project3_1

Jaya Chandar Payyavula, Jean-Claude Pasay,Oscar Lomibao Jr.

November 28, 2016

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.3.2
```

```
##
## 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
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```r
library(gapminder)
```

```
## Warning: package 'gapminder' was built under R version 3.3.2
```

```r
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.3.2
```

```r
library(broom)
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.3.2
```

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.3.2
```

```r
library(cvTools)
```

```
## Warning: package 'cvTools' was built under R version 3.3.2
```

```
## Loading required package: lattice
```

```
## Loading required package: robustbase
```

```
## Warning: package 'robustbase' was built under R version 3.3.2
```

```r
library(tree)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

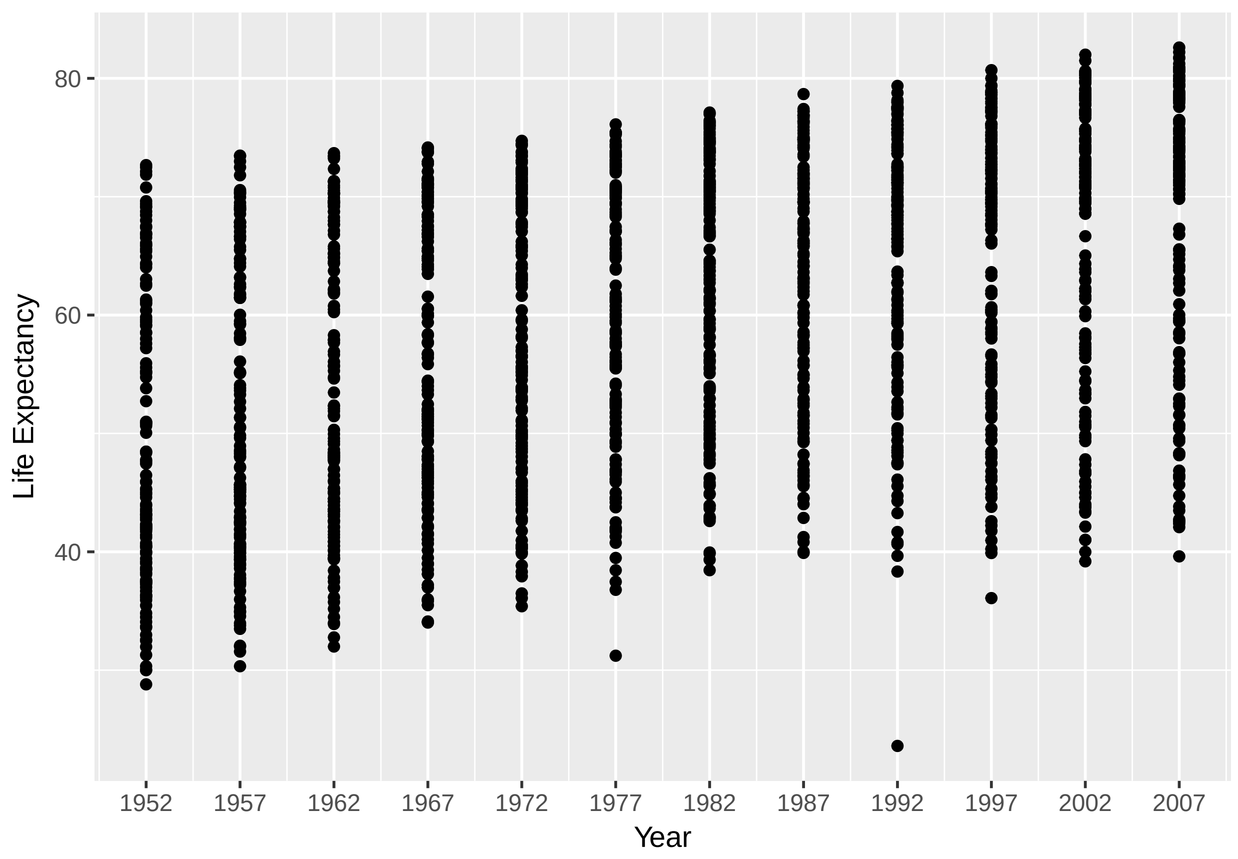
```
library(tidyr)
data(Weekly)
data(Smarket)
```

Exercise 1 So what the code below shows a few entities and then
creates a scatter plot with the x-axis as years and the y-axis as life expectancy

```
data(gapminder)
head(gapminder)
```

```
## # A tibble: 6 × 6
##       country continent  year lifeExp      pop gdpPercap
##        <fctr>    <fctr> <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan      Asia  1952  28.801  8425333  779.4453
## 2 Afghanistan      Asia  1957  30.332  9240934  820.8530
## 3 Afghanistan      Asia  1962  31.997 10267083  853.1007
## 4 Afghanistan      Asia  1967  34.020 11537966  836.1971
## 5 Afghanistan      Asia  1972  36.088 13079460  739.9811
## 6 Afghanistan      Asia  1977  38.438 14880372  786.1134
```

```
gapminder %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_point() +
  xlab("Year") +
  ylab("Life Expectancy") +
  scale_x_continuous( breaks=c(1952,1957,1962,1967,1972,1977,1982,1987,1992,1997,2002,2007) )
```



## Question 1 After plotting the scatter plot of life expectancy across time, we could see that the majority of the trend is linear. We could also see this after plotting the line of best fit (regression line) .

Question 2

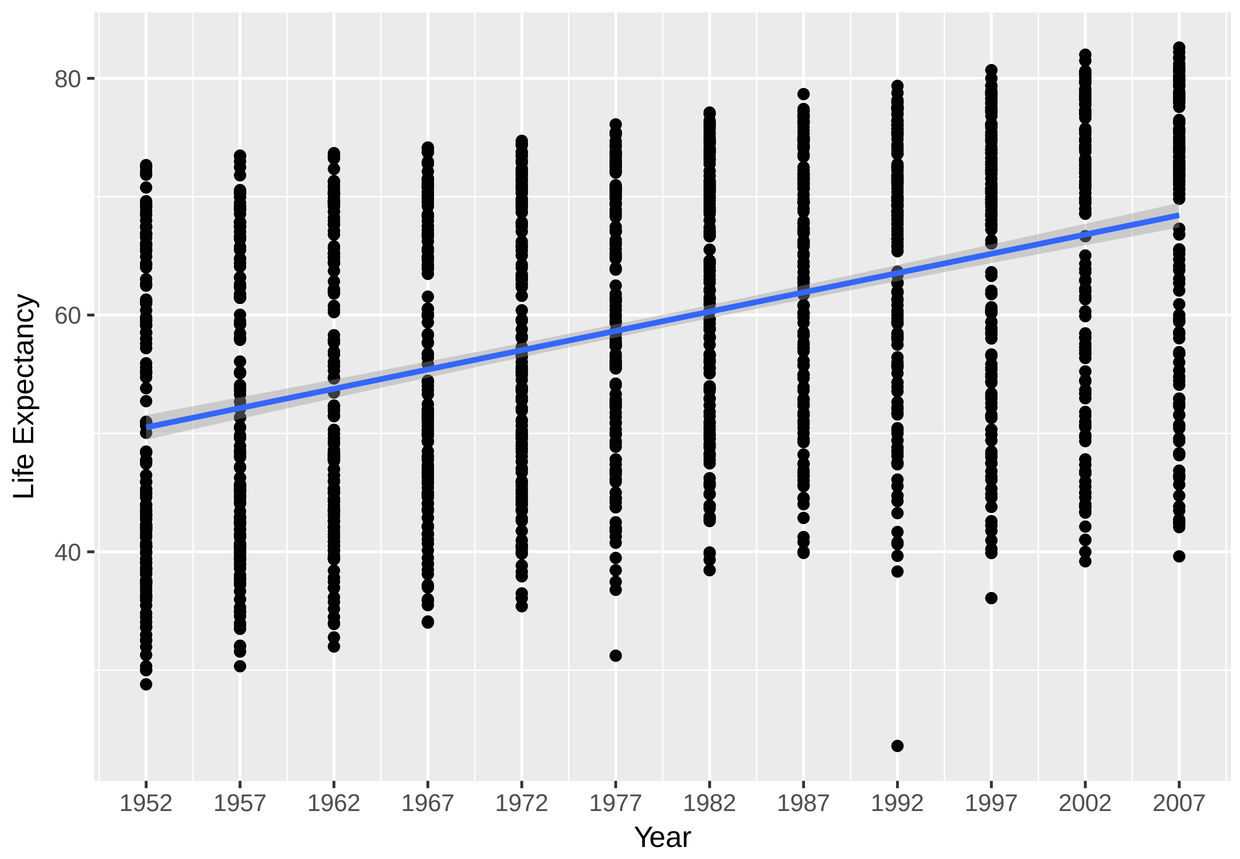
The the distribution of life expectancy seems to start off as bimodal during 1952 and becomes more skewed (unimodal) throughout the rest of the years; the distribution is not symetteric around the center.

Question 3 – (Linear Regression Model; Life Exp v. Year)

The relationship between year and life expectancy is that life expectancy increases throughout the years; we could see this by looking at the clusters of dots (life expectancy) in the graph moving up (increasing) across the y-axis; therefore, we can reject the null hypothesis and say that life expectancy and year have a relationship. You could also view this in the Linear Regression Model we plotted below.

The code below shows the original scatter plot, but shows the regression line of the plot.

