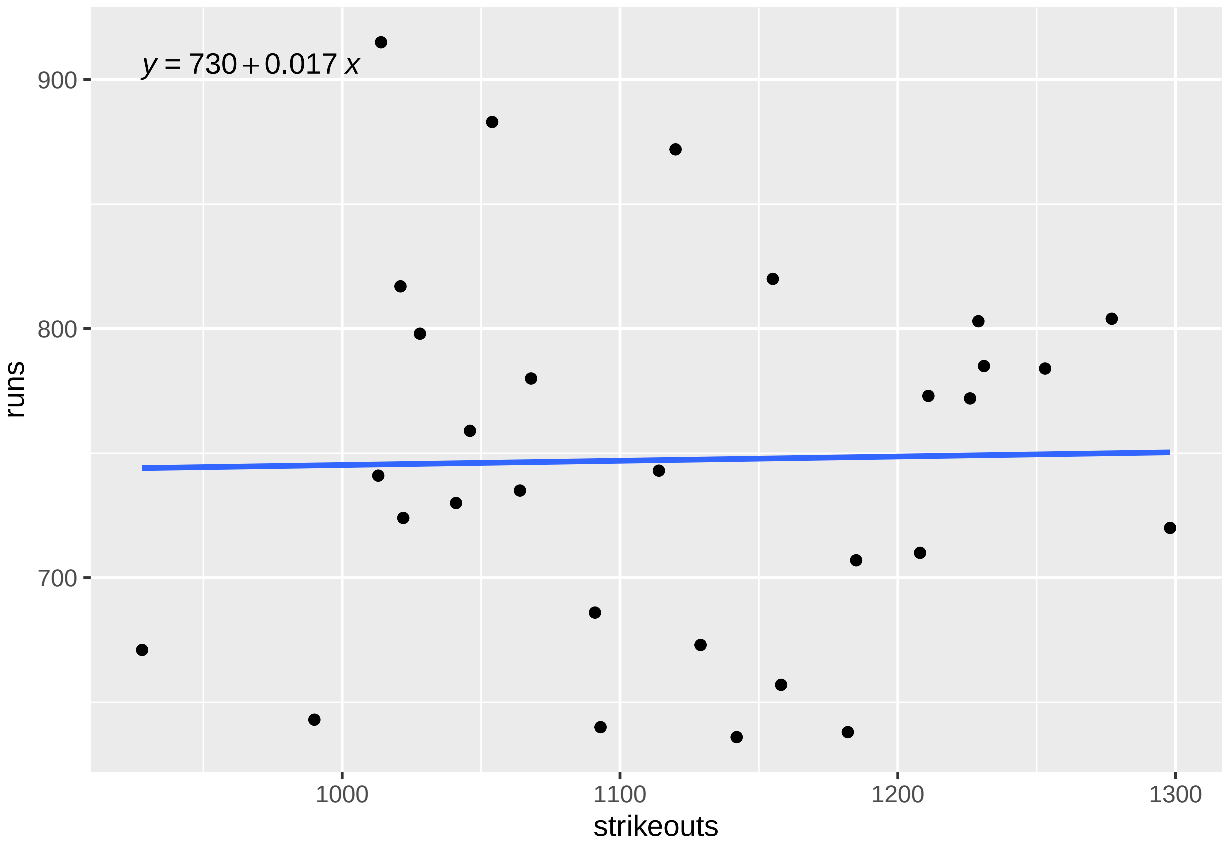
```
## [1]  427.9689  730.1689 1032.3689
```

Since my linear model is based off of `mlb09.csv` with `at_bats` range 5398, 5696, any potential inputs that are outside this range may be less accurate when predicting the number of runs. In this case, 5000 and 6000 at-bats are outisde of the range, so I am more hesitant in my ability to perform predictions for these 2 inputs.

Done by Alexis
(g) Suppose a rival team coach claims that `strikeouts` are the most important factor in scoring runs. Investigate his claim by creating the scatterplot of `strikeouts` vs. `runs`. Make an argument as to whether he is correct or not. Fit a regression and use the coefficient of determination to bolster your argument.

```r
mlb_plot = ggplot(data, aes(x = strikeouts, y = runs,))+
  geom_point()+
  geom_smooth(method = "lm", se = FALSE)+
  stat_regline_equation()
mlb_plot
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```r
corelation_coefficient =cor(data$strikeouts, data$runs)
coeffincient_determination = corelation_coefficient^2
```

```
coeffincient_determination
```

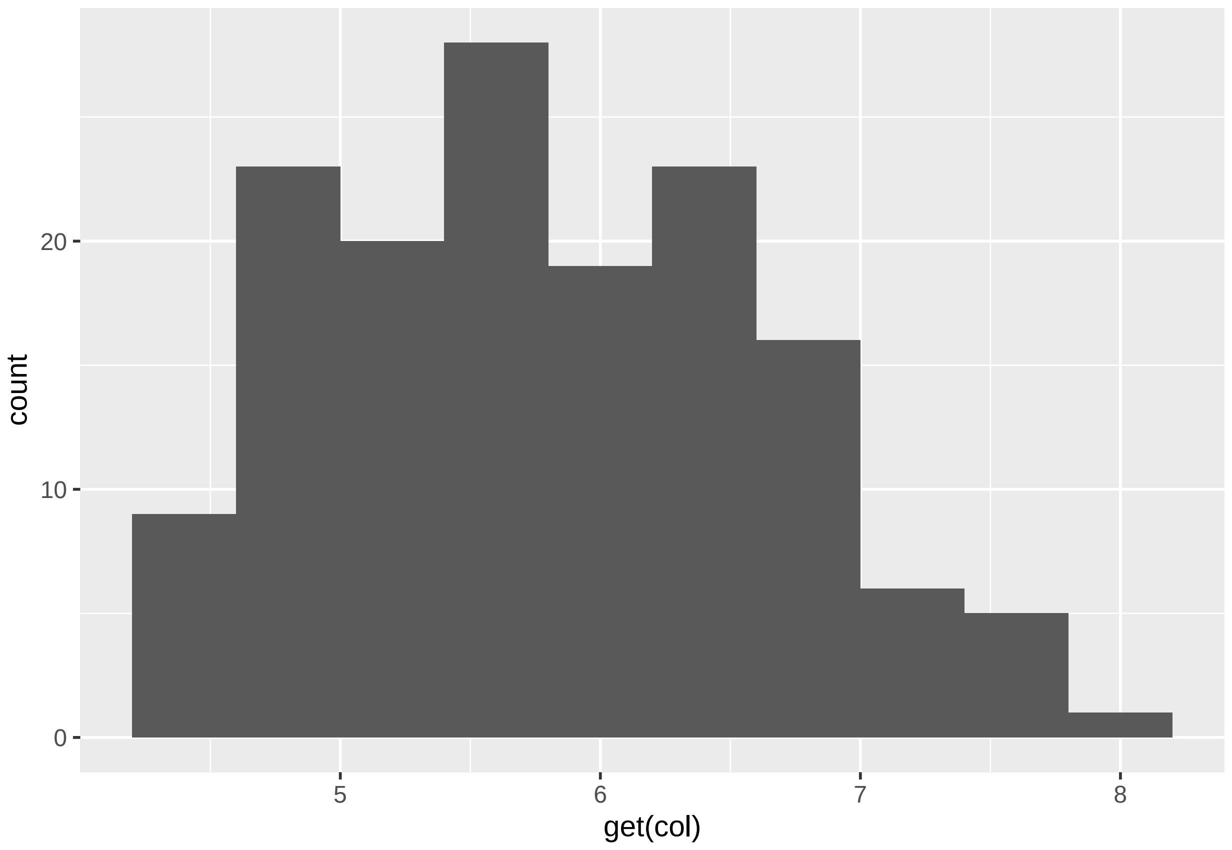
```
## [1] 0.0004775284
```

Strikeouts are not the most important factor in scoring runs. The line of best fit has a slop of 0.017 which is close to 0 and the coefficient of determination is close to 0.0, which support the claim that there is not a correlation between the strikeouts and runs.

