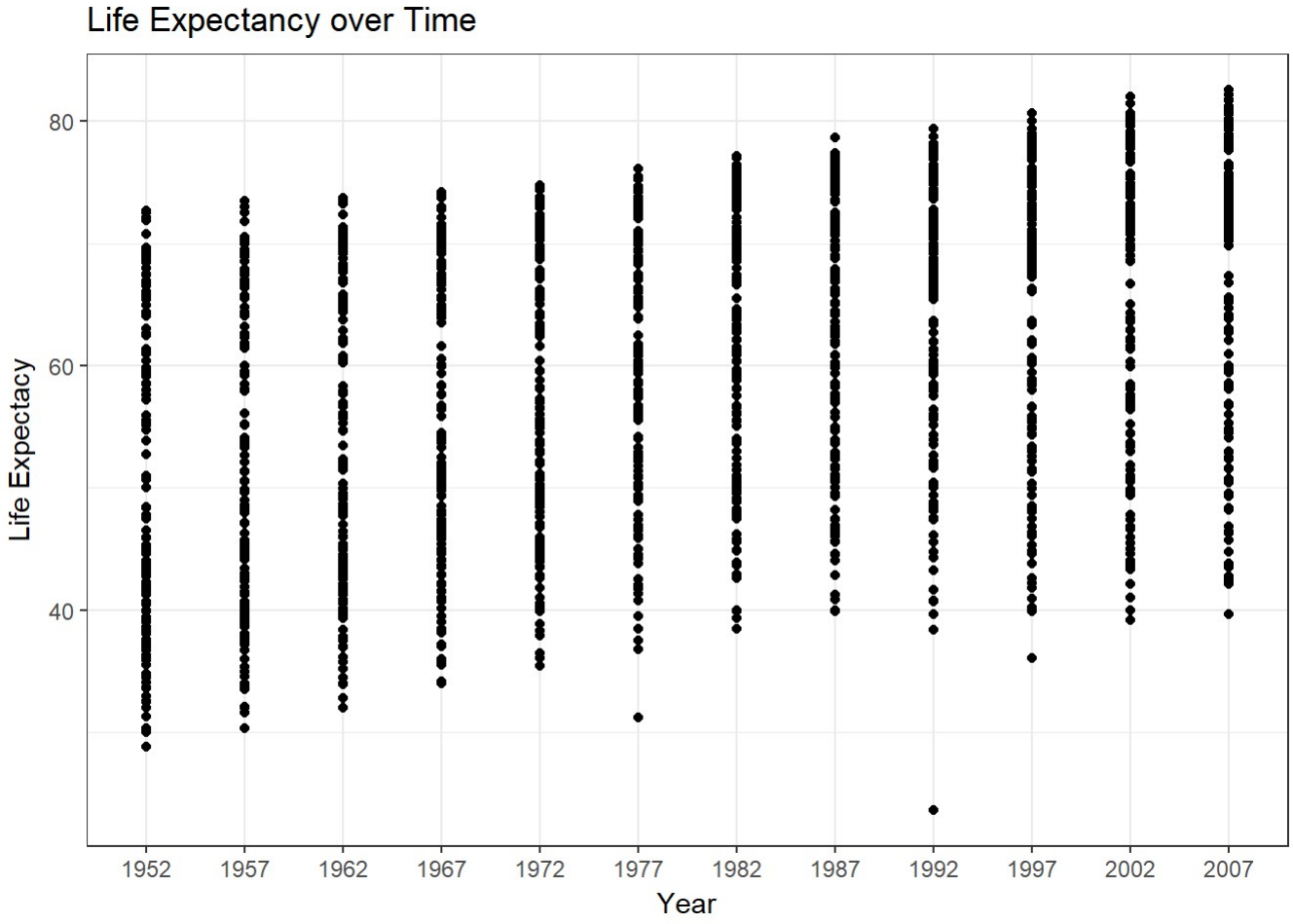
Regression

Q1

Based purely qualitatively, it appears that life expectancy increases over time in a linear fashion

```
df <- gapminder

df1 <- df %>% ggplot(aes(x= factor(year), y = lifeExp)) + geom_point() + labs(title =
"Life Expectancy over Time", x = "Year", y = "Life Expectacy")
df1
```