```r
gapminder %>%
  ggplot(aes(x = year, y = lifeExp)) +
  geom_point() +
  scale_x_continuous( breaks=c(1952,1957,1962,1967,1972,1977,1982,1987,1992,1997,2002,2007) ) +
  geom_smooth(method=lm) +
  xlab("Year") +
  ylab("Life Expectancy")
```



## Question 4 A violin plot of residuals from the linear model in Question 3 vs. year would seem to have a bimodal distribution and may also be nearly symmetric. For example, in 1952, the distance between the lowest life expectancy and the point on the linear regression line would be the highest out of all the life expectancies from the lowest life expectancy through the linear regression line. Hence, the violin plot for this half would look skewed. We go through this process again for the other half (highest life expectancy and the point on the linear regression line) and get another skewed distribution. Therfore, the entire violin plot for 1952 will have a bimodal distribution; the distribution will become skewed across the years (simlar to Question 2).

Question 5

According to the assumptions of the linear regression model, the violin plot SHOULD show that 1952 has a high the amount of countries that have a low life expectancy. As years pass by, the violin plot should show that the cluster of data moves up the y-axis (increases), which indicates that the majority of countries are increasing their life expectancy throughout the years.

Exercise 2 The code below creates a linear model between the variables year and life expectancy.

```r
library(broom)
auto_fit <- lm(gapminder$year~gapminder$lifeExp, data = gapminder)
exer2 <- broom::tidy(auto_fit)
exer2
```

```
##                term     estimate  std.error  statistic      p.value
## 1       (Intercept) 1944.8710743 1.77488709 1095.77172 0.000000e+00
## 2 gapminder$lifeExp    0.5822489 0.02916335   19.96509 7.546795e-80
```

Question 6

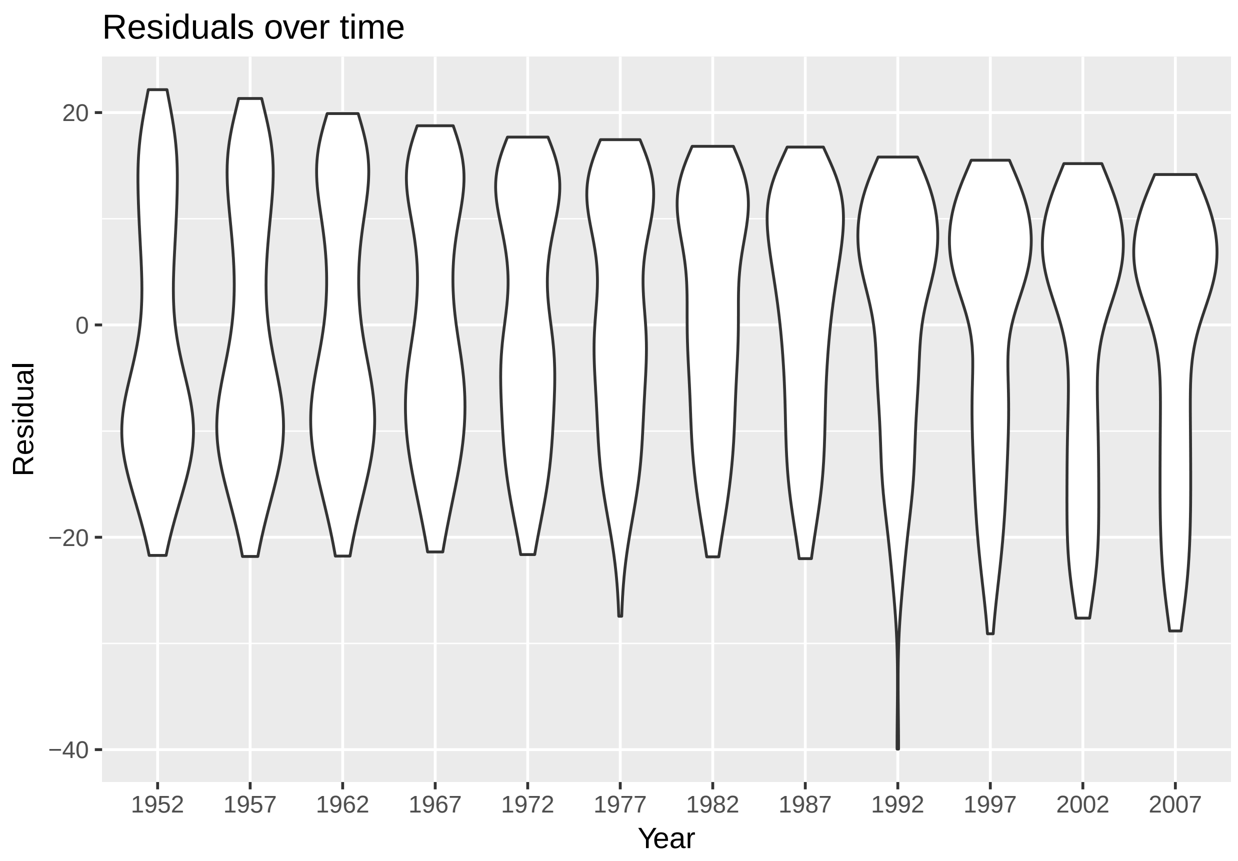
According to the output above, life expectancy increases every year by 0.5822

Question 7

No, I reject the null hypothesis as there is a relation between life expectancy and years. Life expectancy increases over the years.

Exercise 3 The code below first creates a linear model with year and life expectancy and then create a data frame from the linear model. Then to remove any negative values we used the absulte function to change them to positive. Then we grabbed the year from gapminder and inserted it into the new dataframe. Next we created a violin plot of residuals vs year using the new data frame. And then we passed linear_model through broom::augment.