Alessandro (h) We want to determine which of the 7 traditional variables (at-bats, hits, homeruns, batting average, strikeouts, walks, and stolen bases) best predicts `runs` with a linear model, by investigating their relationship to `runs` one at a time, visually. Write a function which creates the scatterplot and line of best fit for any given variable on the x-axis and `runs` on the y-axis. For the output of the function, return the scatterplot saved as a ggplot object.

```
Implement defensive programming to ensure that you receive the expected input; write a descriptive erro
```

```
ex_function = function(col){
  gg = ggplot(iris, aes(x = get(col))) +
    geom_histogram(bins = 10)
  return(gg)
}

ex_function("Sepal.Length")
```



```
best_fit = function(col){
  if (col %in% colnames(data)){
    gg = ggplot(data, aes(x = get(col), y = runs)) +
      geom_point() + xlab(col) +
      geom_smooth(method = "lm", se = FALSE)
```

```r
    return(gg)
  } else {
    stop("Error: Column not in data")
  }
}
```

Alessandro (i) Now we want to which of the 7 traditional variables (at-bats, hits, homeruns, batting average, strikeouts, walks, and stolen bases) best predicts `runs` with a linear model, by investigating their relationship to `runs` one at a time, *statistically*.

Write a function which fits a linear model and returns the R^2 value for any given variable on the x-axis and `runs` on the y-axis. For the output of the function, return a list of the R^2 values that result from regressing `runs` onto each variable.

```r
best_linear_model = function(col){
  if (col %in% colnames(data)){
    model = lm(runs ~ get(col), data = data)
    r_squared = summary(model)$r.squared
    return(r_squared)
  } else {
    stop("Error: Column not in data")
  }
}
```

Alessandro (j) Apply your functions from (h) and (i) to the dataset *by using a for loop* over the 7 traditional variables. Which seems to be best for predicting `runs`? Be sure to argue why you think your variable is best, using numeric and graphical evidence.

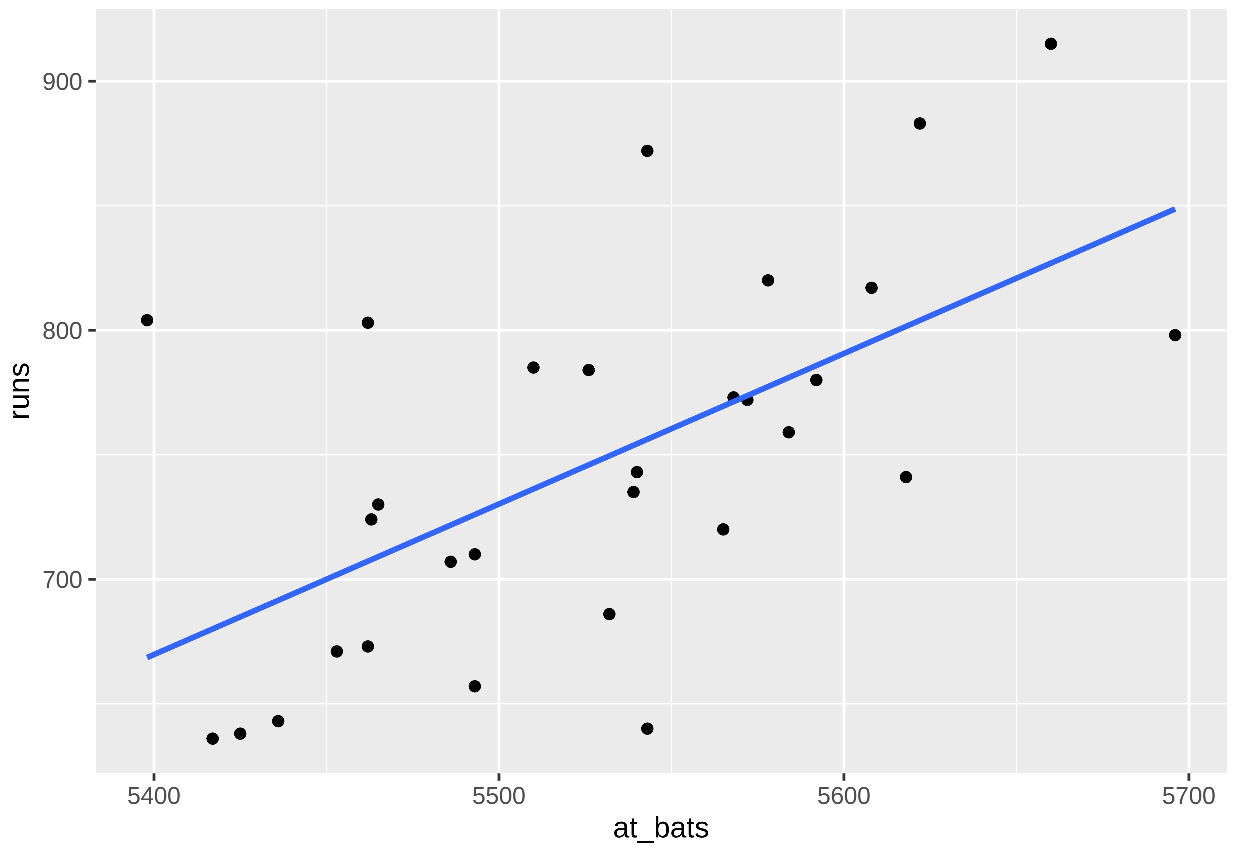
```r
seven_traditional_values = c("at_bats", "hits", "homeruns", "bat_avg", "strikeouts", "walks", "stolen_ba

r_squared_values = c()
for (i in seven_traditional_values){
  r_squared_values = c(r_squared_values, best_linear_model(i))
}

plots = list()
for (i in seven_traditional_values){
  plots[[i]] = best_fit(i)
}

r_squared_values
```

```
## [1] 0.3620410178 0.5844128949 0.5531983078 0.4752832907 0.0004775284
## [6] 0.4107462280 0.1362998528
```

```r
plots
```