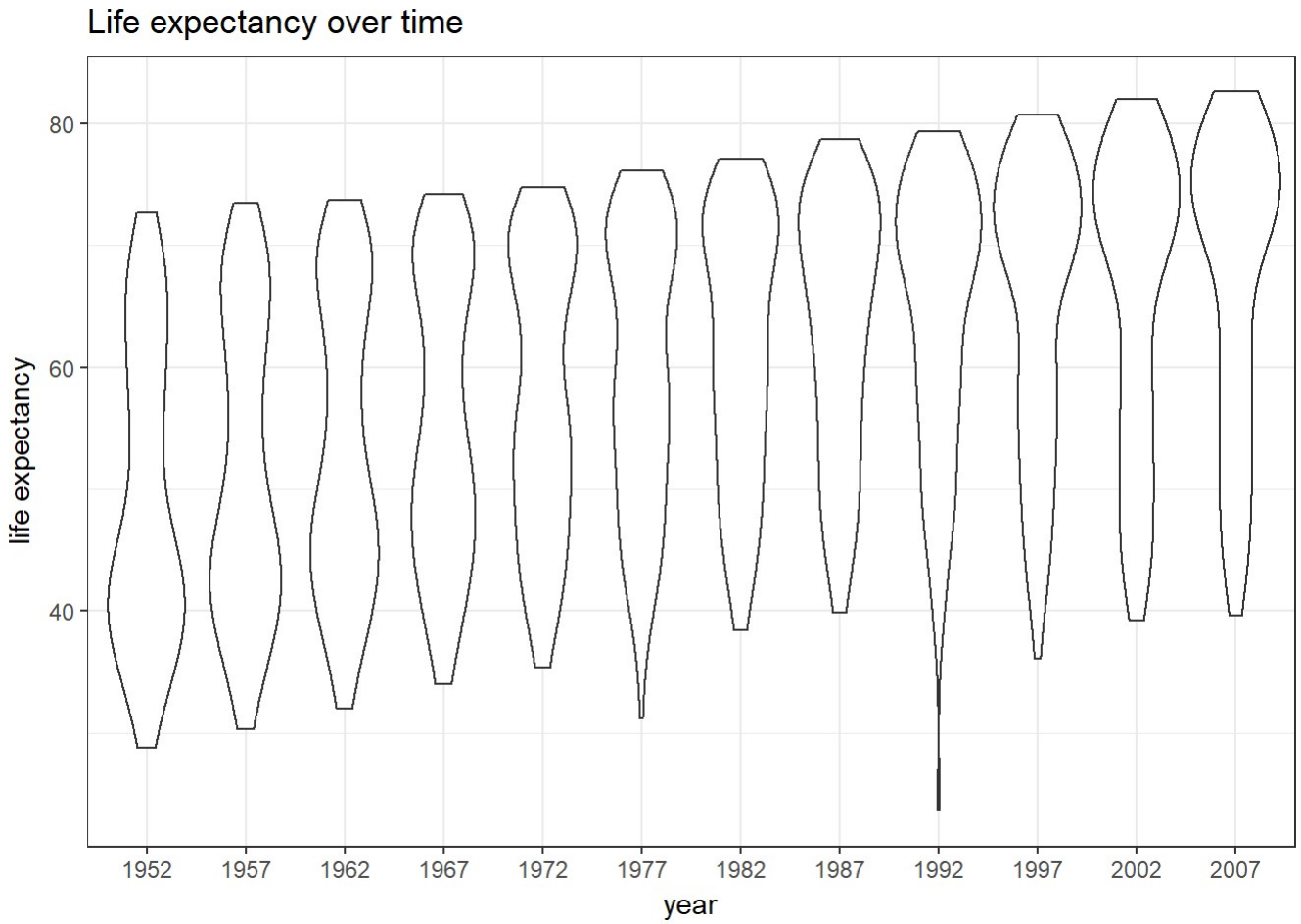


Q2

Most country's life expectancy is around 40 in 1952 and that is the lower echelon of life expectancy. Over time most country's lifeexpectancy bubbles up to the higher exchelon until 2007 where most country's are around 75and the least amount of countries are in the 40's. Our data is skewed because our range of data on one side of the median is larger than the other half of the median. In 1952, there is more range past the median, where as in the later years there is more range in data before the median(assuming counting increasingly 40…41..42.. etc.). In some years, our data is unimodal, meaning having one clear mode. For exaple in 1952 its

40, and in 2007 its around 75. However other years such as 1967 show signs of bimodal data where we have two peaks at a little bit less than 70 and mid 50's. None of the years really show any noticeable symmtery where the range on one half of the median is equal to the other half of the median. The plot below is essentially a box plot that shows density. Each violin corresponds to the distribution of life expectancy for that given year and the density is determined by the thickness of the violin plot at the given value.

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



Q3

Based on only intuition, I would reject the null hypothesis of no relationship. By looking at the violin plot it appears that there is indeed a realtionship between year and life expectancy. However, to prove this, we would need to do some calculations that prove that we reject the null hypothesis. The violin plot above has violins that have most of their distributions towards the bottom half(which is noticeable by the thickness of each violin) but then over time the most dense distribution occurs towards the top half (so this makes it look like an upside down violin).

Q4

A violin plot of residuals will look like a random distrubition with residuals randomly placed on the violin because i'm using my intuition and assuming life expectancy increases linearly.

Q5

If life expectnacy increases linearly then we can assume that the violin plot's will have violins look more like rectangles (uniformly distributed). The density will be around the same for each value ina given year (assuming linear).

E2

```
gapminder_fit <- lm(lifeExp~year, data=gapminder)
e2 <- gapminder_fit
broom::tidy(gapminder_fit)
```

```
##          term     estimate   std.error statistic      p.value
## 1 (Intercept) -585.6521874 32.31396452 -18.12381 2.897807e-67
## 2        year    0.3259038  0.01632369  19.96509 7.546795e-80
```

Q6

Based on our fit of a linear regression model, we extrapolate to say that every year, life expectancy increases by .3259038 years.