```
m1 <- lm(gapminder$lifeExp ~ gapminder$year)
list <- as.data.frame(resid(m1)) # List of residuals
list <- abs(list)
list$year <- gapminder$year

list %>%
  ggplot(aes(x=factor(year), y=resid(m1))) +
    geom_violin() +
    labs(title="Residuals over time",
        x = "Year",
        y = "Residual")
```



```r
head(broom::augment(auto_fit))
```

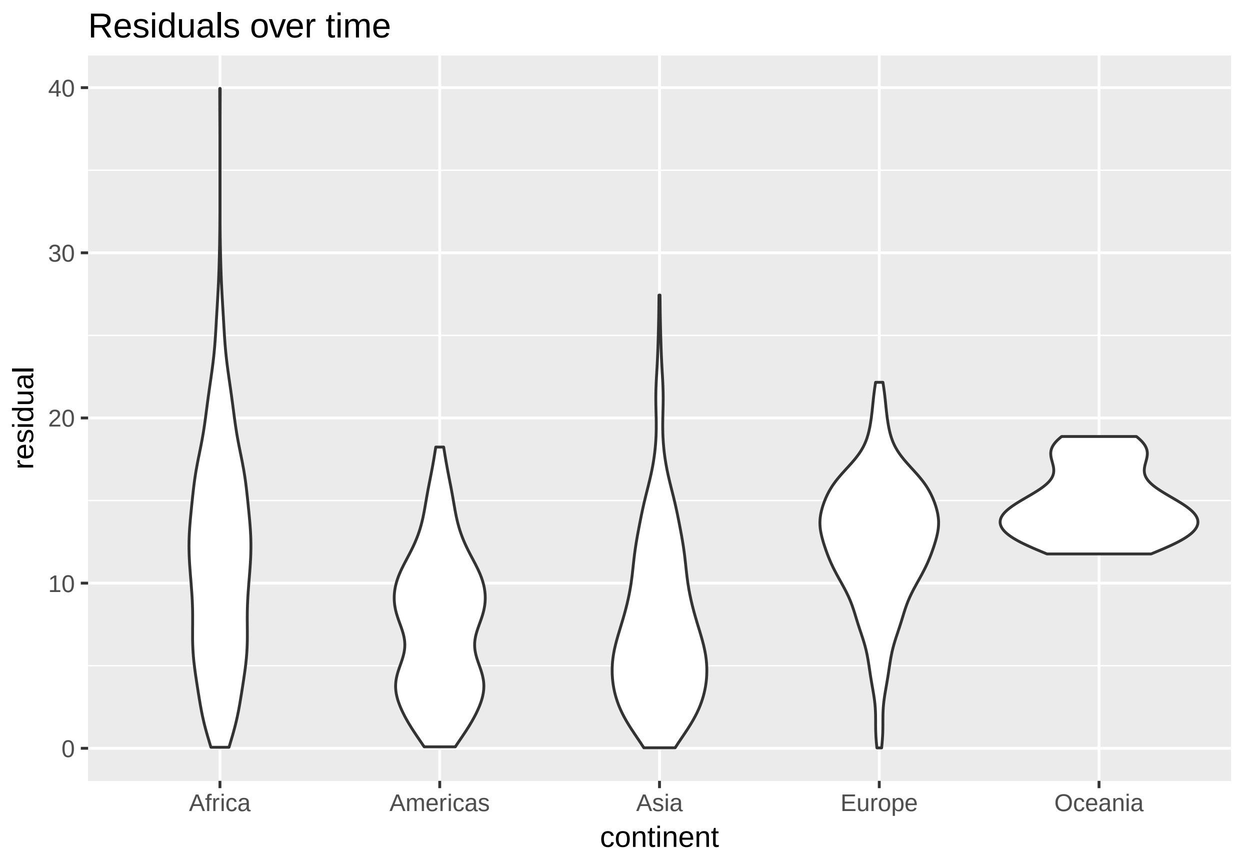
```
##   gapminder.year gapminder.lifeExp  .fitted   .se.fit    .resid
## 1           1952            28.801 1961.640 0.9705805 -9.640424
## 2           1957            30.332 1962.532 0.9295909 -5.531847
## 3           1962            31.997 1963.501 0.8854156 -1.501292
## 4           1967            34.020 1964.679 0.8323990  2.320819
## 5           1972            36.088 1965.883 0.7790924  6.116728
## 6           1977            38.438 1967.252 0.7198597  9.748443
##          .hat   .sigma      .cooksd  .std.resid
## 1 0.003898011 15.54850 7.554013e-04 -0.62134702
## 2 0.003575722 15.54968 2.280161e-04 -0.35648233
## 3 0.003243951 15.55022 1.522568e-05 -0.09672987
## 4 0.002867101 15.55016 3.213434e-05  0.14950465
## 5 0.002511643 15.54955 1.954027e-04  0.39396282
## 6 0.002144251 15.54846 4.234095e-04  0.62775671
```

Question 8

Yes

Exercise 4 The code below creates a new data frame with the linear_model2 and removes all negative values. Then we created a new empty data frame with 1704 rows and added continents and residuals from gapminder and the new data frame we made from linear_model2. We also removed one column because it isn't needed. Then we created a boxplot model of residuals vs. continent.

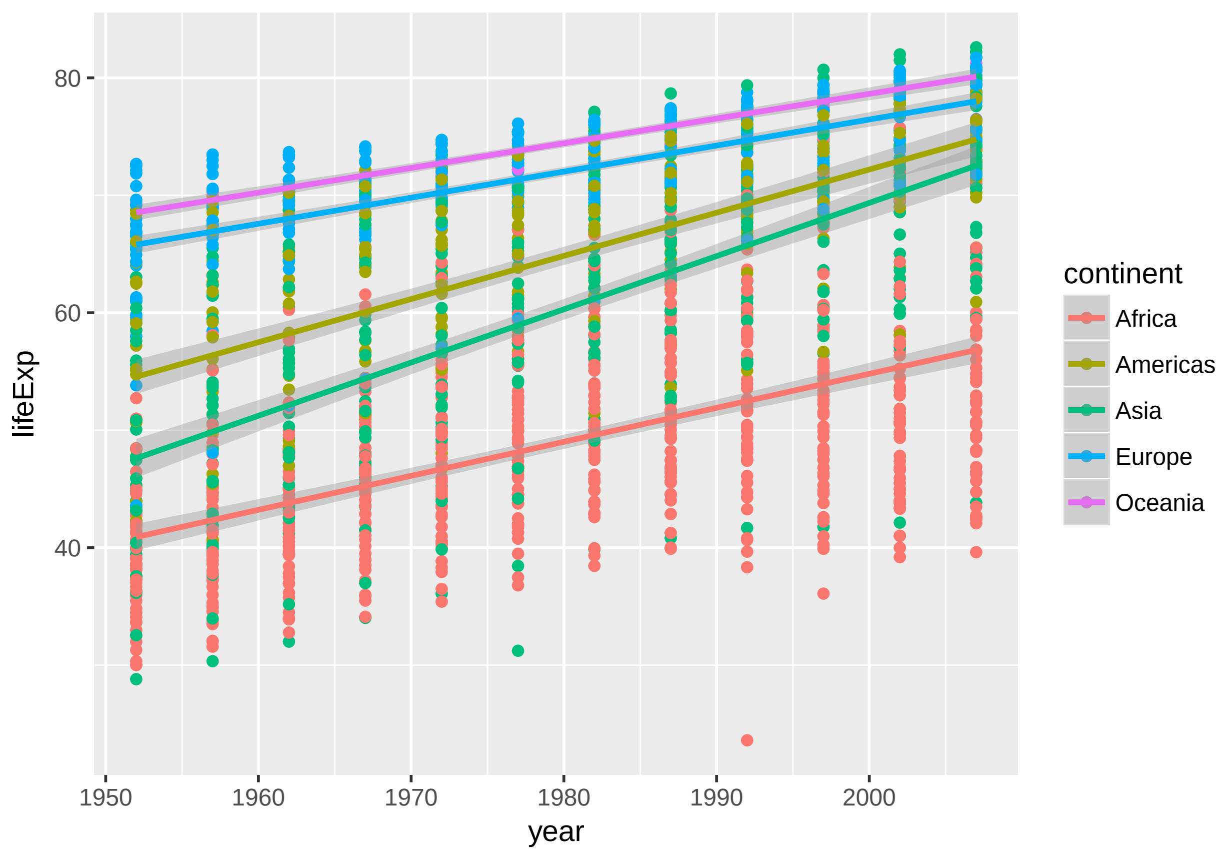
```r
m2 <- lm(gapminder$lifeExp ~ gapminder$year)
list2 <- as.data.frame(resid(m2)) # List of residuals
list2 <- abs(list2)
list2$continet <- gapminder$continent
colnames(list2) <- c("residuals", "continent")

list2 %>%
  ggplot(aes(x=factor(continent), y= residuals)) +
    geom_violin() +
    labs(title="Residuals over time",
        x = "continent",
        y = "residual")
```



## Question 9 There is no dependency between model residuals and continent.

Exercise 5 The code below creates a scatterplot of year and life expectancy with a linear regression of the relationship.