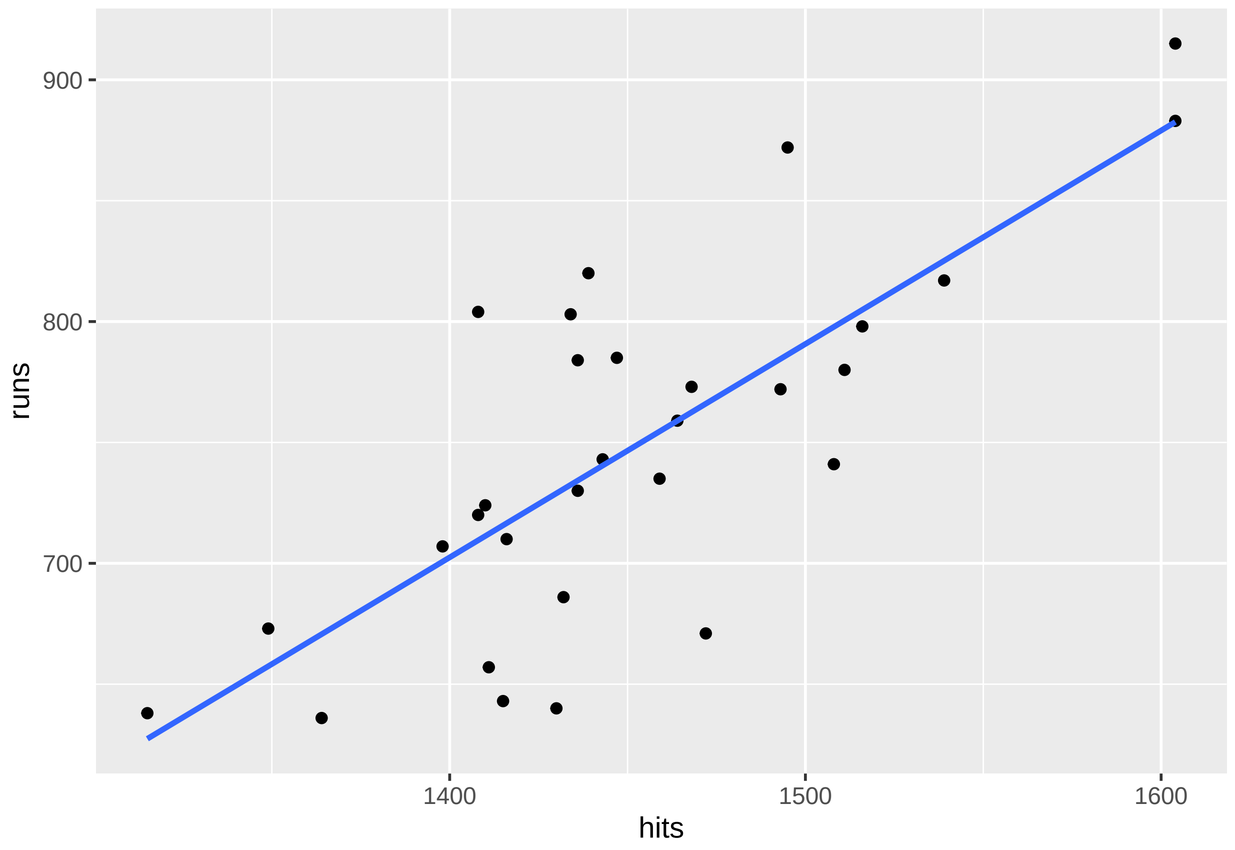
```
## $at_bats
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



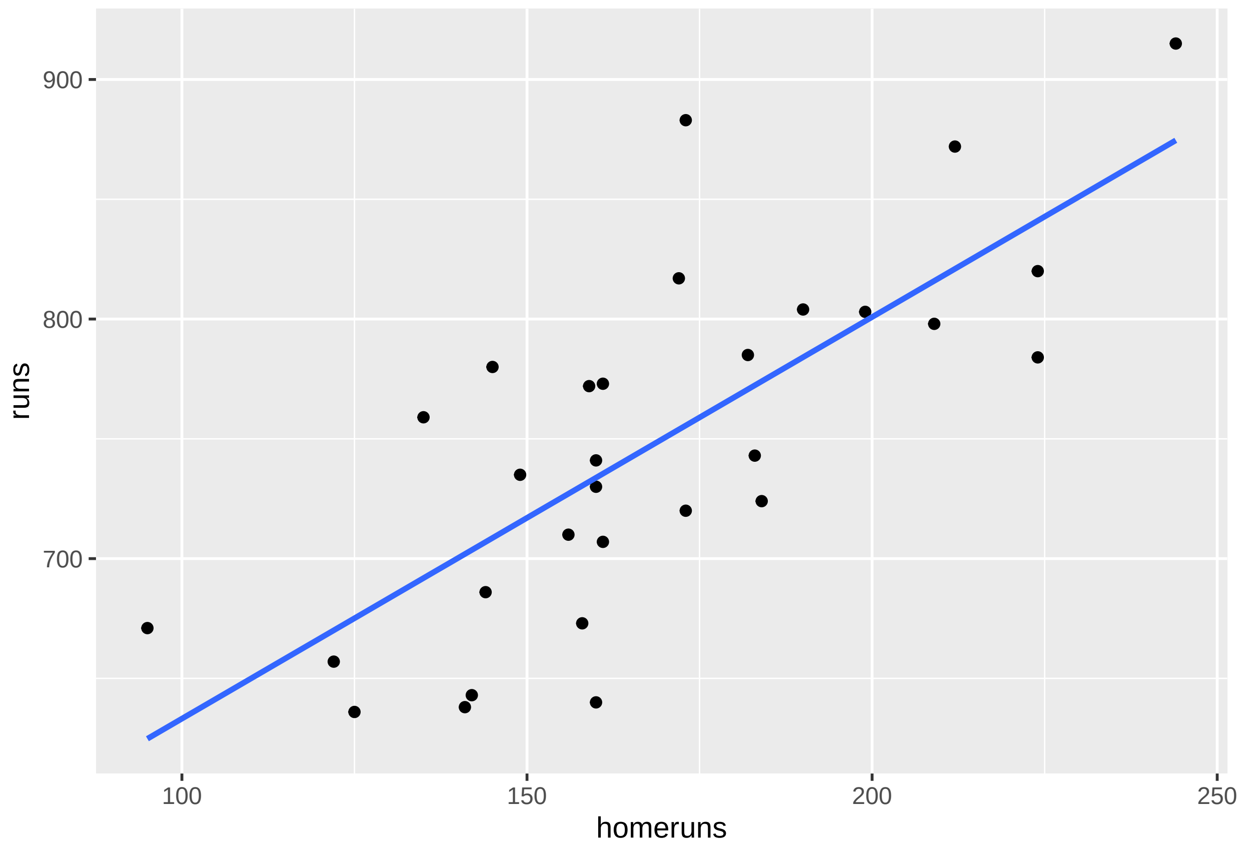
```
##
## $hits
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



