Nishant Arora

CMSC 320

Project 3

Part 1: Regression Analysis of Gapminder Data

Goals:

- To practice and experiment with linear regression using data from gapminder.org (part 1)
- To explore how life expectancy has changed over 50 years across the world. (part 1)

Libraries and Loading the Data

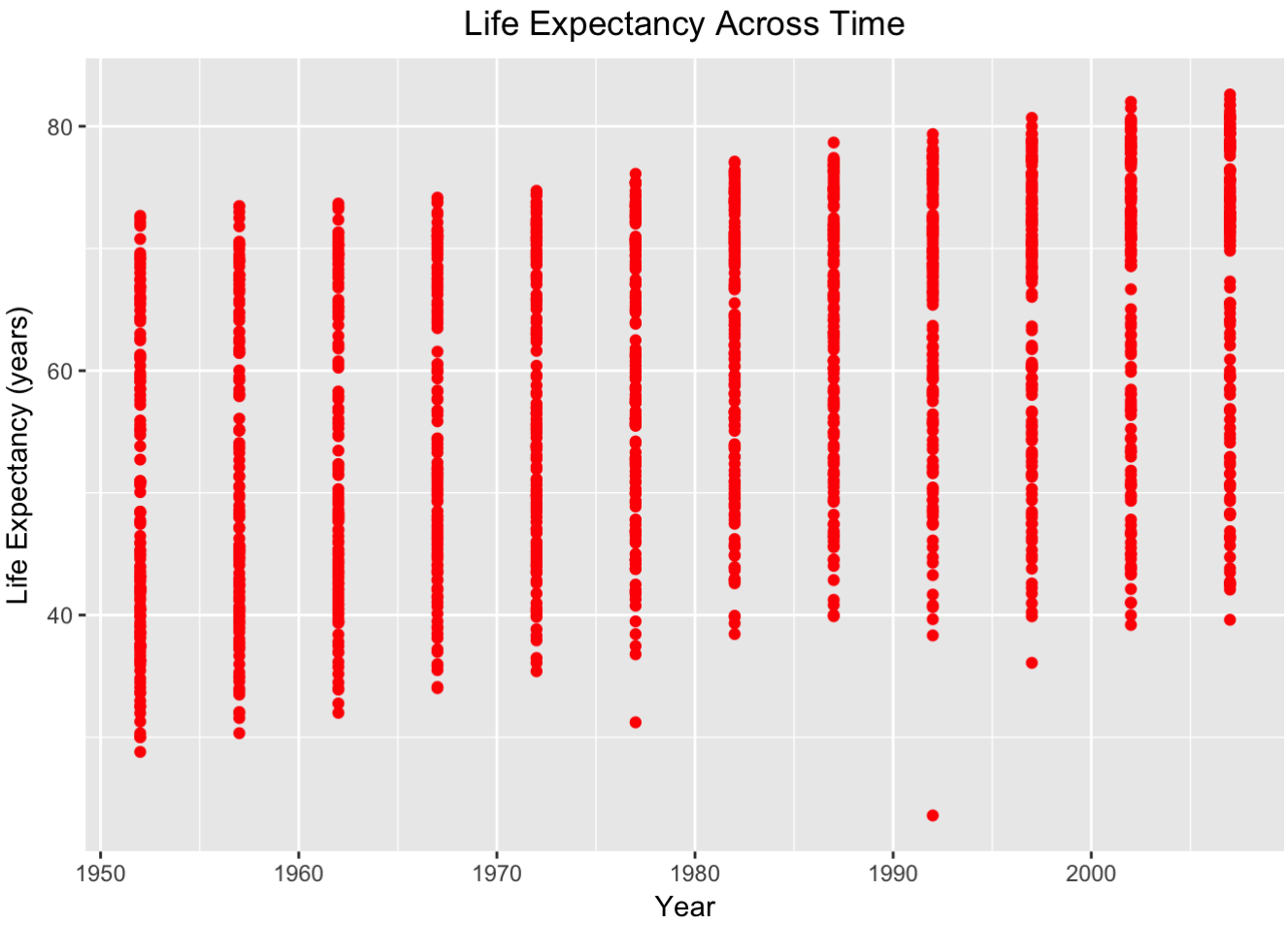
```
library(dplyr, quietly=TRUE, warn.conflicts=FALSE)
library(ggplot2, quietly=TRUE, warn.conflicts=FALSE)
library(broom, quietly=TRUE, warn.conflicts=FALSE)

# The following commands load the dataset:
library(gapminder)
data(gapminder)

head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333       779
## 2 Afghanistan Asia       1957    30.3  9240934       821
## 3 Afghanistan Asia       1962    32.0 10267083       853
## 4 Afghanistan Asia       1967    34.0 11537966       836
## 5 Afghanistan Asia       1972    36.1 13079460       740
## 6 Afghanistan Asia       1977    38.4 14880372       786
```

Exercises and Questions:

Exercise 1: Make a scatter plot of life expectancy across time.

```
gapminder %>%
  ggplot(aes(x=year, y=lifeExp)) +
    geom_point(color="red") +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Life Expectancy Across Time", x="Year", y="Life Expectancy (years)")
```

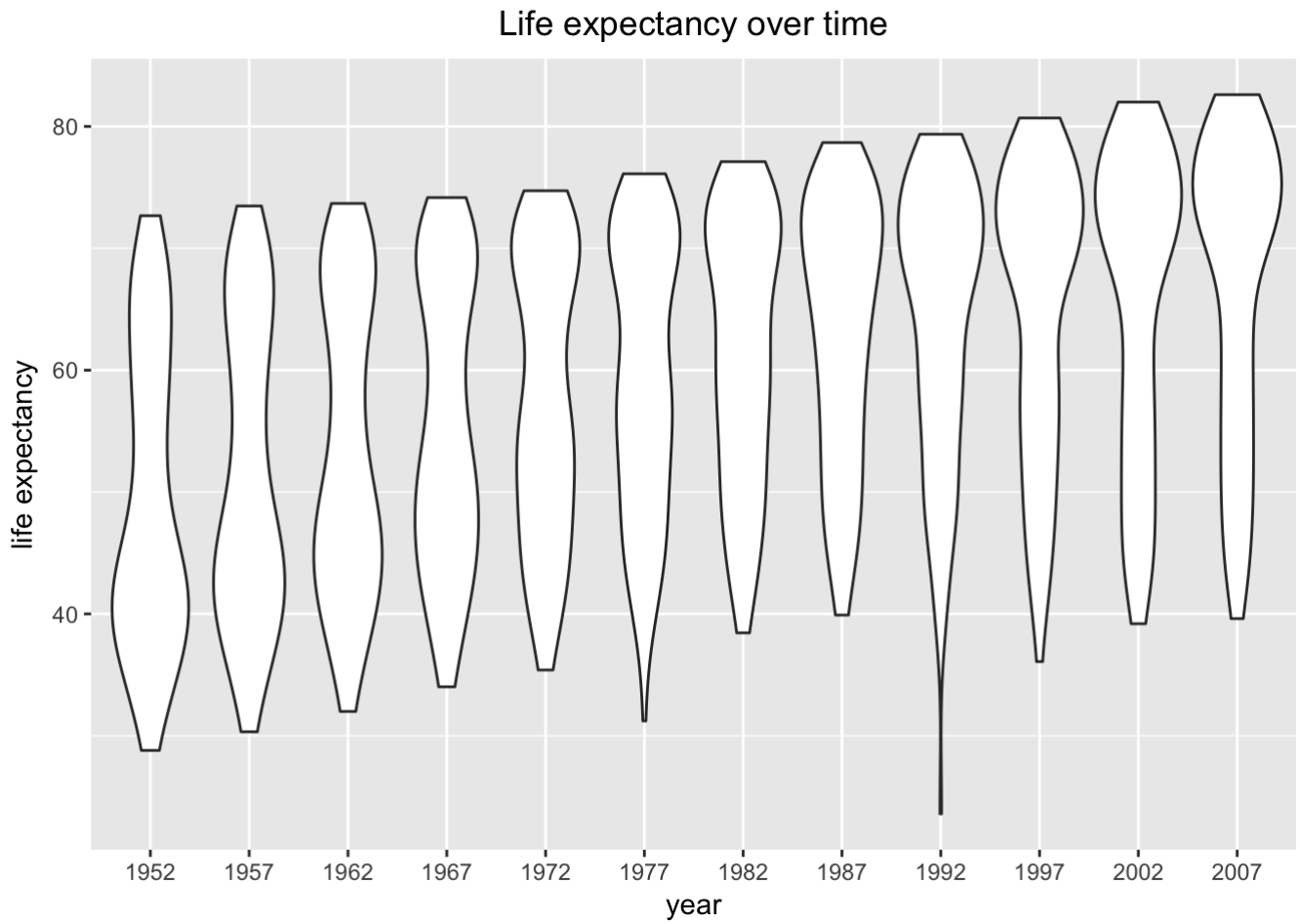


Question 1: Is there a general trend (e.g., increasing or decreasing) for life expectancy across time? Is this trend linear? (answering this qualitatively from the plot, you will do a statistical analysis of this question shortly)

Without perfroming a statistical analysis, there appears to be an increasing linear trend for life expectancy over time.