```
gapminder %>%
ggplot(aes(x = year, y = lifeExp, color = continent)) + geom_point() +
  geom_smooth(method=lm)
```



Question 10

Yes, we need to include an interatction term between continent and year. Adding an interaction term between continent and model will help in expanding the relationship between these two variables and it will allow to test more hypotheses.

Exercise 6 The code below use the interaction to create a factor that has the interaction between year and continent. Then using lm we created a linear model with life expectancy and the interact factor table. Then we broomed it.

```
linear_model <- lm(gapminder$lifeExp~gapminder$continent*gapminder$year)
result6 <- broom::tidy(linear_model)
as.data.frame(result6)
```

```
##                                         term      estimate   std.error
## 1                                (Intercept) -524.25784607  32.96342596
## 2                 gapminder$continentAmericas -138.84844718  57.85057778
## 3                     gapminder$continentAsia -312.63304922  52.90355242
## 4                   gapminder$continentEurope  156.84685210  54.49775866
## 5                  gapminder$continentOceania  182.34988290 171.28298566
## 6                              gapminder$year    0.28952926   0.01665177
## 7  gapminder$continentAmericas:gapminder$year    0.07812167   0.02922373
## 8      gapminder$continentAsia:gapminder$year    0.16359314   0.02672470
## 9    gapminder$continentEurope:gapminder$year   -0.06759712   0.02753003
## 10  gapminder$continentOceania:gapminder$year   -0.07925689   0.08652512
##      statistic      p.value
## 1  -15.9042281 3.436134e-53
## 2   -2.4001220 1.649695e-02
## 3   -5.9094907 4.139916e-09
## 4    2.8780423 4.051687e-03
## 5    1.0646118 2.872034e-01
## 6   17.3872996 1.953998e-62
## 7    2.6732271 7.584665e-03
## 8    6.1214213 1.149941e-09
## 9   -2.4553961 1.417280e-02
## 10  -0.9159986 3.597980e-01
```

```r
head(result6)
```

```
##                          term     estimate    std.error  statistic
## 1                 (Intercept) -524.2578461  32.96342596 -15.904228
## 2 gapminder$continentAmericas -138.8484472  57.85057778  -2.400122
## 3     gapminder$continentAsia -312.6330492  52.90355242  -5.909491
## 4   gapminder$continentEurope  156.8468521  54.49775866   2.878042
## 5  gapminder$continentOceania  182.3498829 171.28298566   1.064612
## 6              gapminder$year    0.2895293   0.01665177  17.387300
##        p.value
## 1 3.436134e-53
## 2 1.649695e-02
## 3 4.139916e-09
## 4 4.051687e-03
## 5 2.872034e-01
## 6 1.953998e-62
```

Question 11

Not all parameters are significantly different from zero. Expecially the Pvalues are very close to zero. ## Question 12

```r
result6 <- result6[-c(1,2,3,4,5),]
subset(result6, select = c("term", "estimate"))
```

```
##                                         term    estimate
## 6                             gapminder$year  0.28952926
## 7  gapminder$continentAmericas:gapminder$year  0.07812167
## 8      gapminder$continentAsia:gapminder$year  0.16359314
## 9    gapminder$continentEurope:gapminder$year -0.06759712
## 10  gapminder$continentOceania:gapminder$year -0.07925689
```

The life-expectancy in Africa, Americas and Asia have increased by 0.2895,0,0781,0.163 respectively. But in Europe and Oceania it increased by -0.0675 and -0.0792. **Exercise 7** The bottome code uses anvoa to compare the two models together and prints out the

```r
result7 <- anova(auto_fit, linear_model)
```

```
## Warning in anova.lmlist(object, ...): models with response '"gapminder
## $lifeExp"' removed because response differs from model 1
```

```r
result7
```

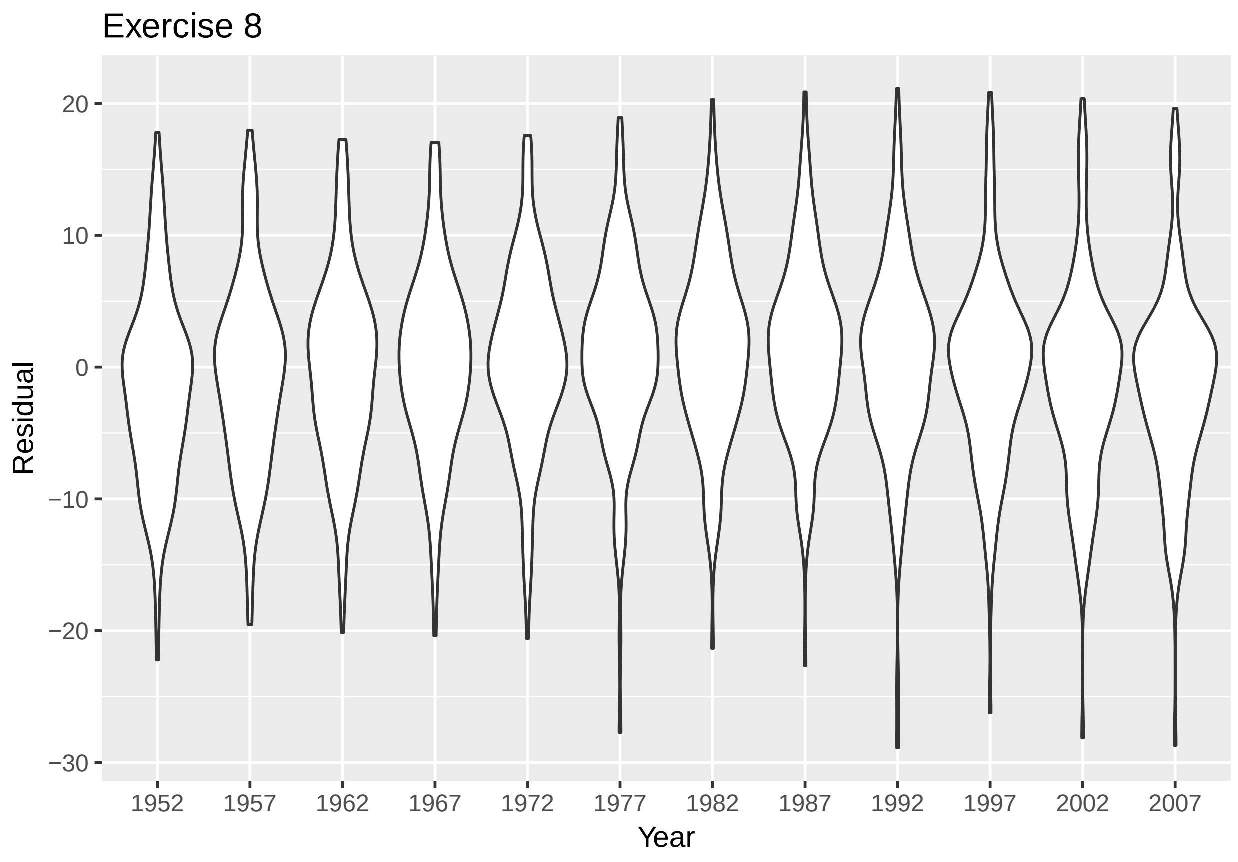
```
## Analysis of Variance Table
##
## Response: gapminder$year
##                       Df Sum Sq Mean Sq F value    Pr(>F)    
## gapminder$lifeExp    1  96330   96330   398.6 < 2.2e-16 ***
## Residuals         1702 411320     242                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 13