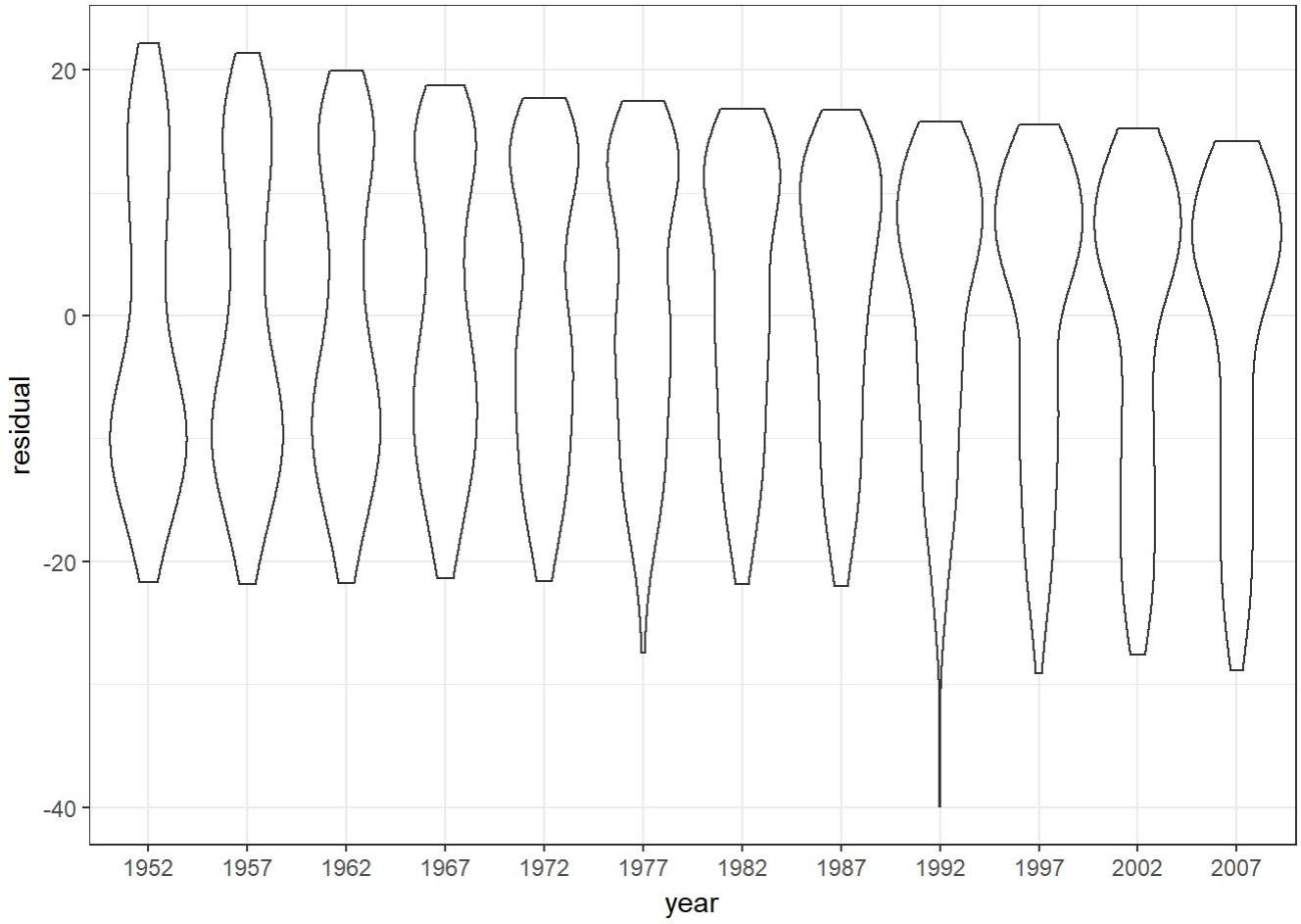
Q7

It is now easily seen why we would reject our null hypothesis of no relationship. The p value is essentially 0(7.5 times 10 to the negative 80).

E3

```
augmented_gapminder <- gapminder_fit %>%
  broom::augment()

augmented_gapminder %>%
ggplot(aes(x = factor(year), y = .resid)) + geom_violin() + geom_smooth() + labs(x =
"year", y=  "residual")
```

```
## `geom_smooth()` using method = 'loess'
```



Q8

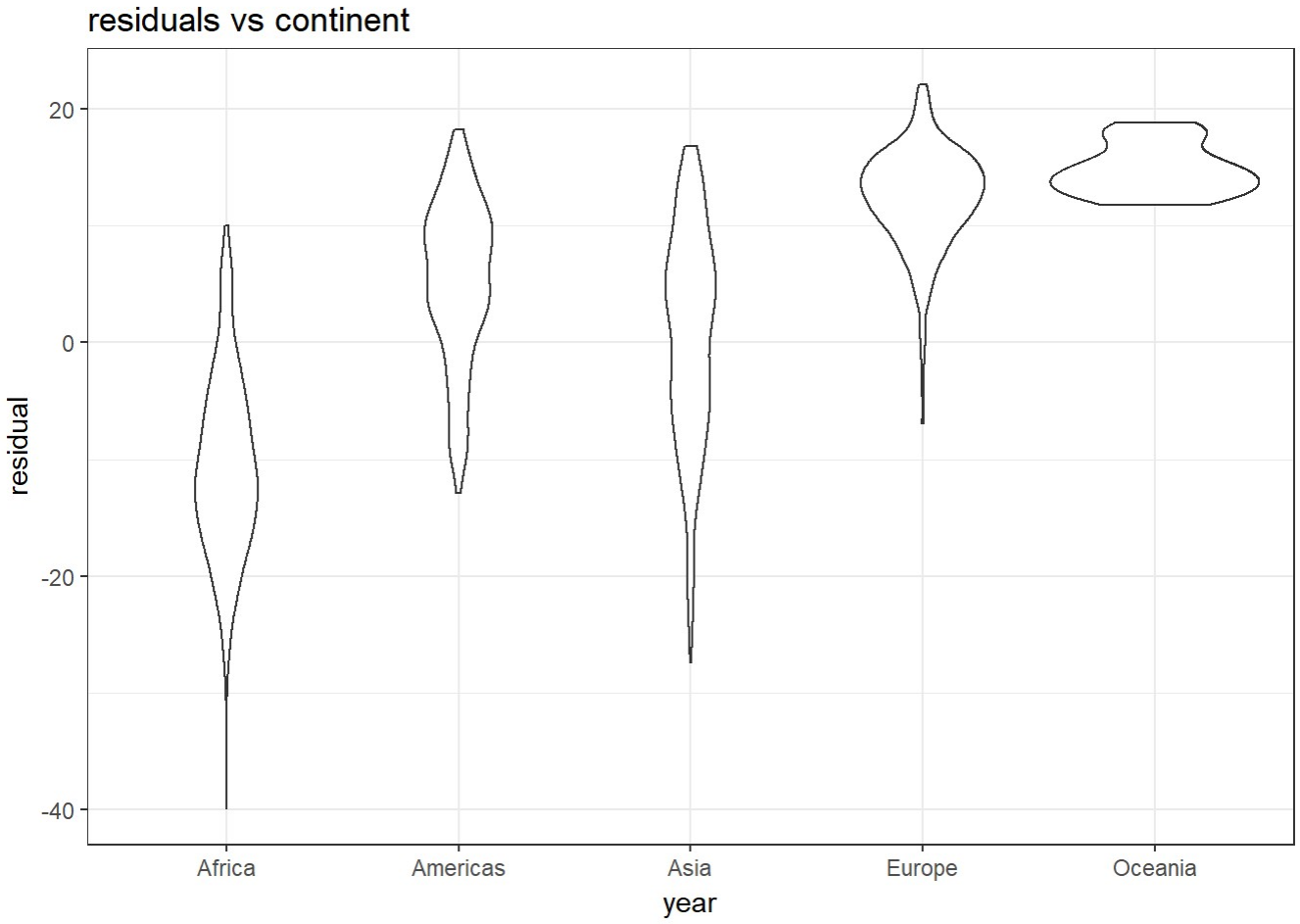
No. I was expecting the iolin plot to have “violins” that resembled a shape more like a rectangle. However, now that I have done the exercise the plot makes sense.