A slightly different way of making the same plot is looking at the distribution of life expectancy across countries as it changes over time:

```
gapminder %>%
  ggplot(aes(x=factor(year), y=lifeExp)) +
    geom_violin() +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Life expectancy over time", x = "year", y = "life expectancy")
```



This type of plot is called a violin plot, and it displays the distribution of the variable in the y-axis for each value of the variable in the x-axis.

Question 2: How would you describe the distribution of life expectancy across countries for individual years? Is it skewed, or not? Unimodal or not? Symmetric around its center?

The distribution has more data on the low end of each “violin” in the beginning but it begins to move to the upper end over time (bottom heavy to top heavy). This shows that life expectancy across countries is increasing over time.

The data is most bottom heavy in 1952. 1962-1972 seems to be bimodal as it is hard to tell where the data is most concentrated. The other years appear to be unimodal. The data is clearly top heavy from 1997 to 2007. There is obvious skew in 1992 andd possibly 1977 as seen by the sharp tails.

Based on this plot, consider the following questions.

Question 3: Suppose I fit a linear regression model of life expectancy vs. year (treating it as a continuous variable), and test for a relationship between year and life expectancy, will you reject the null hypothesis of no relationship? (do this without fitting the model yet. I am testing your intuition.)

It seems as if there is a strong relationship between year and life expectancy. I would reject the null hypothesis since I do not believe this large sample could have been obtained if the null hypothesis was true.

Question 4: What would a violin plot of residuals from the linear model in Question 3 vs. year look like? (Again, don’t do the analysis yet, answer this intuitively)

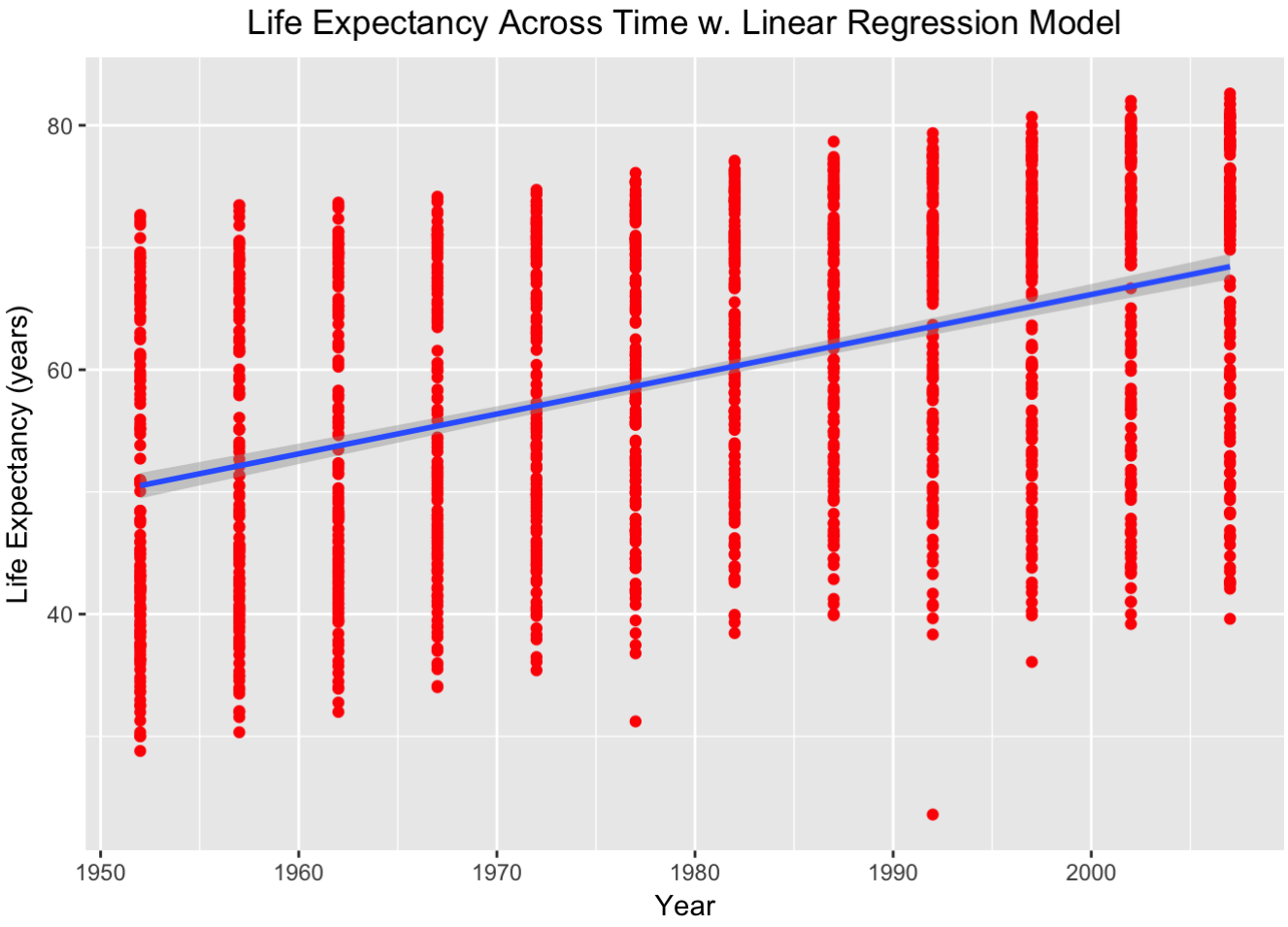
I expect that the data on the residual plot will be linear because I believe there is a linear relationship between life expectancy and year.

Question 5: According to the assumptions of the linear regression model, what should that violin plot look like?

The residual plot should be centered around 0 since I am expecting a linear model is being fitted to data that has a linear relationship.

Exercise 2: Fit a linear regression model using the lm function for life expectancy vs. year (as a continuous variable). Use the broom::tidy to look at the resulting model.

```
gapminder %>%
  ggplot(aes(x=year, y=lifeExp)) +
    geom_point(color="red") +
    geom_smooth(method=lm) +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Life Expectancy Across Time w. Linear Regression Model", x="Year", y="Li
fe Expectancy (years)")
```




```
fit <- lm(year~lifeExp, data=gapminder)
stats <- tidy(fit)

stats
```

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

Question 6: On average, by how much does life expectancy increase every year around the world?