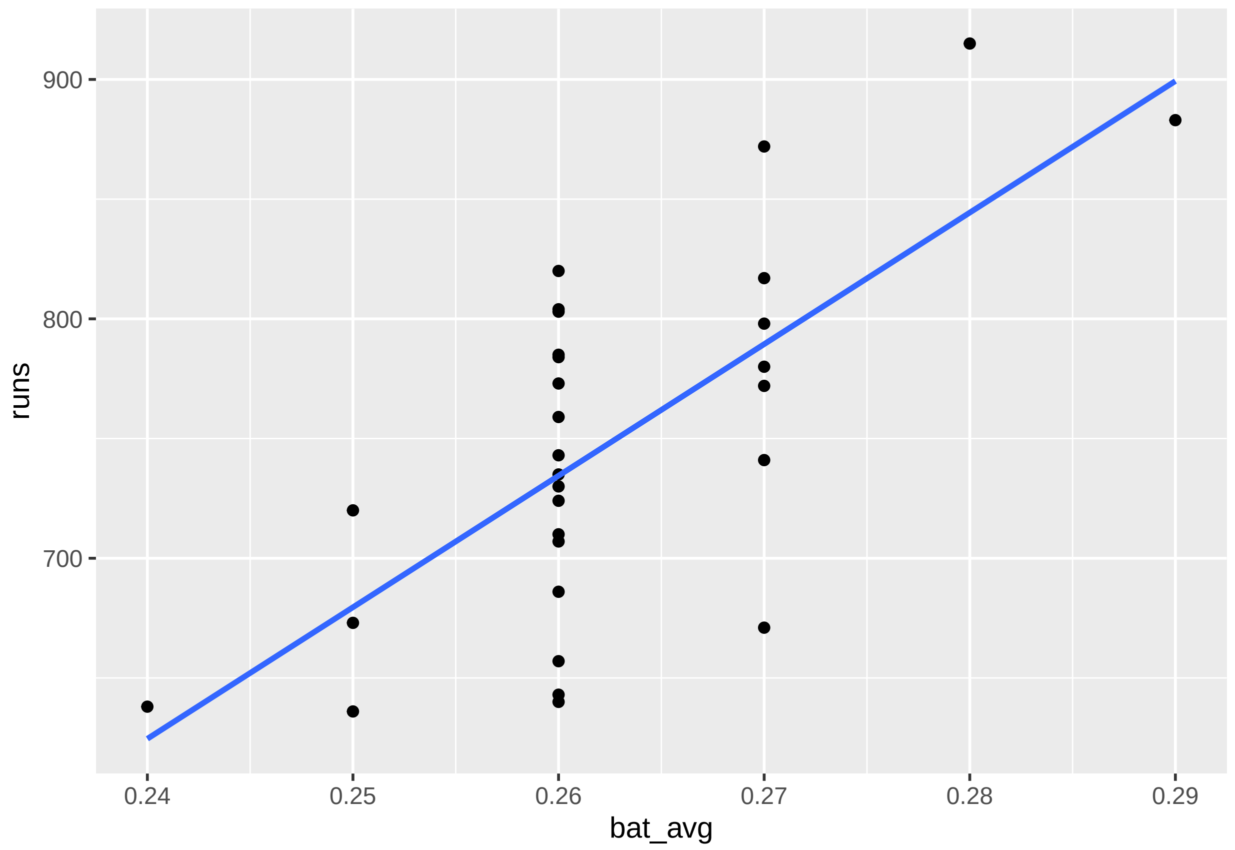
```
##
## $homeruns
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



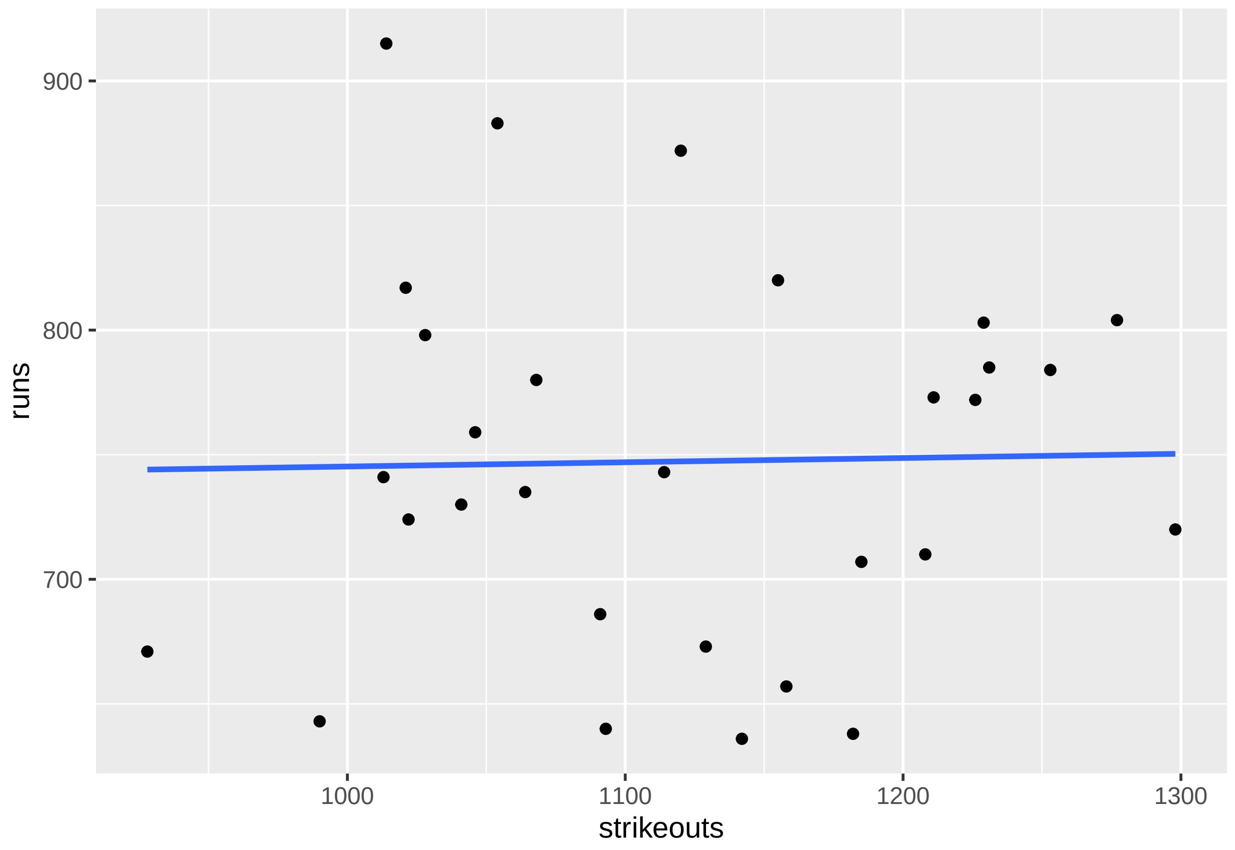
```
##
## $bat_avg
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