E4

```
augmented_gapminder %>% left_join(gapminder) %>%
ggplot(aes(x = continent, y = .resid)) + geom_violin() + geom_smooth() + labs(x = "ye
ar", y=  "residual", title = "residuals vs continent")
```

```
## Joining, by = c("lifeExp", "year")
```

```
## `geom_smooth()` using method = 'loess'
```

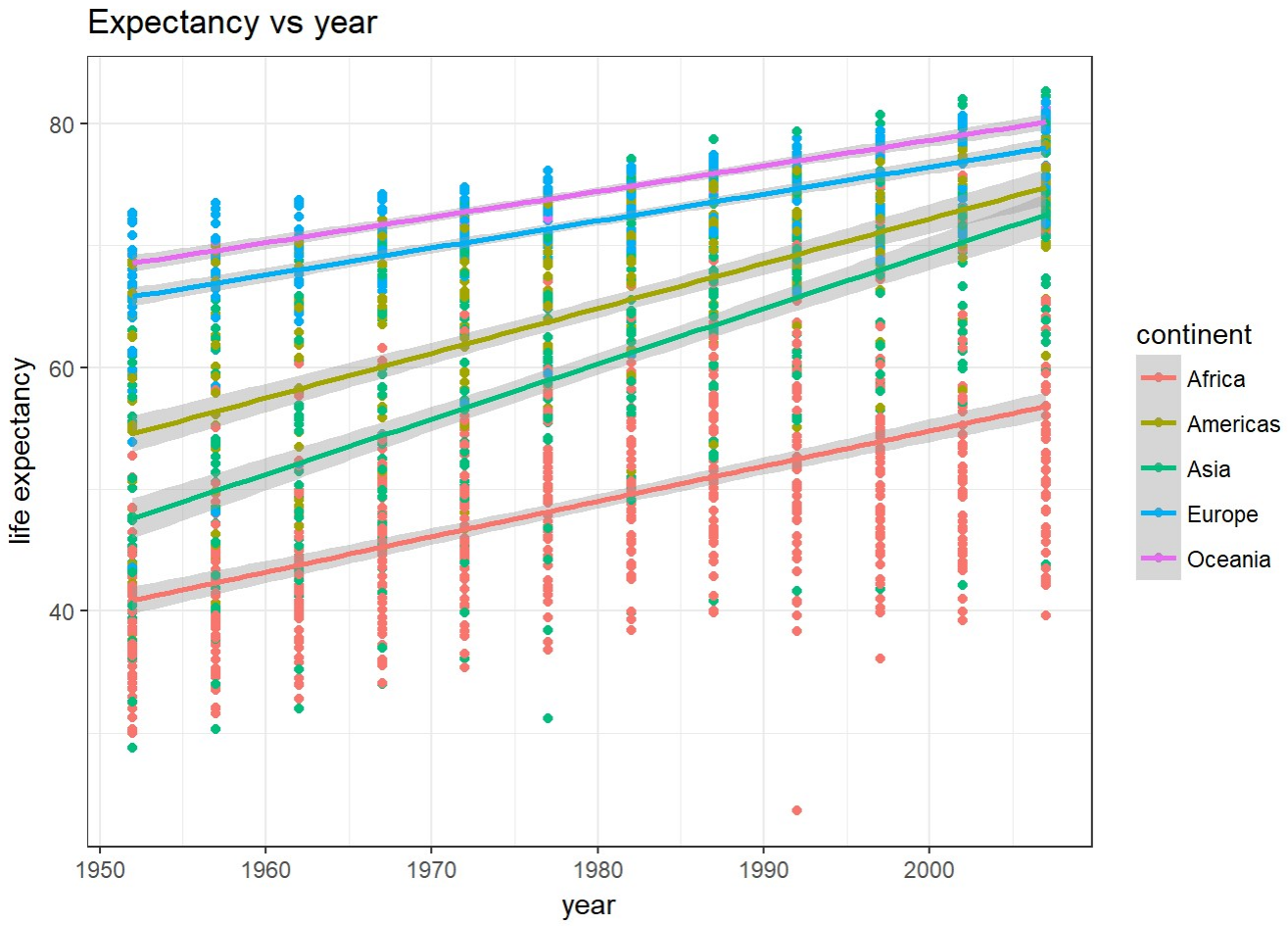


Q9

Yes. This means that we should take into account continent AND year in our model and analysis because that will be a more accurate fit.