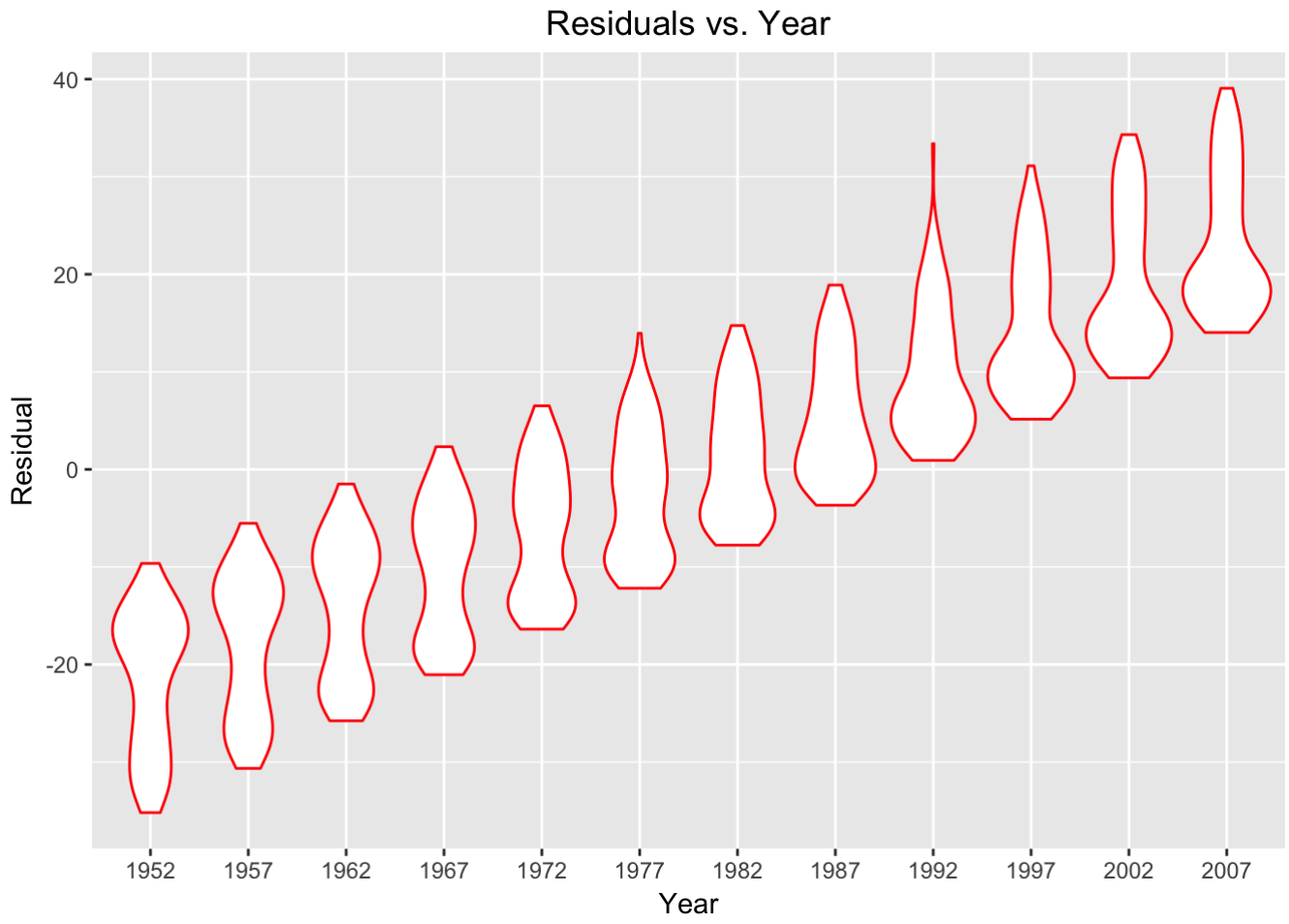
On average, life expectancy increases by 0.5822489 years every year around the world.

Question 7: Do you reject the null hypothesis of no relationship between year and life expectancy? Why?

We reject the null hypothesis of no relationship between year and life expectancy. The p-value is extremely small to the point of being negligible (7.546795e-80). This is the probability that null hypothesis is true (no relationship between year and life expectancy). Since this value is so small, we reject the null hypothesis.

Exercise 3: Make a violin plot of residuals vs. year for the linear model from Exercise 2 (use the broom::augment function).

```
augmented <- augment(fit)

augmented %>%
  ggplot(aes(x=factor(year), y=.resid)) +
    geom_violin(color="red") +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Residuals vs. Year", x="Year", y="Residual")
```

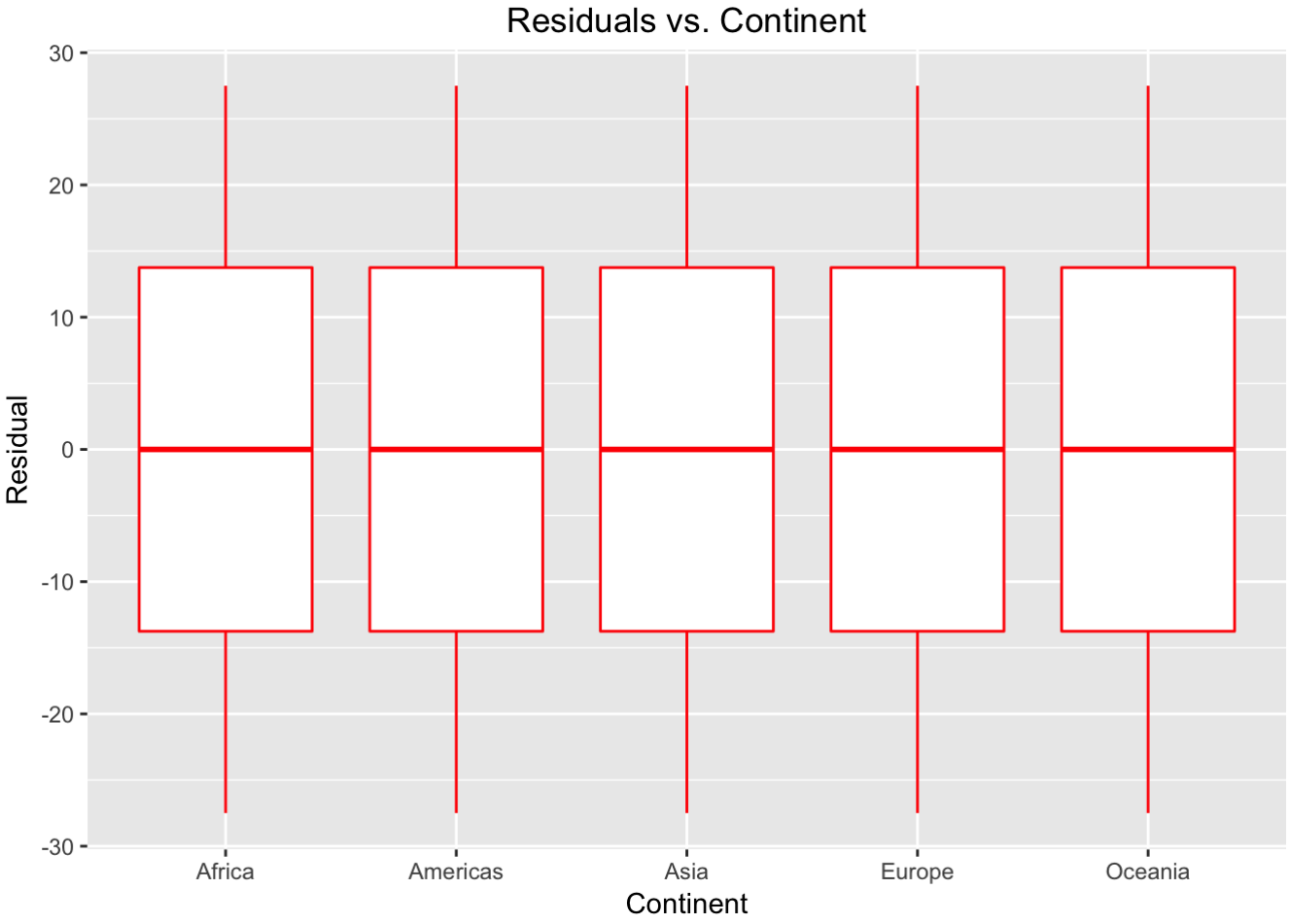


Question 8: Does the plot of Exercise 3 match your expectations (as you answered Question 4)?

Yes. The results are centered around 0 which is what I expected. This seems to be a well fitted model.