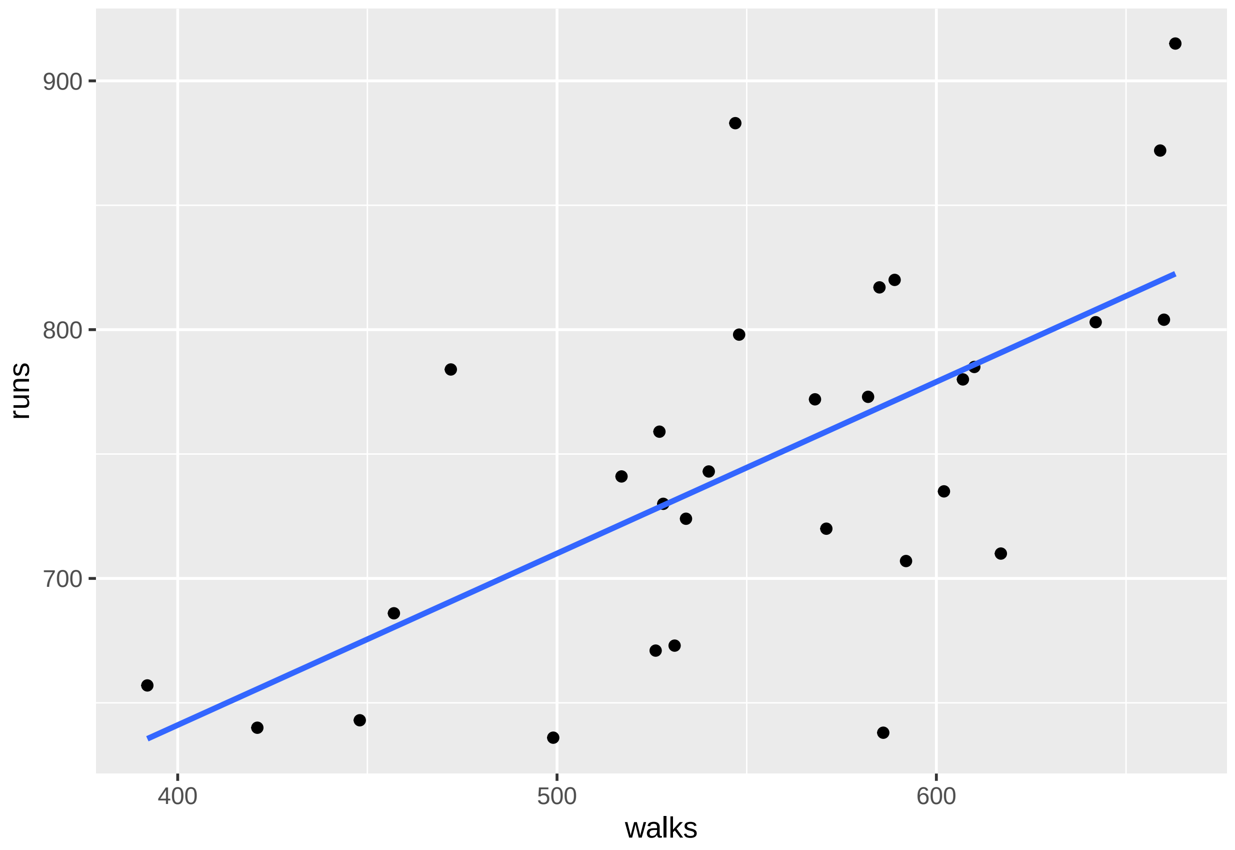
```
##
## $strikeouts
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



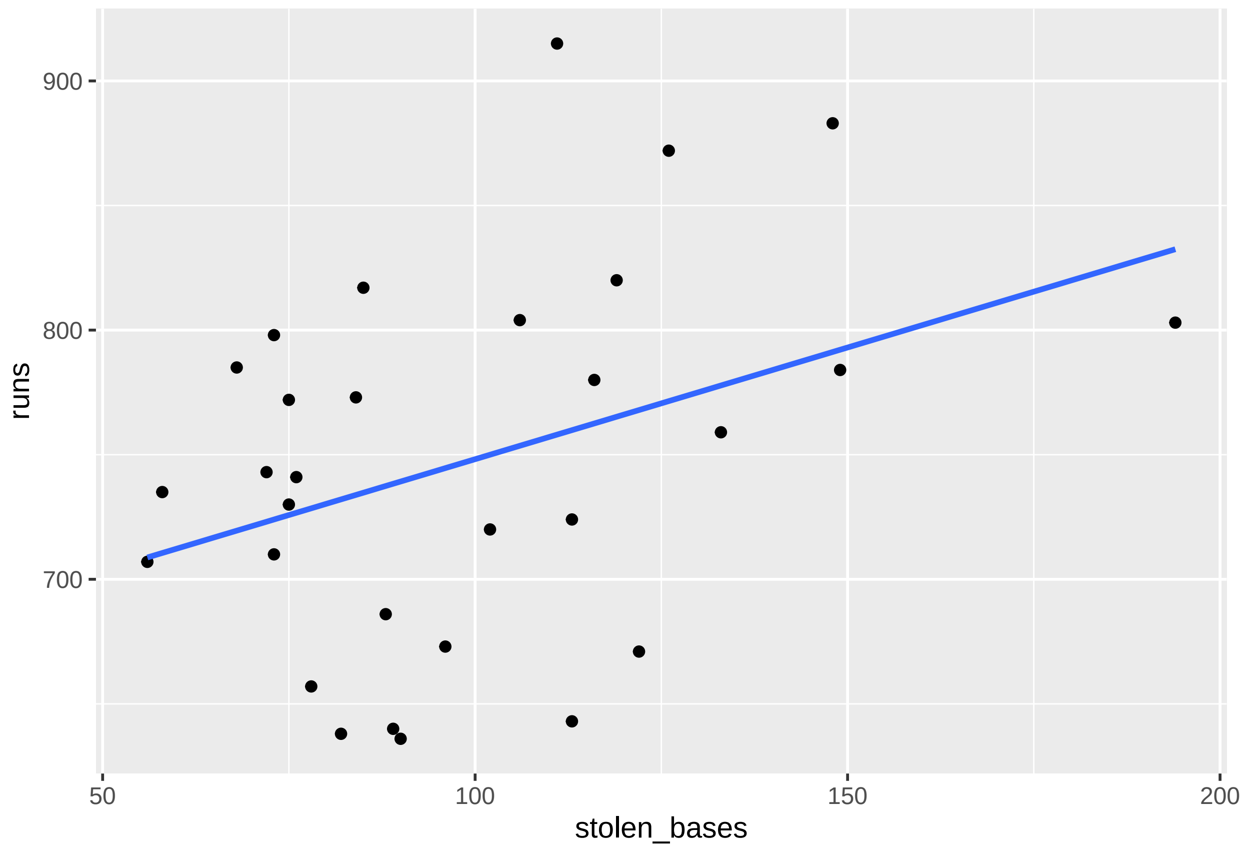
```
##
## $walks
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
##
## $stolen_bases
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The best variable for predicting runs is hits. Because it has the highest R^2 value, it is the best predictor of runs. The graph shows a positive linear relationship between hits and runs.

Alessandro (k) Apply your functions from (h) & (i) to the dataset *without using a for loop* but still using functional programming (i.e. do not repeat yourself!) over the 7 traditional variables. Which seems to be best for predicting `runs`? Does it agree with your results from (j)?

```
seven_traditional_values = c("at_bats", "hits", "homeruns", "bat_avg", "strikeouts", "walks", "stolen_ba

r_squared_values = lapply(seven_traditional_values, best_linear_model)
plots = lapply(seven_traditional_values, best_fit)

r_squared_values
```

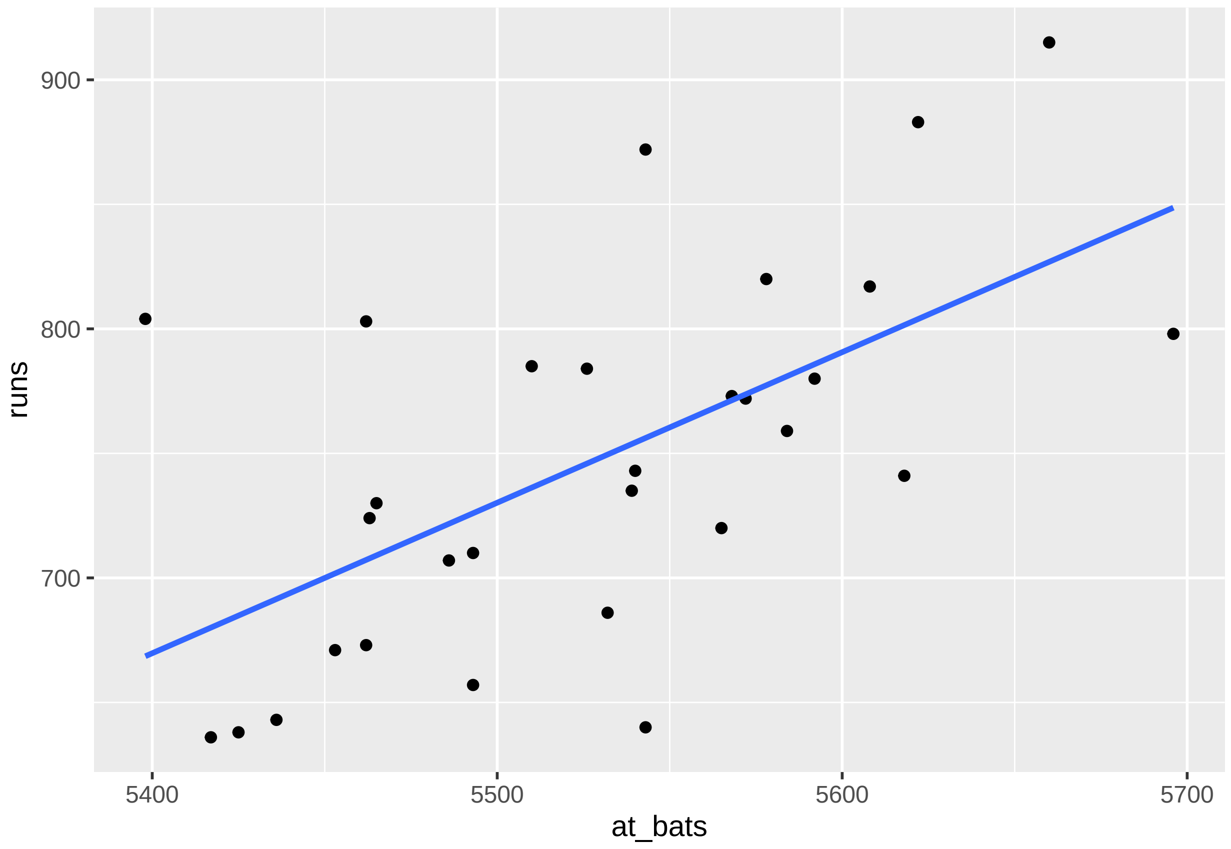
```
## [[1]]
## [1] 0.362041
##
## [[2]]
## [1] 0.5844129
##
## [[3]]
## [1] 0.5531983
##
## [[4]]
## [1] 0.4752833
##
## [[5]]
## [1] 0.0004775284
##
```

```
## [[6]]
## [1] 0.4107462
##
## [[7]]
## [1] 0.1362999
```

```
plots
```