E5

```
continent <- gapminder %>% group_by(continent) %>% ggplot(aes(x = year, y= lifeExp, c
olor = continent)) + geom_point() + geom_smooth(method = lm) + labs(y = "life expecta
ncy", title = "Expectancy vs year")
continent
```



Q10

Our model should include an interaction term for year and continent because different continents undergo different changes in life expectancy. The plot above shows how life expectancy changes over the years AND how it differs between each continent(which is why we assigned colors to each continent).

E6

```
gapminder_fit <- lm(lifeExp~year*continent, data=gapminder)
e6 <- gapminder_fit
e6_fit <- broom::tidy(gapminder_fit)
```

Q11

The p-values are very close to 0 (essentially 0) and therefore less than our alpha value (.05) and so our parameters are statistically different from 0. However the continent parameter for Oceania and the continent*year interaction for Oceania are not. Both of these p-values for Oceania are greater than our alpha value of .05 and so the estimates are not significantly different from 0 because they cross the. The p-values for the rest of the parameters are all less than 0.05 and so that makes the estimates significantly different from zero

Q12

```
e6_fit %>% select(term, estimate)
```

```
##                       term      estimate
## 1              (Intercept) -524.25784607
## 2                     year    0.28952926
## 3       continentAmericas -138.84844718
## 4           continentAsia -312.63304922
## 5         continentEurope  156.84685210
## 6        continentOceania  182.34988290
## 7  year:continentAmericas    0.07812167
## 8      year:continentAsia    0.16359314
## 9    year:continentEurope   -0.06759712
## 10  year:continentOceania   -0.07925689
```

E7

```
broom::tidy(anova(e2))
```

```
##        term   df     sumsq     meansq statistic      p.value
## 1      year    1  53919.18 53919.1842  398.6047 7.546795e-80
## 2 Residuals 1702 230229.20   135.2698        NA           NA
```

```
broom::tidy(anova(e6))
```

```
##             term   df      sumsq     meansq  statistic       p.value
## 1           year    1  53919.184 53919.1842 1046.02790 4.048499e-179
## 2      continent    4 139343.166 34835.7915  675.81159  0.000000e+00
## 3 year:continent    4   3566.089   891.5223   17.29546  6.463379e-14
## 4      Residuals 1694  87319.944    51.5466         NA            NA
```

Q13

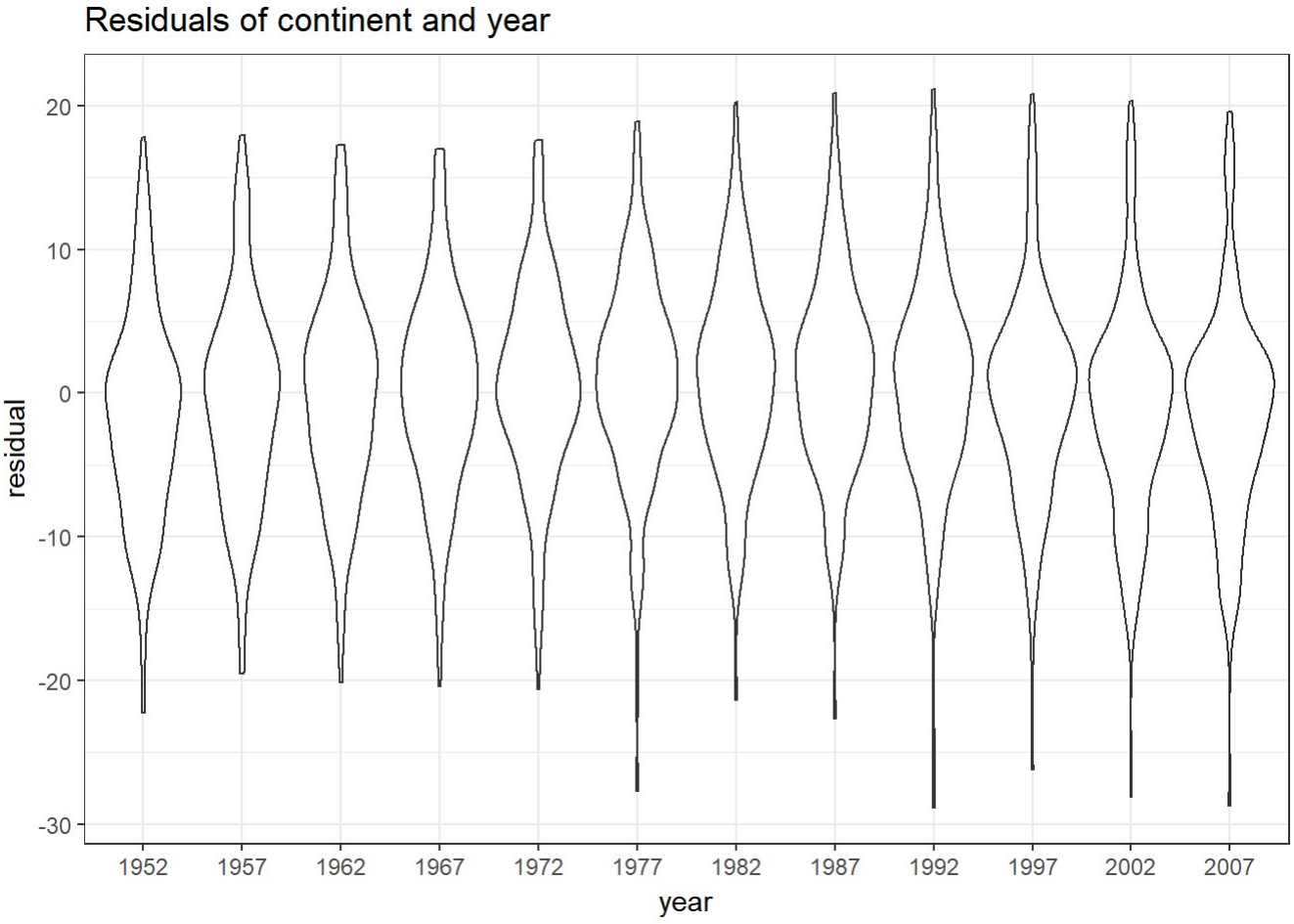
The interaction model is better than the year only model because the f value is higher (675.81159 for the year only model vs 1046.02790 for the interaction model). This means that the model using year and ocntinent is a much better model.