Exercise 4: Make a boxplot (or violin plot) of model residuals vs. continent.

```
augment(lm(year~continent, data=gapminder)) %>%
  ggplot(aes(x=factor(continent), y=.resid)) +
    geom_boxplot(color="red") +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Residuals vs. Continent", x="Continent", y="Residual")
```

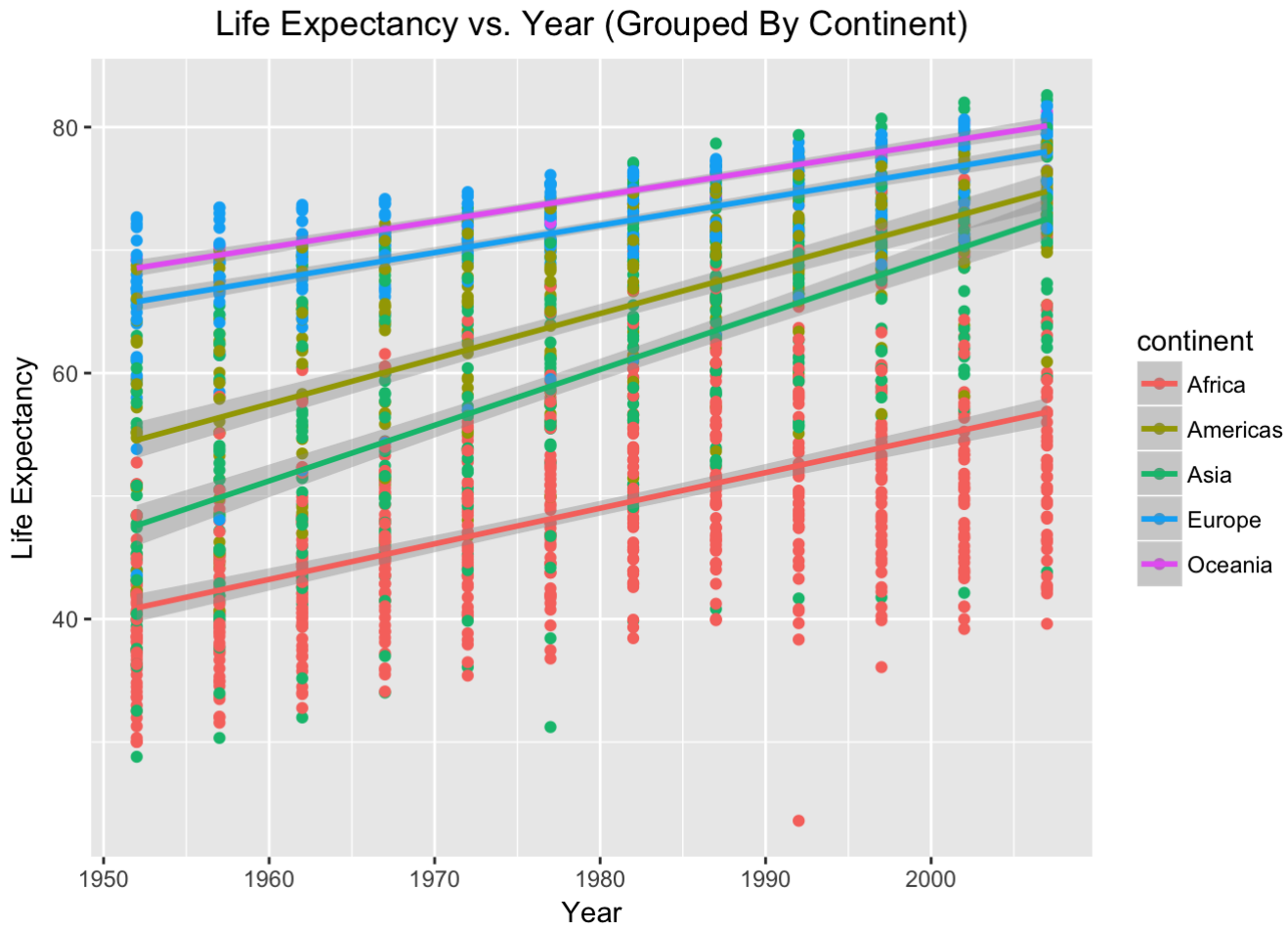


Question 9: Is there a dependence between model residual and continent? If so, what would that suggest when performing a regression analysis of life expectancy across time?

There is not a dependence between model residual and continent. The residuals are independent and have the same distribution. There is no variation in the residuals at all when plotted as a function of continent. Since they are the same for every continent, the residuals are independent.

Exercise 5: Use geom_smooth(method=lm) in ggplot as part of a scatter plot of life expectancy vs. year, grouped by continent (e.g., using the color aesthetic mapping).

```
gapminder %>%
  ggplot(aes(x=year, y=lifeExp, color=continent)) +
    geom_point() +
    geom_smooth(method=lm) +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Life Expectancy vs. Year (Grouped By Continent)", x="Year", y="Life Expe
ctancy")
```



Question 10: Based on this plot, should your regression model include an interaction term for continent and year? Why?

Yes, the regression model should include an interaction term for continent and year. The coefficients for the linear models for each continent are not the same (they all have varying slopes). For instanct, the regression line for Asia is clearly steeper than the others. By including an interaction term for continent and year, we can account for the differences among continents and make our model more accurate.

Exercise 6: Fit a linear regression model for life expectancy including a term for an interaction between continent and year. Use the broom::tidy function to show the resulting model.

```
fit_term <- lm(lifeExp~year*continent, data=gapminder)
stats_term <- tidy(fit_term)

stats_term
```

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

Question 11: Are all parameters in the model significantly different from zero? If not, which are not significantly different from zero?

All of the parameters in the model are extremely close to 0.

Question 12: On average, by how much does life expectancy increase each year for each continent? (Write code to answer this question by extracting relevant estimates from model fit)

```
# extract relevant estimates
estimates <- summary(fit_term)$coefficients

year <- estimates["year", "Estimate"]
americas <- estimates["year:continentAmericas", "Estimate"]
asia <- estimates["year:continentAsia", "Estimate"]
europe <- estimates["year:continentEurope", "Estimate"]
oceania <- estimates["year:continentOceania", "Estimate"]

# get the values
americas_increase <- year + americas
africa_increase <- year
asia_increase <- year + asia
europe_increase <- year + europe
oceania_increase <- year + oceania

# make vectors (cols in table)
increases <- c(americas_increase, africa_increase, asia_increase, europe_increase, ocean
ia_increase)
continents <- c("Americas", "Africa", "Asia", "Europe", "Oceania")

# make table
life_exp_increase_continent <- data.frame(Continent = continents, LifeExp_Increase = inc
reases)

life_exp_increase_continent
```

```
##   Continent LifeExp_Increase
## 1  Americas        0.3676509
## 2    Africa        0.2895293
## 3      Asia        0.4531224
## 4    Europe        0.2219321
## 5   Oceania        0.2102724
```

Exercise 7: Use the anova function to perform an F-test that compares how well two models fit your data: (a) the linear regression models from Exercise 2 (only including year as a covariate) and (b) Exercise 6 (including interaction between year and continent).

```
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: year
##             Df Sum Sq Mean Sq F value    Pr(>F)
## 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
```

```
anova(fit_term)
```