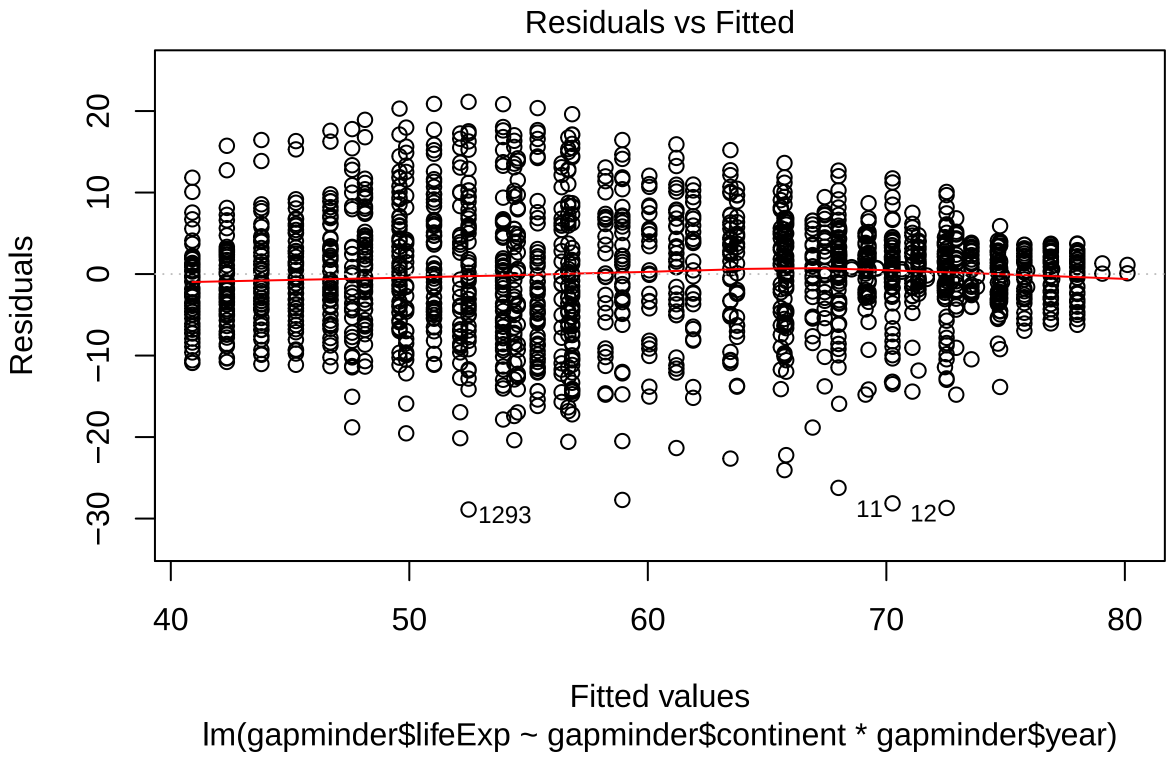
Looking at the F value and comparing it to a confidence of 95% we see that the F value is significantly smaller than .05 meaning that the interaction model is much more accurate.

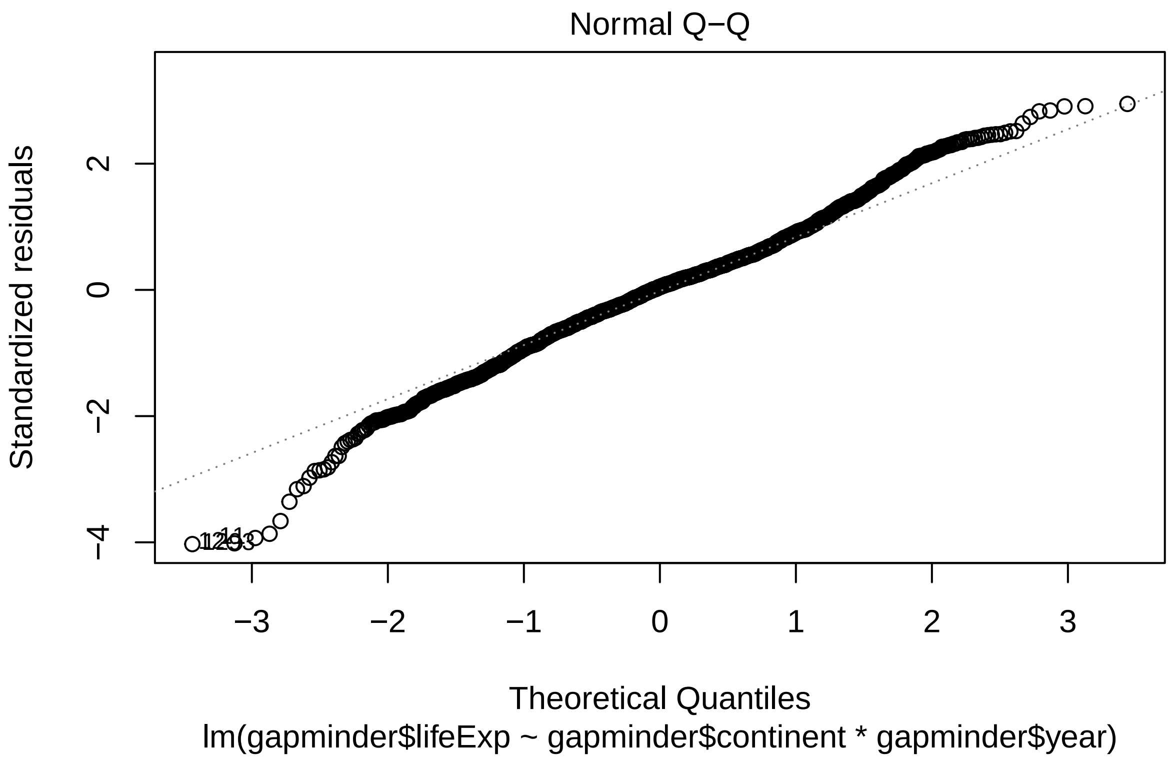
Exercise 8 The code below creates a residuals vs year violin plot for the interaction model. ##############Add comment#########################

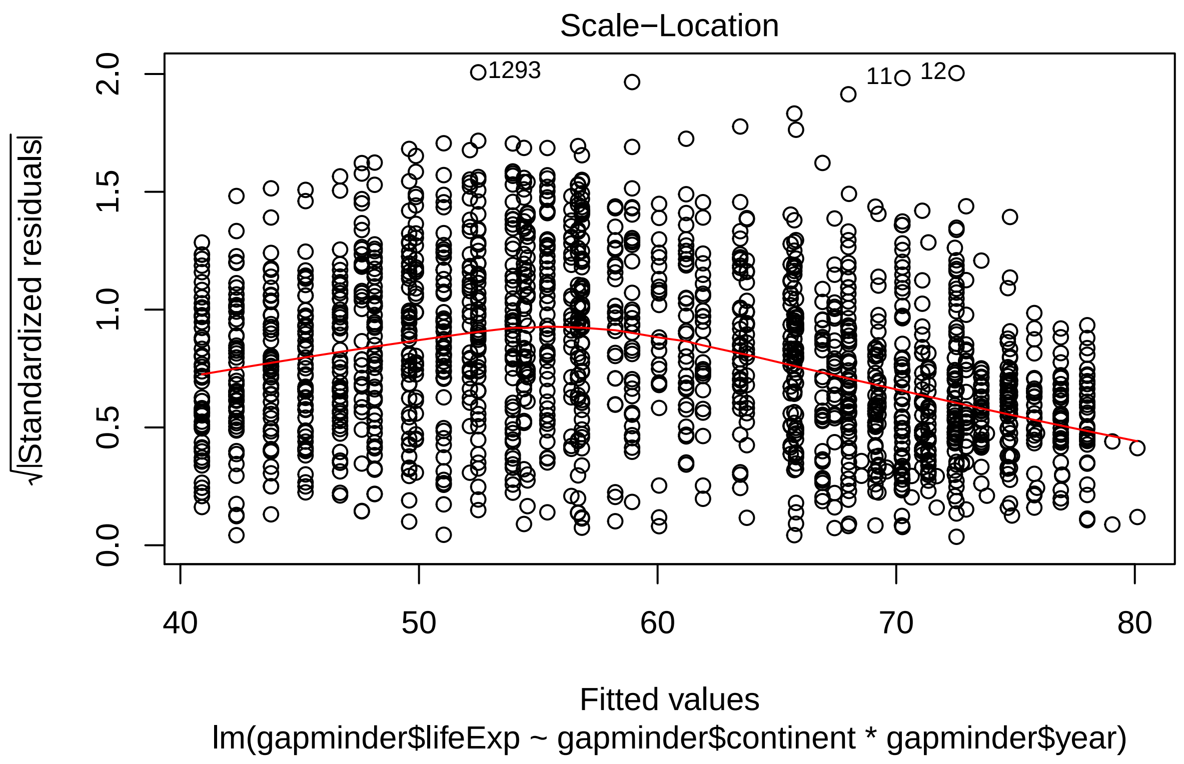
```
lin_aug <- linear_model %>% augment()

lin_aug %>%
ggplot(aes(x = factor(gapminder$year), y = .resid)) +
  geom_violin() +
  labs(title = "Exercise 8", x = "Year", y = "Residual")
```