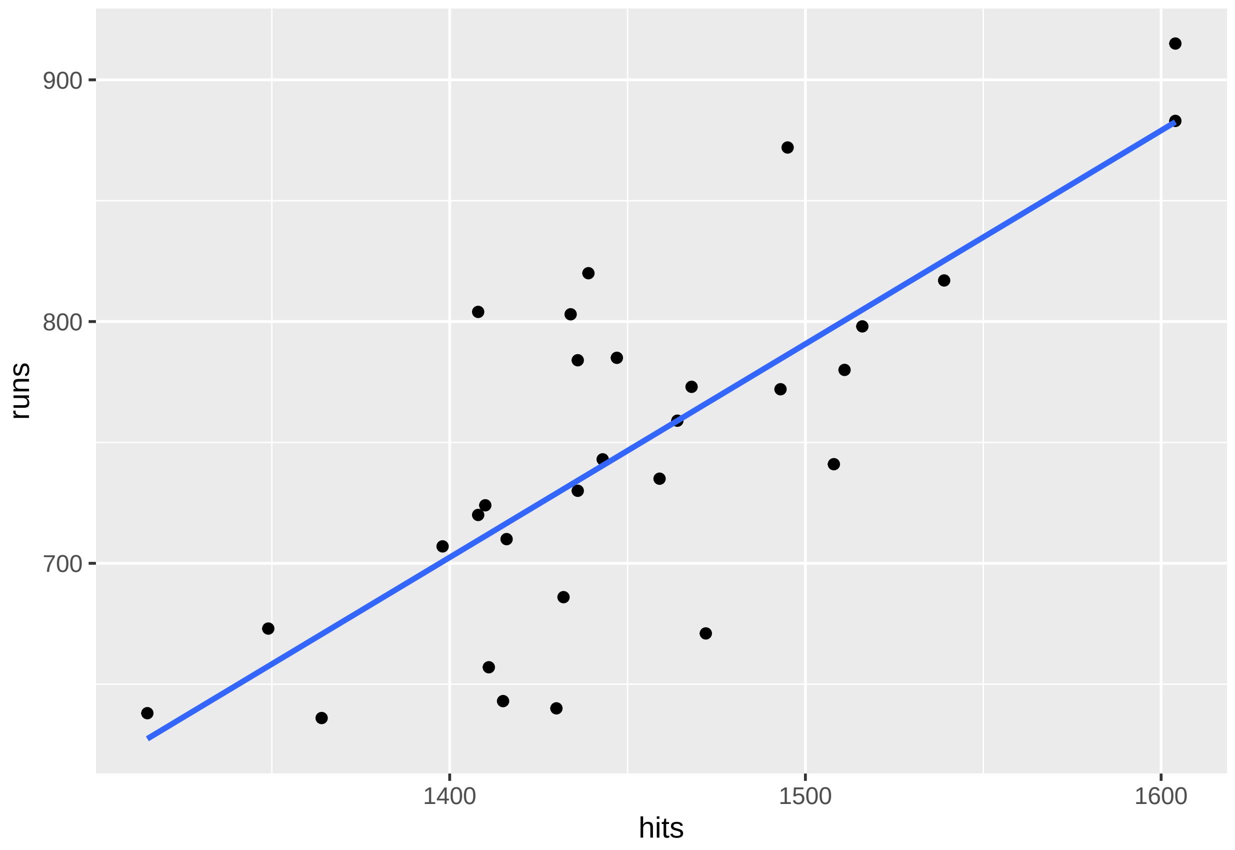
```
## [[1]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



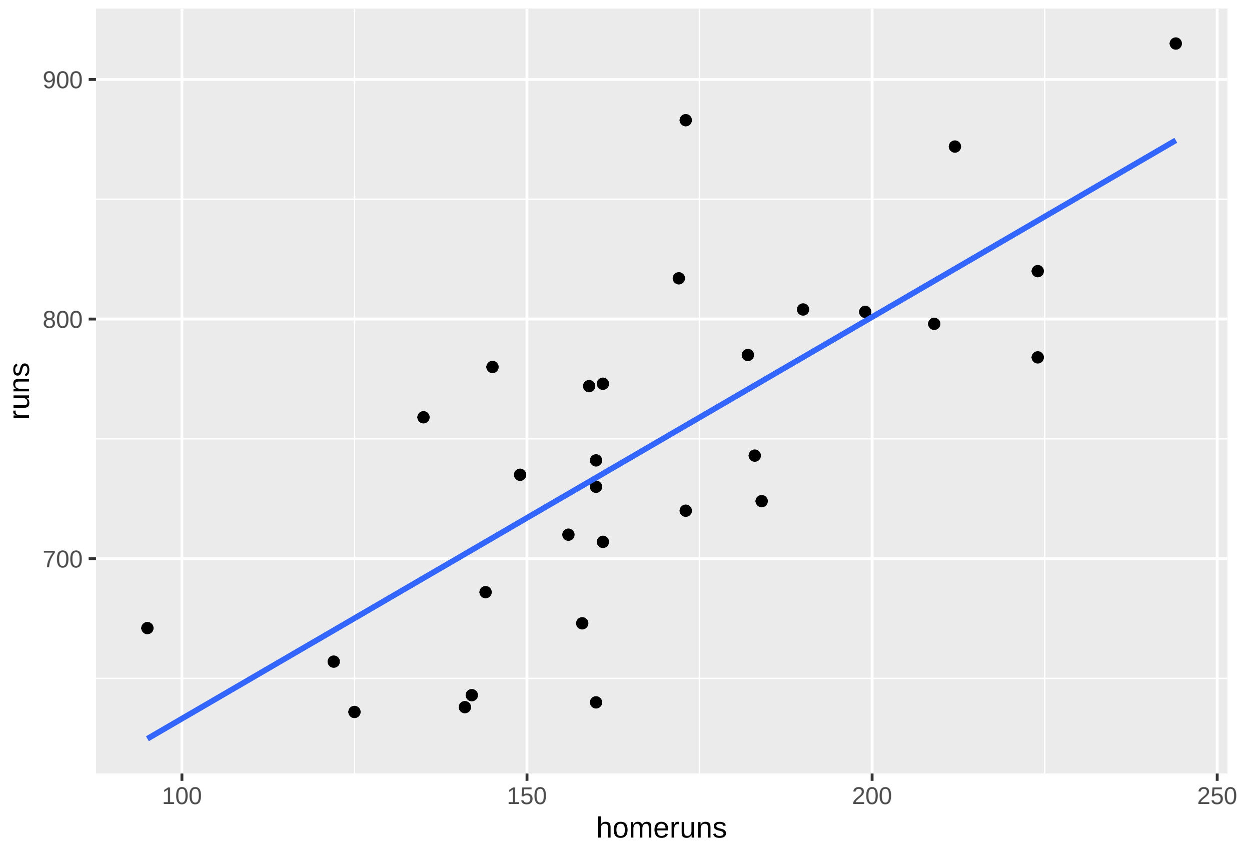
```
##
## [[2]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