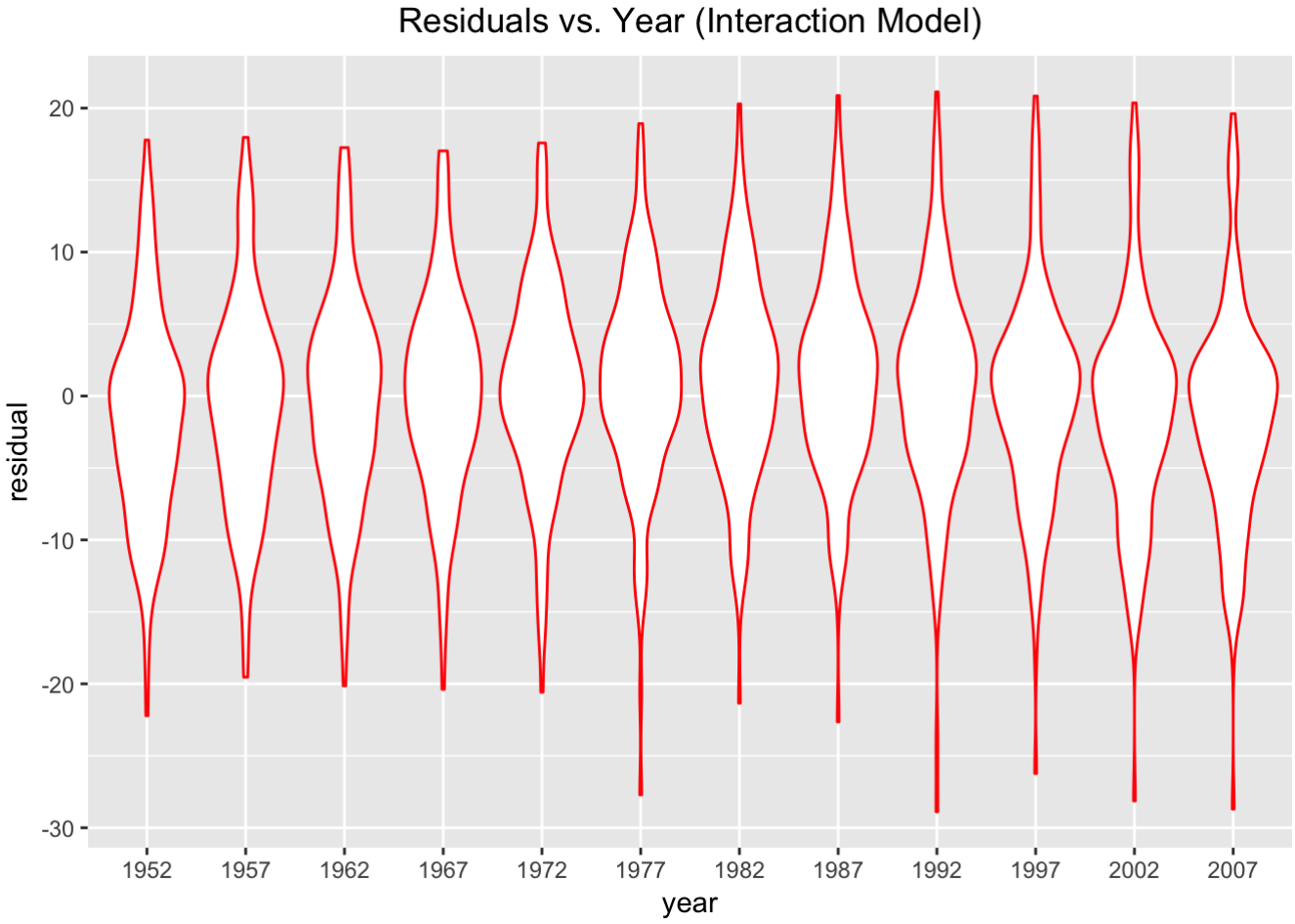
```
## Analysis of Variance Table
##
## Response: lifeExp
##                  Df Sum Sq Mean Sq  F value    Pr(>F)
## year              1  53919   53919 1046.028 < 2.2e-16 ***
## continent         4 139343   34836  675.812 < 2.2e-16 ***
## year:continent    4   3566     892   17.296 6.463e-14 ***
## Residuals      1694  87320      52
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Question 13: Is the interaction model significantly better than the year-only model? Why?

The RSS value and F-Test value are both greater than that of the year only model. In context of the data, this makes sense since we know life expectancy is increasing over time across continents at different rates. By creating an interaction between contienent and year we are able to more accurately model life expectancy. Therfore the interaction model is better than the year only model.

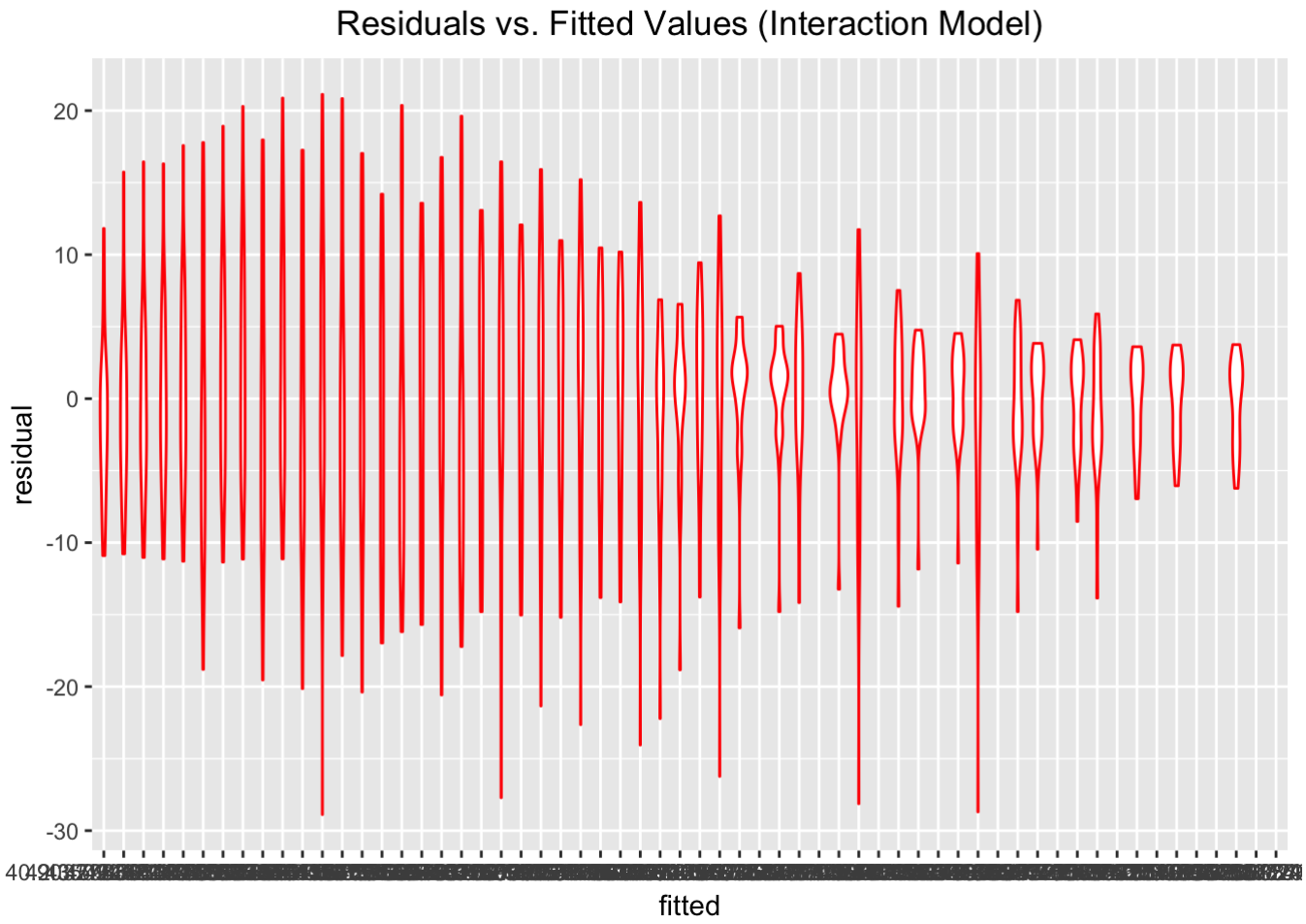
Exercise 8: Make a residuals vs. year violin plot for the interaction model. Comment on how well it matches assumptions of the linear regression model. Do the same for a residuals vs. fitted values model. (You should use the broom::augment function).

```
augment(fit_term) %>%
  ggplot(aes(x=factor(year), y=.resid)) +
    geom_violin(color="red") +
    labs(title="Residuals vs. Year (Interaction Model)", x="year", y="residual") +
    theme(plot.title = element_text(hjust = 0.5))
```



The residuals are centered at 0. They match the assumptions of the linear regression model.