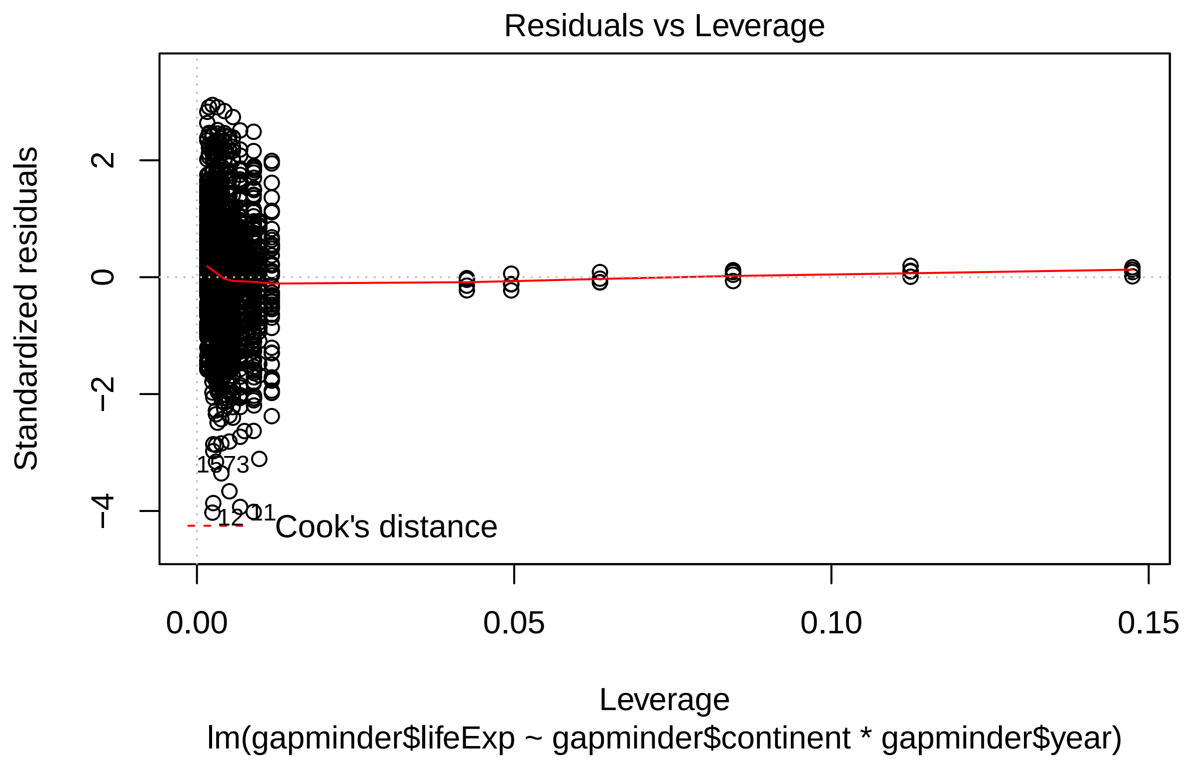
```
plot(linear_model)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(gapminder$lifeExp ~ gapminder$continent * gapminder$year)
−4
−2
0
2
−3
−2
−1
0
1
2
3
1293
1211



Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
40
50
60
70
80
1293
11
12
Fitted values
lm(gapminder$lifeExp ~ gapminder$continent * gapminder$year)



answer for exercise 8

Since the interaction model makes the variables more consistent, our interaction model is in fact more consistent/accurate than the linear regression model due to the fact that there are more smaller valued residuals (closer to zero); this also means that the residuals are more compact (together) throughout time; whereas the residuals of the linear regression model increases over time. The plot for residuals vs fittied is also consistent as the points are evely distributed on both sides of the line.

Part 2

Question 1 For the code on the bottom we used the gradient descent you used from the lecture notes. So throughout we update the beta estimate until we get a convergence, when the difference reach 0 or we do 50 iterations. This function returns the betas calculated.

```
# Implementation of gradient descent for least squares regression
# for a single predictor (x)
#
# There is some code here that is only used to generate illustrative plots and would not be part of rea
gradient_descent <- function(x, y, tol=1e-6, maxit=50) {
  # initialize estimate
  beta <- rep(0, (ncol(x)+1))
  old_beta <- Inf; i <- 0; beta_keep <- NA


  # starting step size
  alpha <- 1e-3
  difference <- Inf
  # check for convergence
  # (in practice, we do include a limit on the number of iterations)
  while ((difference > tol) && (i < maxit) && !is.na(difference)) {

    # store the last estimate to check convergence
    old_beta <- beta

    beta_col <- cbind(1,x)
    # update estimate
    f <- beta_col %*% beta

    updates = rep(0, ncol(beta_col))
    for(k in seq(1,nrow(beta_col))) {
      updates <- updates + (y[k]-f[k])*beta_col[k,]
    }

    beta <- beta + alpha * updates

    # compute difference after taking step
    # to check convergence
    sum_beta <- sum(beta)
    sum_oldbeta <- sum(old_beta)
    difference <- (sum_beta - sum_oldbeta)^2 / (sum_oldbeta)^2