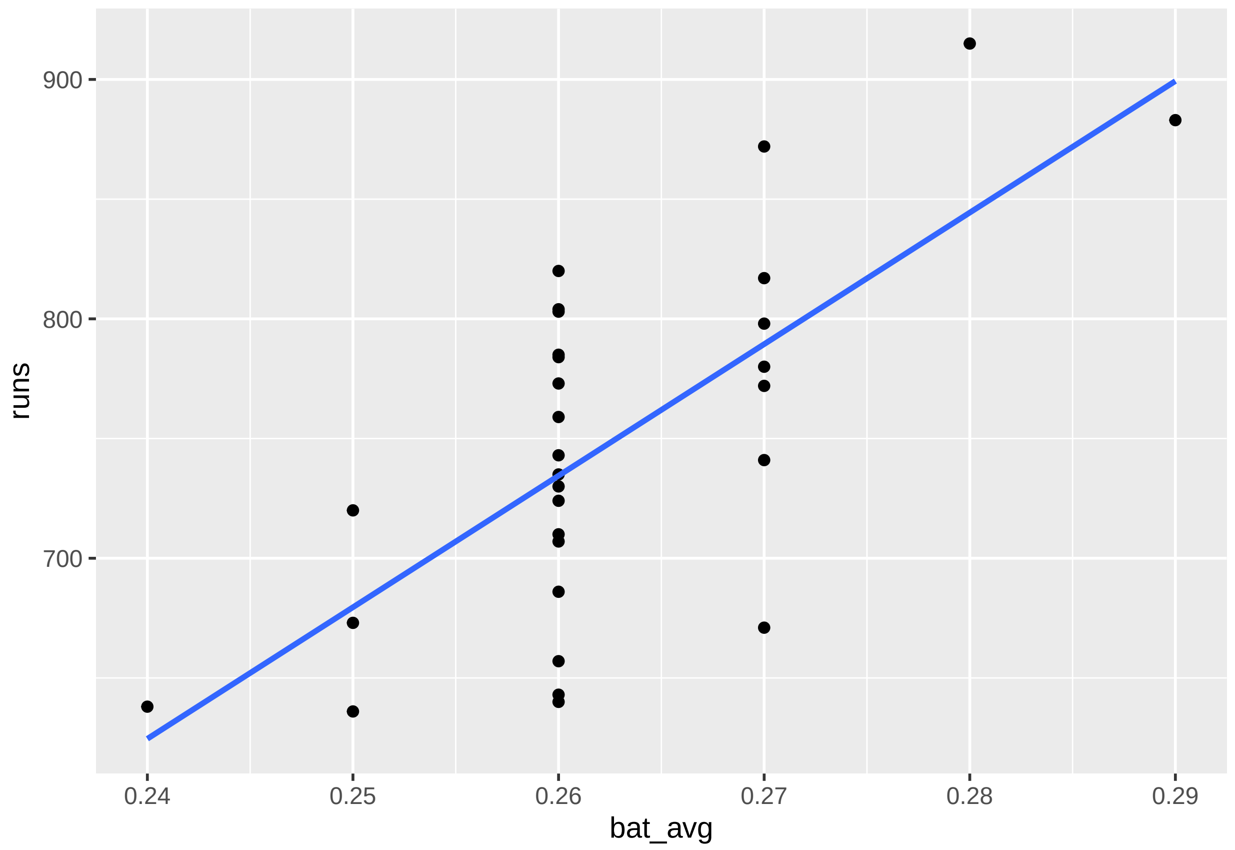
```
##
## [[3]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



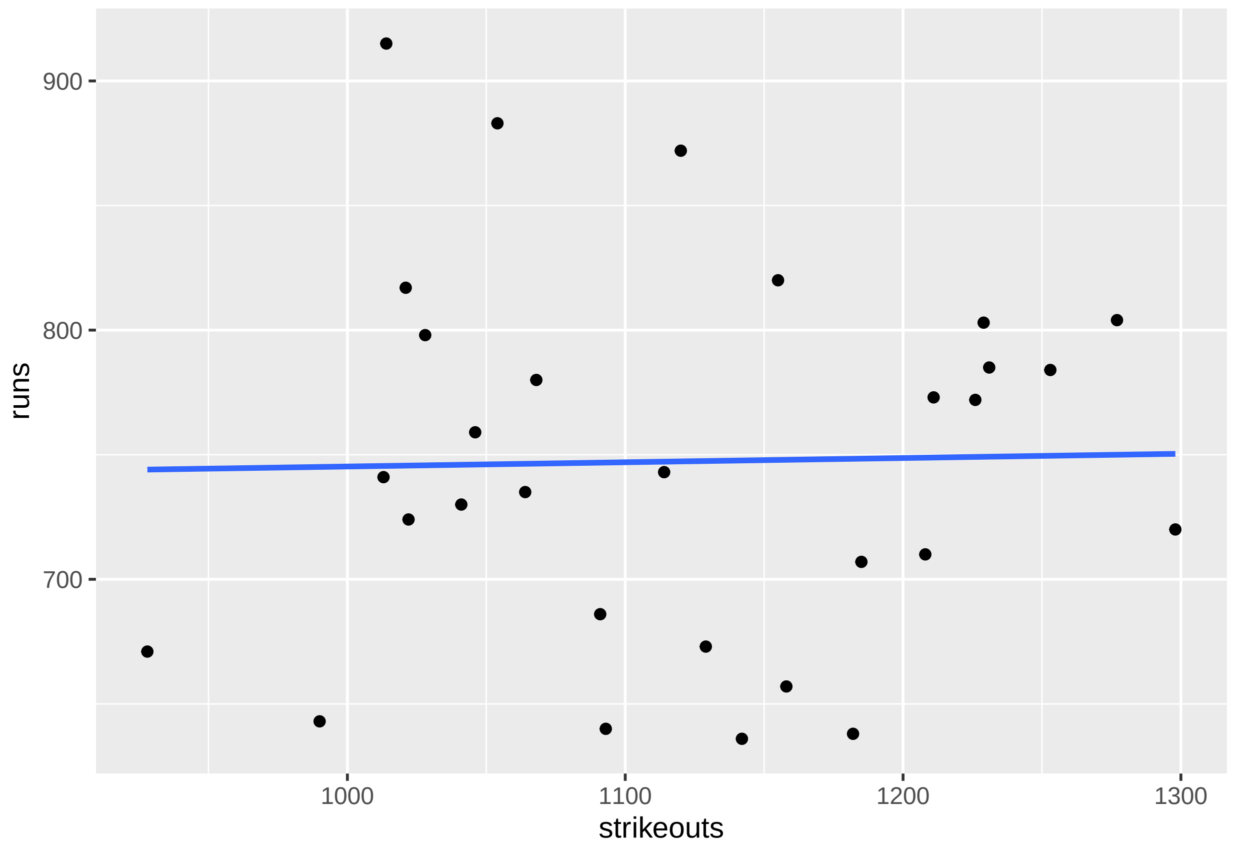
```
##
## [[4]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