```
augment(fit_term) %>%
  ggplot(aes(x=factor(.fitted), y=.resid)) +
    geom_violin(color="red") +
    theme(plot.title = element_text(hjust = 0.5)) +
    labs(title="Residuals vs. Fitted Values (Interaction Model)", x="fitted", y="residua
l")
```



The residuals are once again centered at 0. They match the assumptions of the linear regression model.

Part 2: Classification from Zillow Data

Goals:

- To use Mortgage Affordability data from Zillow to experiment with classification algorithms
- To predict if mortgage affordability will increase or decrease a year from now

Libraries and Loading the (tidy) Data

```
library(tidyverse, quietly=TRUE, warn.conflicts=FALSE)
library(lubridate, quietly=TRUE, warn.conflicts=FALSE)
library(dplyr, quietly=TRUE, warn.conflicts=FALSE)
library(tidyr, quietly=TRUE, warn.conflicts=FALSE)
library(randomForest, quietly=TRUE, warn.conflicts=FALSE)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
library(caret, quietly=TRUE, warn.conflicts=FALSE)
```

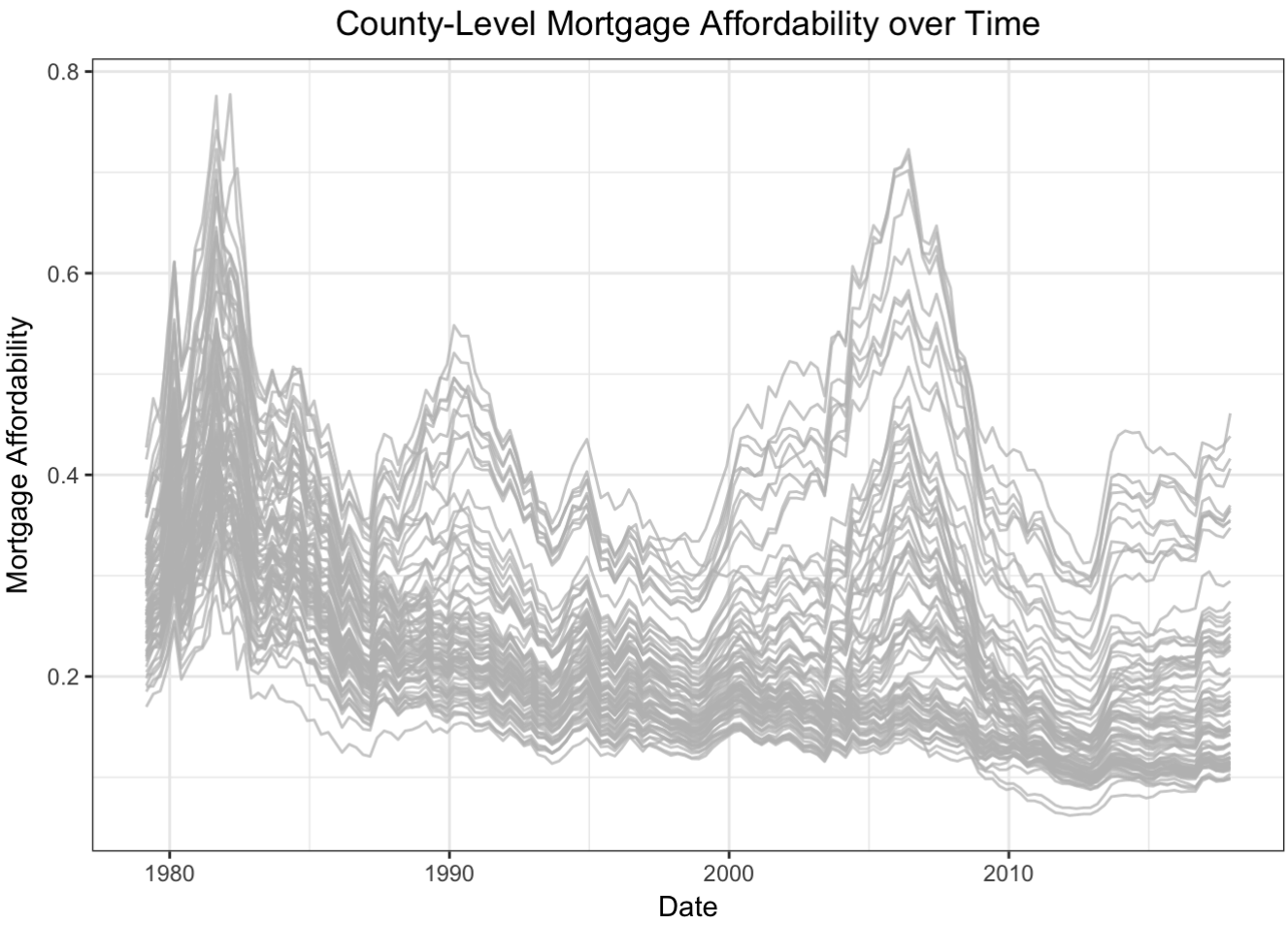
```
## Warning: package 'caret' was built under R version 3.4.4
```

```
library(ROCR, quietly=TRUE, warn.conflicts=FALSE)
theme_set(theme_bw())

# The following commands load the dataset and tidy the data:
csv_file <- "Affordability_Wide_2017Q4_Public.csv"
tidy_afford <- read_csv(csv_file) %>%
  filter(Index == "Mortgage Affordability") %>%
  drop_na() %>%
  filter(RegionID != 0, RegionName != "United States") %>%
  select(RegionID, RegionName, matches("^[1|2]")) %>%
  gather(time, affordability, matches("^[1|2]")) %>%
  type_convert(col_types=cols(time=col_date(format="%Y-%m")))

tidy_afford
```

```
## # A tibble: 12,480 x 4
##    RegionID RegionName                          time       affordability
##       <int> <chr>                               <date>             <dbl>
##  1   394913 New York, NY                        1979-03-01         0.262
##  2   753899 Los Angeles-Long Beach-Anaheim, CA 1979-03-01         0.358
##  3   394463 Chicago, IL                         1979-03-01         0.262
##  4   394514 Dallas-Fort Worth, TX               1979-03-01         0.301
##  5   394974 Philadelphia, PA                    1979-03-01         0.204
##  6   394692 Houston, TX                         1979-03-01         0.243
##  7   395209 Washington, DC                      1979-03-01         0.254
##  8   394856 Miami-Fort Lauderdale, FL           1979-03-01         0.268
##  9   394347 Atlanta, GA                         1979-03-01         0.248
## 10   394404 Boston, MA                          1979-03-01         0.222
## # ... with 12,470 more rows
```

This is what the data looks like:

```
tidy_afford %>%
  ggplot(aes(x=time,y=affordability,group=factor(RegionID))) +
  geom_line(color="GRAY", alpha=3/4, size=1/2) +
  labs(title="County-Level Mortgage Affordability over Time", x="Date", y="Mortgage Affo
rdability") +
  theme(plot.title = element_text(hjust = 0.5))
```



The Prediction Task:

Can we predict if mortgage affordability will increase or decrease a year from now?

Specifically, we will do this for the last observation in the dataset (quarter 4 (Q4) of 2017). To create the outcome we will predict we will compare affordability for Q4 of 2017 and to Q4 of 2016 and label it as up or down depending on the sign of the this difference. Let's create the outcome we want to predict:

```
outcome_df <- tidy_afford %>%
  mutate(yq = quarter(time, with_year=TRUE)) %>%
  filter(yq %in% c("2016.4", "2017.4")) %>%
  select(RegionID, RegionName, yq, affordability) %>%
  spread(yq, affordability) %>%
  mutate(diff = `2017.4` - `2016.4`) %>%
  mutate(Direction = ifelse(diff>0, "up", "down")) %>%
  select(RegionID, RegionName, Direction)
outcome_df
```

```
## # A tibble: 80 x 3
##    RegionID RegionName      Direction
##       <int> <chr>           <chr>
##  1   394304 Akron, OH       down
##  2   394312 Albuquerque, NM down
##  3   394318 Allentown, PA   down
##  4   394347 Atlanta, GA     up
##  5   394355 Austin, TX      up
##  6   394357 Bakersfield, CA down
##  7   394358 Baltimore, MD   down
##  8   394367 Baton Rouge, LA up
##  9   394378 Bellingham, WA  up
## 10   394388 Birmingham, AL  down
## # ... with 70 more rows
```

Now, we have a dataframe with outcomes (labels) for each county in the dataset.

The goal is then given predictors X_i for county i, build a classifier for outcome $G_i \in \{\texttt{up},\texttt{down}\}$.

For your classifiers we use data up to 2016:

```
predictor_df <- tidy_afford %>%
  filter(year(time) <= 2016)
```

The Project:

Conduct an experiment to address a (one, single) technical question (from the provided list) about our ability to make this prediction. Each of them asks two compare two specific choices in the classification workflow (e.g., two classification algorithms, two feature representations, etc.). Implement each of the two choices and use 10-fold cross validation (across RegionID's) to compare their relative performance. Also create an AUROC curve to compare them.

The Question:

I decided to create my own question - **Is a random forest better than logistic regression?**

A Random Forest Classifier

Let's build a Random Forest classifer using quarterly differences after data standardization for years 2014-2016. First, filter to the years of interest and standardize affordability for each region

```
standardized_df <- predictor_df %>%
  filter(year(time) %in% 1976:2016) %>%
  group_by(RegionID) %>%
  mutate(mean_aff = mean(affordability)) %>%
  mutate(sd_aff = sd(affordability)) %>%
  mutate(z_aff = (affordability - mean_aff) / sd_aff) %>%
  ungroup()
```

To train our model we need a table with one row per region, and attributes corresponding to differences in quarterly affordability. We will do this in stages, first we turn the tidy dataset into a wide dataset using tidyr::spread then create a dataframe containing the differences we use as features.

```
wide_df <- standardized_df %>%
  select(RegionID, time, z_aff) %>%
  spread(time, z_aff)
```

Now, we turn this into quarterly differences

```
matrix_1 <- wide_df %>%
  select(-RegionID) %>%
  as.matrix() %>%
  .[,-1]

matrix_2 <- wide_df %>%
  select(-RegionID) %>%
  as.matrix() %>%
  .[,-ncol(.)]

diff_df <- (matrix_1 - matrix_2) %>%
  magrittr::set_colnames(NULL) %>%
  as_data_frame() %>%
  mutate(RegionID = wide_df$RegionID)
```

Finally, add the outcome we want to predict from the data frame we created previously.

```
final_df <- diff_df %>%
  inner_join(outcome_df %>% select(RegionID, Direction), by="RegionID") %>%
  mutate(Direction=factor(Direction, levels=c("down", "up")))
final_df
```

```
## # A tibble: 80 x 153
##        V1      V2    V3    V4     V5      V6    V7       V8      V9   V10    V11
V12     V13    V14    V15    V16
##     <dbl>   <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl>    <dbl>   <dbl> <dbl>  <dbl>   <d
bl>   <dbl>  <dbl>  <dbl>  <dbl>
##  1 0.261   0.116  0.436 0.771 -0.879  0.109  0.613  0.0634  0.459   0.541 -0.568 -0.2
03   0.0700 -0.718 -0.557 -0.222
##  2 0.327   0.0894 0.437 0.940 -0.920  0.217  0.662  0.0725  0.573   0.424 -0.571 -0.1
25   0.222  -0.447 -0.777 -0.251
##  3 0.0255 -0.0181 0.715 1.42  -1.94   0.466  0.703  0.448   0.00116 0.404 -0.633 -0.1
91   0.488  -0.127 -1.17  -0.510
##  4 0.217   0.0953 0.542 0.792 -0.860  0.201  0.473  0.224   0.410   0.387 -0.272  0.0
402 -0.116  -0.512 -0.545 -0.310
##  5 0.125   0.176  0.605 0.533 -0.503 -0.181  0.453  0.201   0.527   0.452 -0.581  0.1
11  -0.134  -0.458 -0.383 -0.298
##  6 0.322   0.0740 0.495 0.542 -0.735  0.263  0.242  0.176   0.312   0.623 -0.197 -0.3
80   0.0383 -0.194 -0.731 -0.181
##  7 0.361   0.0385 0.505 0.829 -0.892  0.243  0.649  0.240   0.659   0.339 -0.289 -0.1
27  -0.480  -0.407 -0.852 -0.208
##  8 0.230   0.0817 0.355 0.737 -0.706  0.0908 0.608  0.00910 0.176   0.590 -0.235  0.0
738 -0.196  -0.365 -0.427 -0.288
##  9 0.258   0.222  0.402 0.594 -0.605  0.410  0.166  0.595   0.528   0.440 -0.466  0.2
58  -0.592  -0.403 -0.613 -0.462
## 10 0.173  -0.0215 0.821 0.295 -1.09   0.635  0.459 -0.142   0.818   0.489 -0.853  0.0
436  0.0230 -0.769 -0.466 -0.281
## # ... with 70 more rows, and 137 more variables: V17 <dbl>, V18 <dbl>, V19 <dbl>, V20
<dbl>, V21 <dbl>, V22 <dbl>,
## #   V23 <dbl>, V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, V29 <dbl>, V30
<dbl>, V31 <dbl>, V32 <dbl>,
## #   V33 <dbl>, V34 <dbl>, V35 <dbl>, V36 <dbl>, V37 <dbl>, V38 <dbl>, V39 <dbl>, V40
<dbl>, V41 <dbl>, V42 <dbl>,
## #   V43 <dbl>, V44 <dbl>, V45 <dbl>, V46 <dbl>, V47 <dbl>, V48 <dbl>, V49 <dbl>, V50
<dbl>, V51 <dbl>, V52 <dbl>,
## #   V53 <dbl>, V54 <dbl>, V55 <dbl>, V56 <dbl>, V57 <dbl>, V58 <dbl>, V59 <dbl>, V60
<dbl>, V61 <dbl>, V62 <dbl>,
## #   V63 <dbl>, V64 <dbl>, V65 <dbl>, V66 <dbl>, V67 <dbl>, V68 <dbl>, V69 <dbl>, V70
<dbl>, V71 <dbl>, V72 <dbl>,
## #   V73 <dbl>, V74 <dbl>, V75 <dbl>, V76 <dbl>, V77 <dbl>, V78 <dbl>, V79 <dbl>, V80
<dbl>, V81 <dbl>, V82 <dbl>,
## #   V83 <dbl>, V84 <dbl>, V85 <dbl>, V86 <dbl>, V87 <dbl>, V88 <dbl>, V89 <dbl>, V90
<dbl>, V91 <dbl>, V92 <dbl>,
## #   V93 <dbl>, V94 <dbl>, V95 <dbl>, V96 <dbl>, V97 <dbl>, V98 <dbl>, V99 <dbl>, V100
<dbl>, V101 <dbl>, V102 <dbl>,
## #   V103 <dbl>, V104 <dbl>, V105 <dbl>, V106 <dbl>, V107 <dbl>, V108 <dbl>, V109 <dbl
>, V110 <dbl>, V111 <dbl>,
## #   V112 <dbl>, V113 <dbl>, V114 <dbl>, V115 <dbl>, V116 <dbl>, …
```

Cross-Validation:

We are looking at 10-fold cross-validation to compare a random forest with 500 trees, with a logistic regression model.

First, we create data frame to contain the results of the experiment. That is, for each observation (region) in the training set (rows) we will have attributes for predictions (as continuous values) for both random forests, the observed labels (‘up’ or ‘down’) and index of the cross-validation fold in which the observation was used as a test example. We will then use this data frame to compute error rates, do a hypothesis test for differences in error and to create an ROC curve.

We use the caret::createFolds function to create a stratified 5-fold cross-validation partition. The result is a list of length 5, containing the indices of the examples used for validation in each of the cross-validation folds.

We pipe the result to the purrr::imap function which applies to each list (and it’s index) a function that (a) splits the data into train and test sets, (b) fits the two random forests, and (c) creates a data frame with the values we need.

The result of purrr::imap is a list of data frames, so we pipe it to purrr::reduce to create a single data frame.