E8

The first plot is a violin plot of residuals for the continent and year model and as you can see they are much more uniformly distributed so the model is much better.

```
e6_augment <-e6 %>% broom::augment() %>% ggplot(aes(x = factor(year), y = .resid)) +
geom_violin() + geom_smooth() + labs(x = "year", y=  "residual", title = "Residuals o
f continent and year")
e6_augment
```

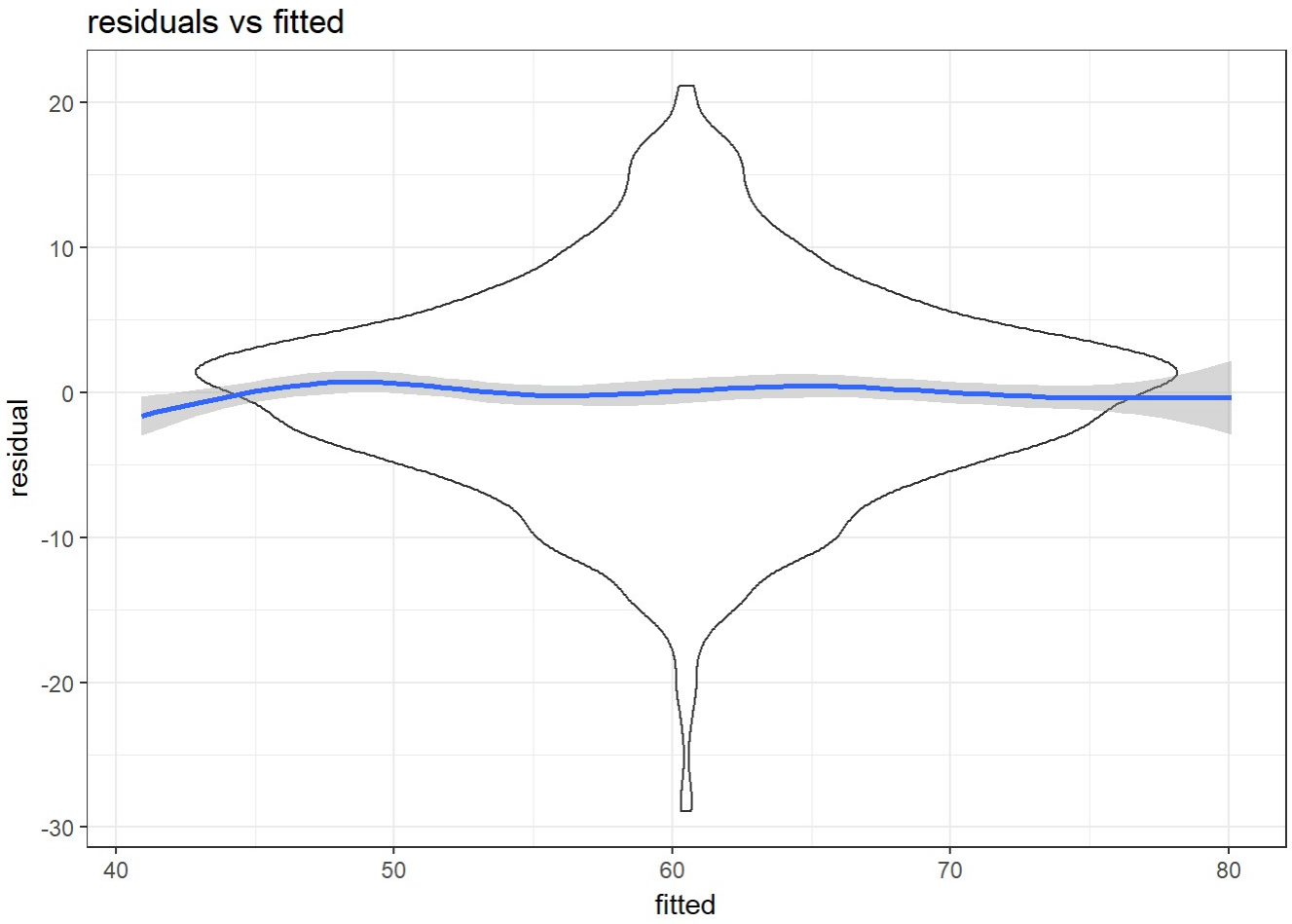
```
## `geom_smooth()` using method = 'loess'
```