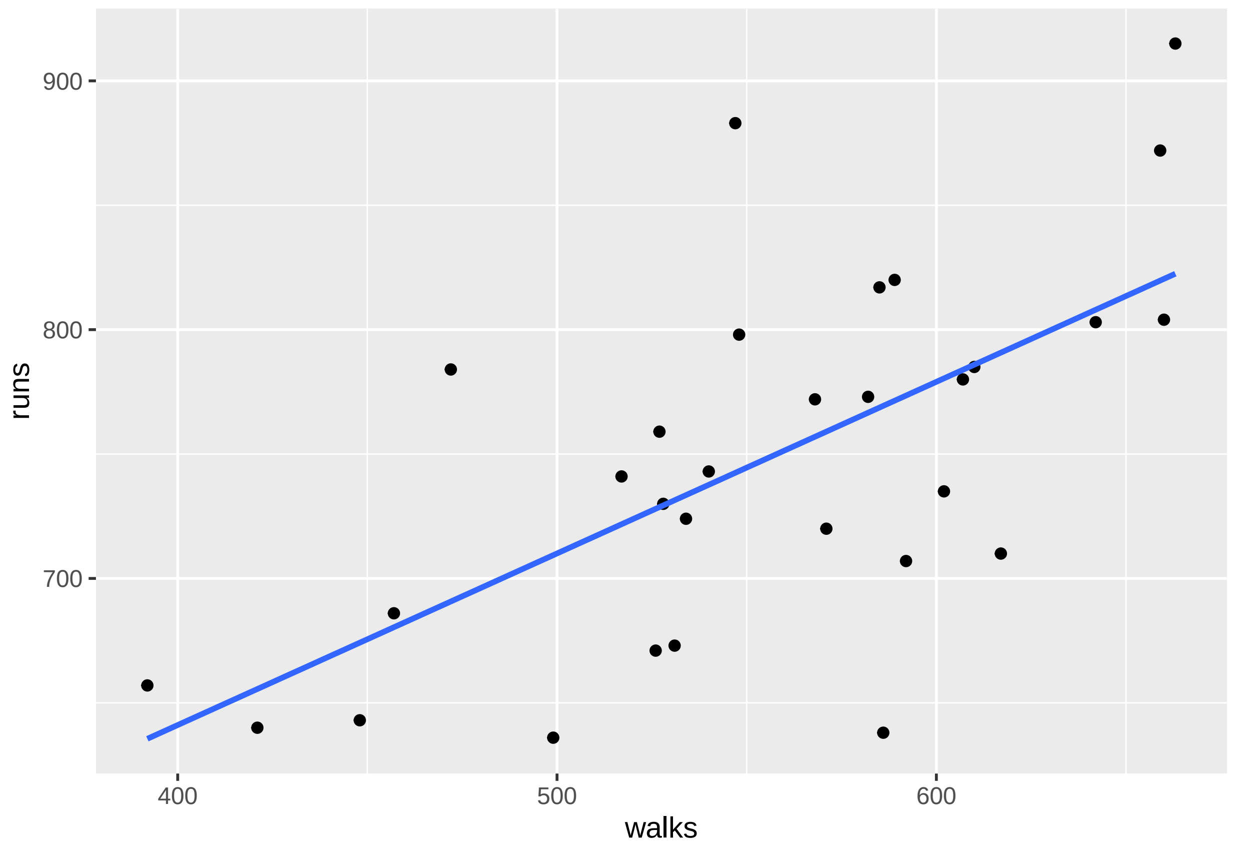
```
##
## [[5]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



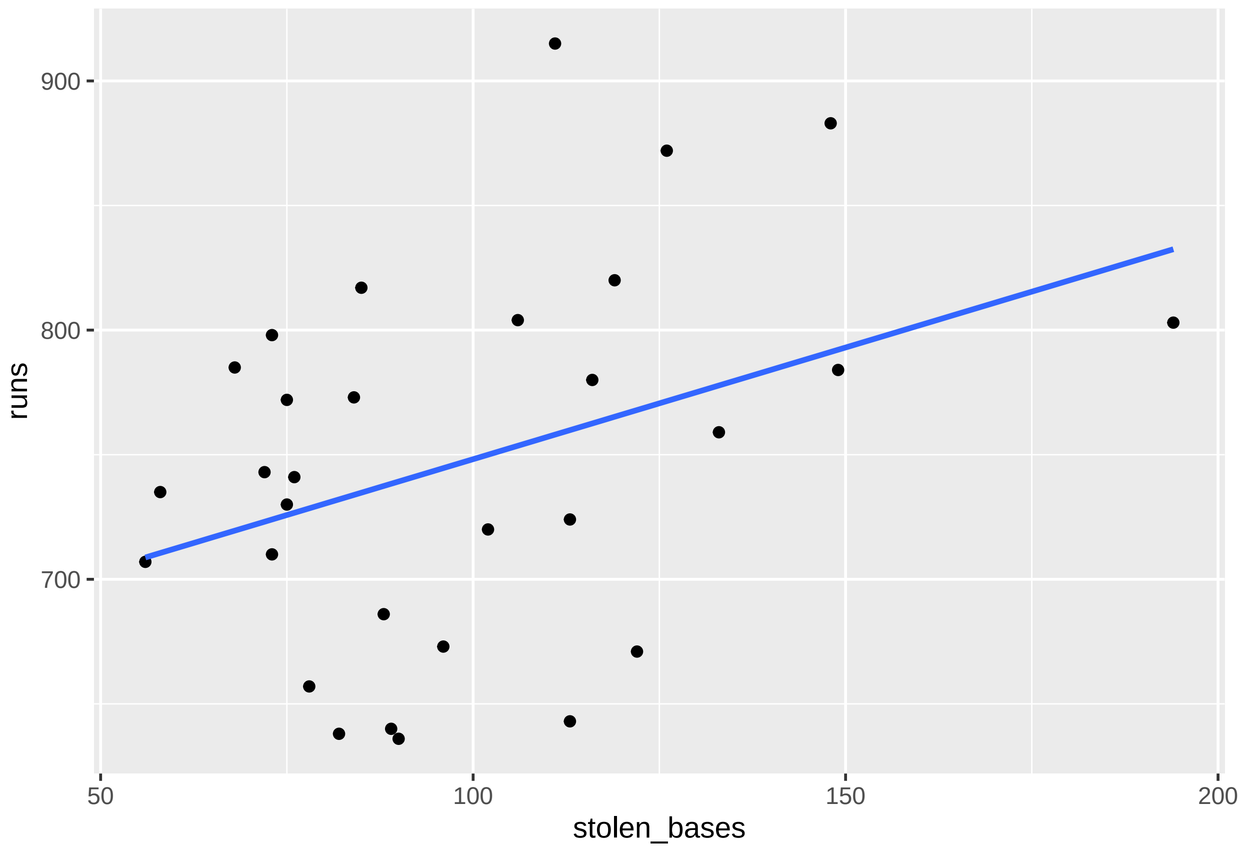
```
##
## [[6]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
##
## [[7]]
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The variable that seems best for predicting runs is hits. Because it has the highest R^2 value, it is the best predictor of runs. The graph shows a positive linear relationship between hits and runs. This agrees with the results from (j).

Done by Alexis

(l) Now imagine you are the manager of the Oakland A's in the early 2000s when Moneyball was set. You suspect that the three "non-traditional" statistics (on-base percentage, slugging percentage, and on-base plus slugging) will give you a leg up on the competition. In general, are they better or worse at predicting `runs` than the traditional variables? Which of the 3 is best? Is it better at predicting than the best "traditional" variable you found in (h)? How do you know?

```
correlation_stat = function(col_name){
  corelation_coefficient =cor(col_name, data$runs)
  coeffincient_determination = corelation_coefficient^2
  return(coeffincient_determination)
}
correlation_stat(data$on_base)
```

```
## [1] 0.6251307
```

```
correlation_stat(data$slugging)
```

```
## [1] 0.8149759
```

```
correlation_stat(data$ob_slg)
```

```
## [1] 0.9130599
```

```
non_traditional_stats = (correlation_stat(data$on_base)+
                           correlation_stat(data$slugging)+
                           correlation_stat(data$ob_slg))/3
non_traditional_stats
```

```
## [1] 0.7843888
```

```
traditional_stats = (correlation_stat(data$at_bats)+
                       correlation_stat(data$hits)+
                       correlation_stat(data$homeruns)+
                       correlation_stat(data$bat_avg)+
                       correlation_stat(data$strikeouts)+
                       correlation_stat(data$walks)+
                       correlation_stat(data$stolen_bases))/7
traditional_stats
```

```
## [1] 0.3603513
```

The non traditional stats are better at predicting runs than the traditional variables. The average coefficient of determination for the traditional stats is 0.36 whereas the average coefficient of determination for the non traditional stats is 0.78, since the non traditional stats coefficient is closer to 1 this supports that there is a relation between on__base, slugging, and ob__slg to runs. Of the three non traditional stats on__base plus slugging is the best at predicting runs since it has the coefficient of determination that is closest t0 1.0.