```
set.seed(1234)
# create the cross-validation partition
result_df <- createFolds(final_df$Direction, k=10) %>%
  # fit models and gather results
  purrr::imap(function(test_indices, fold_number) {
    # split into train and test for the fold
    train_df <- final_df %>%
      select(-RegionID) %>%
      slice(-test_indices)

    test_df <- final_df %>%
      select(-RegionID) %>%
      slice(test_indices)

    # fit the two models
    rf <- randomForest(Direction~., data=train_df, ntree=500)
    lm <- glm(Direction~., data=train_df, family = "binomial")

    # gather results
    test_df %>%
      select(observed_label = Direction) %>%
      mutate(fold=fold_number) %>%
      mutate(prob_positive_rf = predict(rf, newdata=test_df, type="prob")[,"up"]) %>%
      # add predicted labels for rf1 using a 0.5 probability cutoff
      mutate(predicted_label_rf = ifelse(prob_positive_rf > 0.5, "up", "down")) %>%
      mutate(prob_positive_lm = predict(lm, newdata=test_df, type="response")) %>%
      # add predicted labels for lm using a 0.5 probability cutoff
      mutate(predicted_label_lm = ifelse(prob_positive_lm > 0.5, "up", "down"))
}) %>%
  # combine the five result data frames into one
  purrr::reduce(bind_rows)
result_df
```

```
## # A tibble: 80 x 6
##    observed_label fold   prob_positive_rf predicted_label_rf     prob_positive_lm pre
dicted_label_lm
##    <fct>          <chr>             <dbl> <chr>                             <dbl> <ch
r>
##  1 up             Fold01            0.586 up                  1.000                 up

##  2 up             Fold01            0.836 up                  1.000                 up

##  3 up             Fold01            0.760 up                  0.926                 up

##  4 down           Fold01            0.480 down                0.436                 dow
n
##  5 down           Fold01            0.576 up                  0.000000000000000222 dow
n
##  6 up             Fold01            0.604 up                  0.000000000000000222 dow
n
##  7 down           Fold01            0.564 up                  1.000                 up

##  8 up             Fold01            0.840 up                  1.000                 up

##  9 up             Fold02            0.934 up                  1.000                 up

## 10 up             Fold02            0.574 up                  0.000000000000000222 dow
n
## # ... with 70 more rows
```

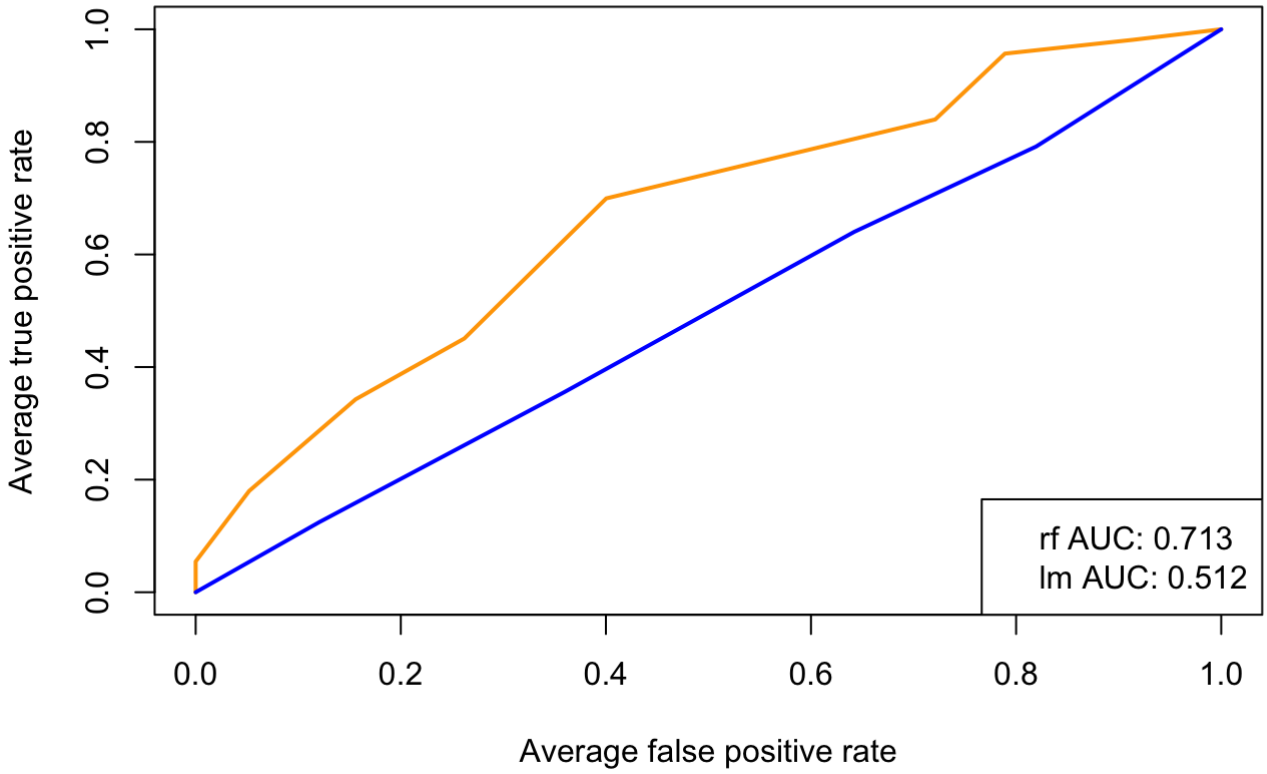
Now let's compute error rates for each model on each fold, and do a test for differences in error rate.

```
result_df %>%
  mutate(error_rf = observed_label != predicted_label_rf,
         error_lm = observed_label != predicted_label_lm) %>%
  group_by(fold) %>%
  summarize(big_rf = mean(error_rf), small_rf = mean(error_lm)) %>%
  tidyr::gather(model, error, -fold) %>%
  lm(error~model, data=.) %>%
  broom::tidy()
```

```
##            term  estimate  std.error statistic      p.value
## 1   (Intercept) 0.3730159 0.05158368  7.231277 1.001472e-06
## 2 modelsmall_rf 0.1053571 0.07295034  1.444231 1.658558e-01
```

Now, let's make an ROC curve. When making ROC curves based on a cross-validation experiment, we need to aggregate across the five folds in some way. The ROCR package provides facilities to do this in the plot function. See comments in code below to see how aggregation is done. For this to work properly the ROCR::prediction function takes lists of predictions and labels instead of vectors to create ROC curves and compute AUROC. We use the split function to create these lists.

```
# create a list of true observed labels
labels <- split(result_df$observed_label, result_df$fold)

# now create a list of predictions for the first (rf) and pass it to the ROCR::predictio
n function
predictions_rf <- split(result_df$prob_positive_rf, result_df$fold) %>% prediction(label
s)

# do the same for the second (lm)
predictions_lm <- split(result_df$prob_positive_lm, result_df$fold) %>% prediction(label
s)

# compute average AUC for the first (rf)
mean_auc_rf <- predictions_rf %>%
  performance(measure="auc") %>%
  # I know, this line is ugly, but that's how it is
  slot("y.values") %>% unlist() %>%
  mean()

# compute average AUC for the second (lm)
mean_auc_lm <- predictions_lm %>%
  performance(measure="auc") %>%
  slot("y.values") %>% unlist() %>%
  mean()

# plot the ROC curve for the first (rf)
predictions_rf %>%
  performance(measure="tpr", x.measure="fpr") %>%
  plot(avg="threshold", col="orange", lwd=2)

# plot the ROC curve for the second (lm)
predictions_lm %>%
  performance(measure="tpr", x.measure="fpr") %>%
  plot(avg="threshold", col="blue", lwd=2, add=TRUE)

# add a legend to the plot
legend("bottomright",
       legend=paste(c("rf", "lm"), "AUC:", round(c(mean_auc_rf, mean_auc_lm), digits=3
)),
       col=c("orange", "blue"))
```



Conclusion

From all of this data, especially the ROC curve, we can see that a random forest is better at our prediction that a logistic regression model.

Random Forests can solve both type of problems (classification and regression) and do a decent estimation for both.

Another benfit is the power to handle large data sets with high dimensionality. Random Forests can handle thousands of input variavles and identify the most significant ones making it useful for dimensionality reduction.

Overall, for this data set the random forest was better with an error rate of 0.05158368 as apposed to the logistic regression model with an error rate of 0.07295034.