    i <- i+1

    # shorten the step size
    if ((i %% 3) == 0) alpha <- alpha / 2
  }
  beta
}
```

Problem 2

$$\prod_{i=1}^{n} p_i(\beta)^{y_i}(1-p_i(\beta))^{(1-y_i)}$$

Applying negative likelihood turn products into sums

$$-\log\prod_{i=1}^{n} p_i(\beta)^{y_i}(1-p_i(\beta))^{(1-y_i)}$$

Turning products into sums

$$\begin{aligned}
L(\beta) &= -\sum_{i=1}^{n} y_i \log(p_i)(\beta) + (1-y_i)\log(1-p_i(\beta)) \\
&= -\sum_{i=1}^{n} y_i \log(p_i(\beta)) + ((1-y_i)(-\log(1+e^f))) \\
&= -\sum_{i=1}^{n} y_i \log(\frac{e^f}{1+e^f}) + ((1-y_i)(-\log(1+e^f)))
\end{aligned}$$

$$= -\sum_{i=1}^{n} y_i \log(e^f) - y_i \log(1+e^f) - \log(1+e^f) + y_i \log(1+e^f)$$

$$= -\sum_{i=1}^{n} y_i \log(e^f) - \log(1+e^f)$$

$$f_i(\beta) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

$$\frac{dL}{dB} = (-\sum_{i=1}^{n} y_i \frac{1}{e^f} * e^f - \frac{1}{1+e^f} * e^f)\vec{x_i}$$

$$\frac{dL}{dB} = -\sum_{i=1}^{n} (y_i \vec{x_i} - p_i(\beta)\vec{x_i}$$

$$\frac{dL}{dB} = -\sum_{i=1}^{n} (y_i - p_i(\beta))\vec{x_i}$$

Problem 3 The gradient descent logistic function is similar to the gradient descent, but with a different formula in the for loop. In the for loop we calculated the p-value and F values. We had to take an account when the P-value got too large so that would mean that the p-value would be one.

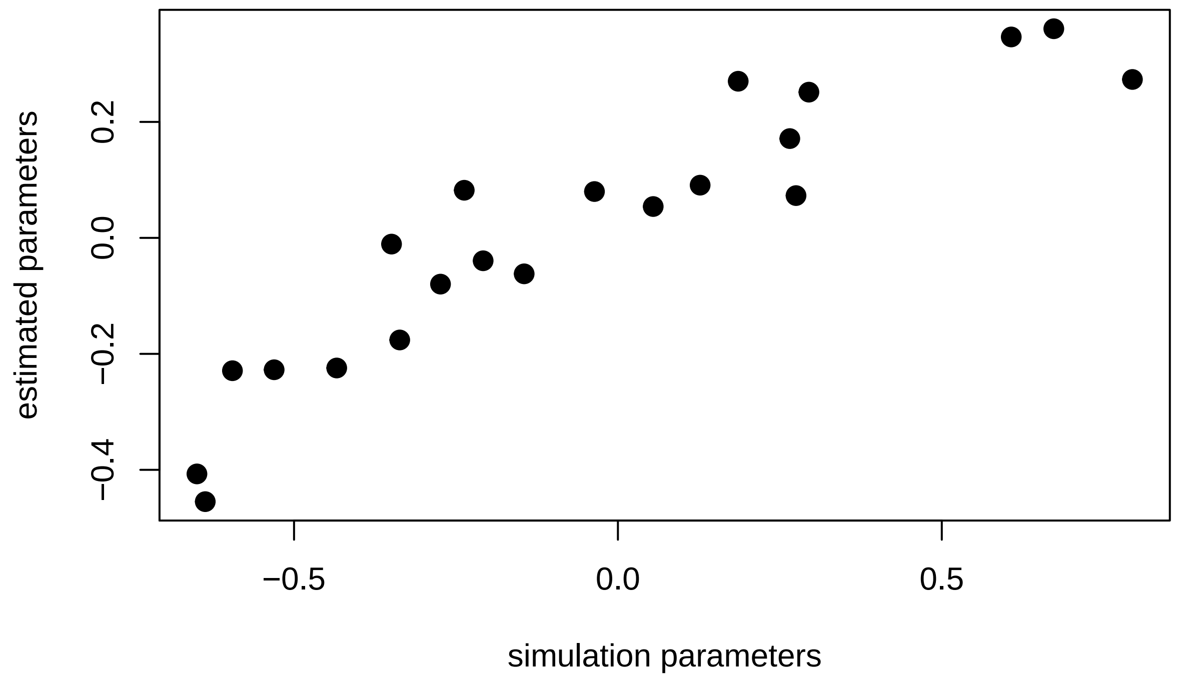
```
gapminder_obs <- gapminder
gapminder_obs$lifeExp <- NULL

gap_matrix <- data.matrix(gapminder_obs, rownames.force = NA)

gap_matrix <- cbind(1,gap_matrix)


# Implementation of gradient descent for least squares regression
# for a single predictor (x)
#
# There is some code here that is only used to generate illustrative plots and would not be part of rea
gradient_descent_log <- function(x, y, tol=1e-6, maxit=50) {
  # initialize estimate
  beta <- rep(0, ncol(x)+1)
  old_beta <- Inf; i <- 0; beta_keep <- NA

  # starting step size
  alpha <- 1e-3
  difference <- Inf

  # check for convergence
  # (in practice, we do include a limit on the number of iterations)
  while ((difference > tol) && (i < maxit) && !is.na(difference)) {

    # store the last estimate to check convergence
    old_beta <- beta

    beta_col <- cbind(1,x)

    # update estimate
    f <- beta_col %*% beta

    updates = rep(0, ncol(beta_col))
    for(k in seq(1,nrow(beta_col))) {

      exp_fk <- exp(f[k])


      ifelse((exp_fk != Inf), new_p <- (exp_fk/(1+exp_fk)), p <- 1)

      updates <- updates + (y[k]- new_p)*beta_col[k,]
    }

    beta <- beta + alpha * updates

    # compute difference after taking step
    # to check convergence
    sum_beta <- sum(beta)
    sum_oldbeta <- sum(old_beta)
    difference <- (sum_beta - sum_oldbeta)^2 / (sum_oldbeta)^2

    i <- i+1

    # shorten the step size
    if ((i %% 3) == 0) alpha <- alpha / 2
  }
  beta
}
```

Problem 4 In the code below we are simulating data from the linear regression and logistic regression models to check that our implementations recover the simulation parameters correctly. So we called the simulate regression function and to create date for simulate logistic regression function to simulate data for each function. Then using the simulated data we called the function from question 1 and 3 with the simulated data for the parameters. Finally we added the betas that were returned from the two functions and added them to our linear regression and logisitic regression and plotted the result.

```
# simulate data for linear regression
#
# parameters:
#   - npredictors: number of numeric predictors (variables)
#   - nobservations: number of observations (examples)
#   - sd: standard deviation used in random generation of outcome variable
#
# result: list with following components
#   - y: outcome variable (vector of length nobservations)
#   - x: data matrix (matrix of nobservations rows and npredictors columns)
#   - beta: linear model parameters used to generate data (vector of length
#     npredictors + 1)
simulate_regression <- function(npredictors=20,
                                nobservations = 100,
                                sd=1.5) {
  # generate beta parameters
  beta <- rnorm(npredictors+1, mean=0, sd=10/npredictors)

  # generate data matrix
  x <- matrix(rnorm(nobservations * npredictors),
        nr=nobservations,
            nc=npredictors)

  # generate outcome
  x1 <- cbind(1, x)
  y <- x1 %*% beta + rnorm(nobservations, mean=0, sd=sd)

  # return simulated data
  list(y=y,
       x=x,
       beta=beta)
}

# simulate data for logistic regression
#
# parameters:
#   - npredictors: number of numeric predictors (variables)
#   - nobservations: number of observations (examples)
#
# result: list with following components
#   - g: outcome variable (vector of length nobservations, values are 0 or 1)
#   - x: data matrix (matrix of nobservations rows and npredictors columns)
#   - beta: linear model parameters used to generate data (vector of length npredictors + 1)
simulate_logistic_regression <- function(npredictors = 20,
                                         nobservations = 100) {
  # generate parameters
  beta <- rnorm(npredictors+1, mean=0, sd=10/npredictors)

  x <- matrix(rnorm(nobservations * npredictors),
              nr=nobservations,
              nc=npredictors)

  x1 <- cbind(1, x)