The violin plot of residuals vs year with the interaction model matches the assumption of the linear regression model really well. The individual violins and distributions are more symmetric in regard to residuals. Thus, the interaction model is good. The violin plot of the residuals vs fitted values shows a single violin. It is very symmetric and the line resulting from using geom_smooth is quite horizontal. Thus, again, the interaction model is accurate and matches the linear regression model well.

```
e62_augment <-e6 %>% broom::augment() %>% ggplot(aes(x = .fitted, y = .resid)) + geom
_violin() + geom_smooth() + labs(x = "fitted", y=  "residual", title = "residuals vs
fitted")
e62_augment
```

```
## `geom_smooth()` using method = 'gam'
```



Tidy 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") %>%
  dplyr::select(RegionID, RegionName, matches("^[1|2]")) %>%
  gather(time, affordability, matches("^[1|2]")) %>%
  type_convert(col_types=cols(time=col_date(format="%Y-%m")))
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   RegionID = col_integer(),
##   RegionName = col_character(),
##   SizeRank = col_integer(),
##   Index = col_character()
## )
```

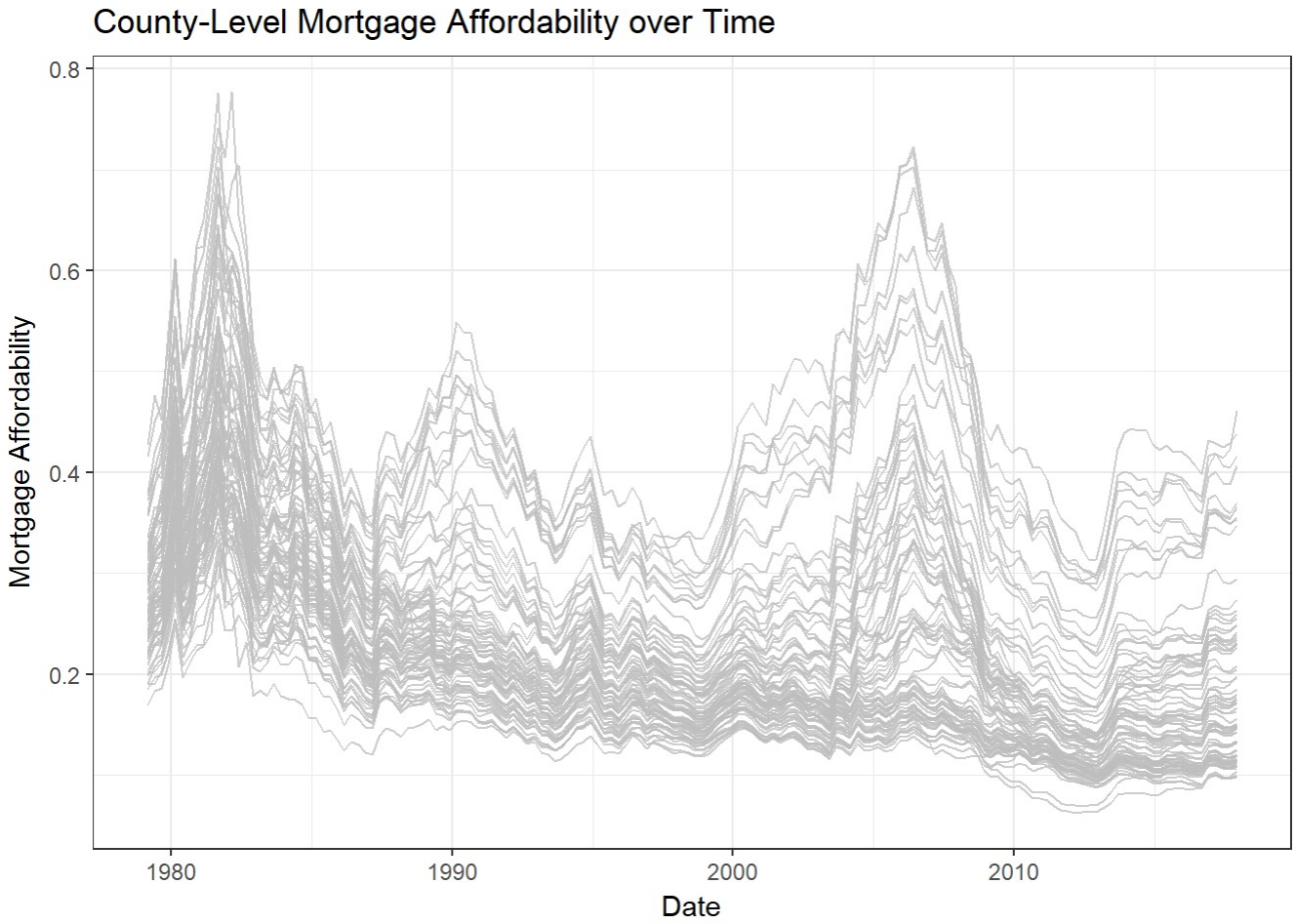
```
## See spec(...) for full column specifications.
```

```
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
```

Visualization

```
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 Affordability")
```



Prediction Task

```
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
```

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

Standardized

```
standardized_df <- predictor_df %>%
  filter(year(time) %in% 2014:2016) %>%
  group_by(RegionID) %>%
  mutate(mean_aff = mean(affordability)) %>%
  mutate(sd_aff = sd(affordability)) %>%
  mutate(z_aff = (affordability - mean_aff) / sd_aff) %>%
  ungroup()
standardized_df
```