  # generate outcome, i.e., do coin flips
  p <- plogis(x1 %*% beta)
  g <- rbinom(nobservations, size=1, prob=p)

  # return simulated data
  list(g=g,
       x=x,
       beta=beta)
}

#Simulating Regression

test_regress1 <- simulate_regression(20,100,1.5)
sim_beta <- gradient_descent(test_regress1$x, test_regress1$y, tol=1e-06, maxit=50)

sim_beta - test_regress1$beta
```

```
##  [1] -0.036043842 -0.000444348  0.168568769  0.209750199  0.116130383
##  [6]  0.194133893 -0.094103879  0.182251754 -0.202005874  0.366027607
## [11] -0.312312642  0.082707653 -0.521159111  0.303325513 -0.260757906
## [16]  0.084376473 -0.043563928  0.160793480  0.319363535  0.242905690
## [21]  0.338778677
```

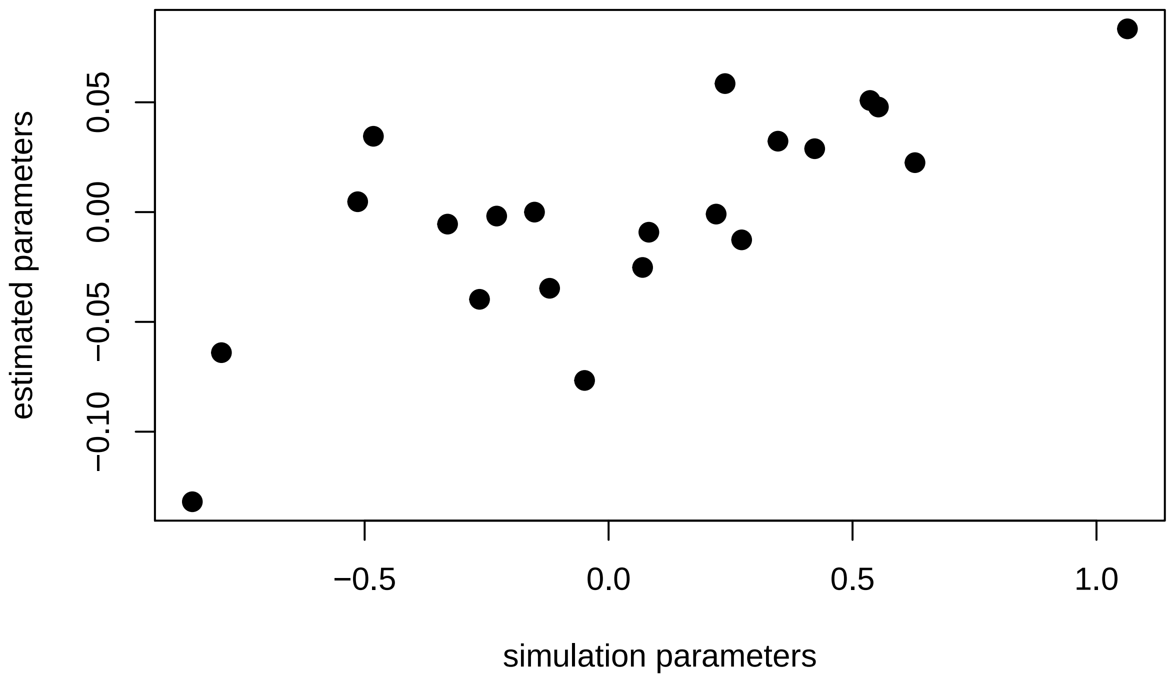
```
plot(test_regress1$beta, sim_beta, xlab="simulation parameters", ylab="estimated parameters", pch=19, c
```



```
#Simulating Logistic
test_regresslg1 <- simulate_logistic_regression(20,100)
lgbeta_test <- gradient_descent_log(test_regresslg1$x, test_regresslg1$g, tol=1e-06, maxit=50)



#How far off our results are from Hectors
lgbeta_test - test_regresslg1$beta
```

```
##  [1]  0.51898815 -0.31489050 -0.18020756 -0.22138395 -0.60549471
##  [6] -0.50519444  0.22754185 -0.48501448 -0.39358938  0.22477797
## [11]  0.08618635 -0.98004805  0.72953391 -0.09169415  0.15184370
## [16] -0.02743366  0.72129108 -0.09492058  0.51656832  0.32457628
## [21] -0.28539510
```

```
plot(test_regresslg1$beta, lgbeta_test, xlab="simulation parameters", ylab="estimated parameters", pch=
```



```
# **Part B**
# We are going to use Linear Discriminate Analysis and Classification Trees.
```

Try it Now a

The dataset we are using is from SMarket that provides 5 different lags that result from different patterns in the stock market, a year, volume which is the number shares, that day's stock as well and the direction of the stock is going. The outcome we are predicting isthe direction and by doing this we are using every variable except the Year.

c

The other two algorithms we are using are the classification trees and random forests (classification). The hyper-parameter that we are using all the lags and volume.

e

We observed significant differences between the three methods since the estimates are bigger than .05

Explanation For the Try it Now we ended up choosing to use classification trees and random forests (classification) to compare with our logistic regression model. From looking at the code you posted in lecture, we made some adjustments to make ours work. We created a dataset that has 10 folds. We then got all indicies of the observations and all the indicies for the parameters we are going to use from the dataset and inserted in two matrices. Then using the matrices we used our gradient descefnt logistic function to get the betas. Then we multiplied the matrix with the parameters we wanted to use with beta we got from our function and changed all values from that Up or Down on whether or not f was greater than or equal to zero. So with the prediction set we used the tree classification method and radom forrest to compare the results with actual results Then we did a test to compare how each model did compared with ours.

```r
y1 <- ifelse(Smarket$Direction == "Down", 0, 1)
x1 <- as.matrix(Smarket[,1:7])
beta <- gradient_descent_log(x1, y1, tol=1e-6, maxit=50)


new_smarket <- Smarket
new_smarket$y1 <- y1


beta <- as.data.frame(beta)
beta <- beta[-1,]
f <- x1 %*% beta
prediction <- ifelse(f < 0, "Down", "Up")
table(Smarket$Direction, prediction)
```

```
##       prediction
##         Up
##   Down 602
##   Up   648
```

```r
hParams <- y1~Lag1+Lag2+Lag3+Lag4+Lag5+Volume
tree <- tree(hParams, data=new_smarket)


fold_indices <- cvFolds(n=nrow(new_smarket), K=10)
error <- sapply(1:10, function(fold_index) {

  # LOGISTIC -------------------------------------------------------------------

  test_indices <- which(fold_indices$which == fold_index)
  train_set <- new_smarket[-test_indices,]
  test_set <- new_smarket[test_indices,]

  trainX <- as.matrix(train_set[,2:8])
  trainY <- train_set[,10]
  testX <- as.matrix(test_set[,2:8])

  betas <- gradient_descent_log(trainX, trainY, tol=1e-6, maxit=50)
  betas <- as.data.frame(betas)
  betas <- betas[-1,]

  f <- testX %*% betas
  prediction <- ifelse(f >= 0, "Up", "Down")
  logis_error <- mean(prediction != test_set$Direction)

  # RANDOM FOREST --------------------------------------------------------------

  train_indices <- sample(nrow(Smarket), nrow(Smarket)/2)
  train_set <- Smarket[train_indices,]
  test_set <- Smarket[-train_indices,]

  hParams2 <- Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume
  auto_rf <- randomForest(hParams2, importance=TRUE, mtry=3, data=train_set)
  rmse <- sqrt( mean( (predict(auto_rf, newdata=test_set) != test_set$Direction)^2 ))



  # CLASSIFICATION TREE --------------------------------------------------------------

  treefit <- tree(Direction~., data=train_set)
  prunedTree <- prune.tree(treefit, best=2)
  treePrediction <- predict(prunedTree, newdata=test_set)
  updated_pred <- vector()

  # Makes more simple to read predictions
  for(i in seq(1, length(treePrediction))) {
    ifelse(treePrediction[i] == 1, updated_pred[i] <- "Up", updated_pred[i] <- "Down"
  )}

  tree_err <- mean(updated_pred != test_set$Direction)
  c(logis_error, rmse, tree_err)

  })


rownames(error) <- c("logis", "randomforest", "tree")
error <- as.data.frame(t(error))


error <- error %>%
  mutate(folds=1:n()) %>%
  gather(method, error, -folds)


# ttest
lm(error~method, data=error) %>%
  tidy() %>%
  knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.0256000	0.0023993	10.66964	0
methodrandomforest	0.6831536	0.0033932	201.33222	0
methodtree	0.4744000	0.0033932	139.81045	0