```
## # A tibble: 960 x 7
##    RegionID RegionName   time       affordability mean_aff  sd_aff   z_aff
##       <int> <chr>        <date>             <dbl>    <dbl>   <dbl>   <dbl>
##  1   394913 New York, NY 2014-03-01         0.258    0.246 0.00854  1.41
##  2   753899 Los Angeles~ 2014-03-01         0.399    0.390 0.0111   0.799
##  3   394463 Chicago, IL  2014-03-01         0.142    0.136 0.00422  1.26
##  4   394514 Dallas-Fort~ 2014-03-01         0.123    0.127 0.00779 -0.503
##  5   394974 Philadelphi~ 2014-03-01         0.152    0.144 0.00518  1.68
##  6   394692 Houston, TX  2014-03-01         0.117    0.119 0.00545 -0.384
##  7   395209 Washington,~ 2014-03-01         0.185    0.177 0.00588  1.47
##  8   394856 Miami-Fort ~ 2014-03-01         0.178    0.194 0.0131  -1.17
##  9   394347 Atlanta, GA  2014-03-01         0.118    0.118 0.00384 -0.0115
## 10   394404 Boston, MA   2014-03-01         0.223    0.217 0.00662  0.875
## # ... with 950 more rows
```

```
wide_df <- standardized_df %>%
  select(RegionID, time, z_aff) %>%
  tidyr::spread(time, z_aff)
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)
final_df1 <- diff_df %>%
  inner_join(outcome_df %>% select(RegionID, Direction), by="RegionID") %>%
  mutate(Direction=factor(Direction, levels=c("down", "up")))
final_df1
```

```
## # A tibble: 80 x 13
##        V1      V2     V3      V4    V5      V6      V7     V8      V9
##     <dbl>   <dbl>  <dbl>   <dbl> <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
##  1 -0.482 -0.224  -1.13  -0.113  1.22   0.0828  0.270  -0.869 -0.187
##  2 -0.656 -0.0920 -1.01  -0.153  0.757 -0.0590 -0.138  -0.908 -0.449
##  3 -0.423 -0.251  -1.20  -0.387  0.646 -0.0271 -0.110  -0.718 -0.498
##  4 -0.308  0.0988 -0.938  0.179  1.26   0.0681  0.392  -0.874 -0.361
##  5 -0.371  0.0929 -1.03   0.150  1.60  -0.107   0.307  -0.541 -0.384
##  6 -0.496  0.239  -0.738  0.362  1.04  -0.0637  0.465  -0.607 -0.0931
##  7 -0.472 -0.118  -1.05  -0.458  0.509 -0.285   0.116  -0.844 -0.327
##  8 -0.485  0.261  -0.661 -0.0810 0.921  0.0145  0.427  -0.213  0.231
##  9 -0.438  0.527  -1.11   0.0971 0.720  0.191   0.186  -0.219  0.196
## 10 -0.919 -0.480  -1.01  -0.170  0.814 -0.407   0.0552 -0.684 -0.441
## # ... with 70 more rows, and 4 more variables: V10 <dbl>, V11 <dbl>,
## #   RegionID <int>, Direction <fct>
```

```
set.seed(1234)
test_random_forest_df <- final_df1 %>%
  group_by(Direction) %>%
  sample_frac(.2) %>%
  ungroup()

train_random_forest_df <- final_df1 %>%
  anti_join(test_random_forest_df, by="RegionID")

rf <- randomForest(Direction~., data=train_random_forest_df %>% select(-RegionID))
rf
```

```
##
## Call:
##  randomForest(formula = Direction ~ ., data = train_random_forest_df %>%      sele
ct(-RegionID))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 42.19%
## Confusion matrix:
##      down up class.error
## down    4 19    0.826087
## up      8 33    0.195122
```

```
test_predictions <- predict(rf, newdata=test_random_forest_df %>% select(-RegionID))
broom::tidy(table(pred=test_predictions, observed=test_random_forest_df$Direction))
```

```
##   pred observed Freq
## 1 down     down    2
## 2   up     down    4
## 3 down       up    1
## 4   up       up    9
```

Not Standardized

```
df <- predictor_df %>%
  filter(year(time) %in% 2014:2016) %>%
  group_by(RegionID) %>%
  ungroup()
df
```

```
## # A tibble: 960 x 4
##    RegionID RegionName                          time       affordability
##       <int> <chr>                               <date>             <dbl>
##  1   394913 New York, NY                        2014-03-01         0.258
##  2   753899 Los Angeles-Long Beach-Anaheim, CA 2014-03-01         0.399
##  3   394463 Chicago, IL                         2014-03-01         0.142
##  4   394514 Dallas-Fort Worth, TX               2014-03-01         0.123
##  5   394974 Philadelphia, PA                    2014-03-01         0.152
##  6   394692 Houston, TX                         2014-03-01         0.117
##  7   395209 Washington, DC                      2014-03-01         0.185
##  8   394856 Miami-Fort Lauderdale, FL           2014-03-01         0.178
##  9   394347 Atlanta, GA                         2014-03-01         0.118
## 10   394404 Boston, MA                          2014-03-01         0.223
## # ... with 950 more rows
```

```
wide_df <- df %>%
  select(RegionID, time, affordability) %>%
  tidyr::spread(time, affordability)
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)
final_df2 <- diff_df %>%
  inner_join(outcome_df %>% select(RegionID, Direction), by="RegionID") %>%
  mutate(Direction=factor(Direction, levels=c("down", "up")))
final_df2
```

```
## # A tibble: 80 x 13
##          V1        V2       V3       V4      V5       V6       V7       V8
##       <dbl>     <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
##  1 -0.00174 -0.000811 -0.00410 -4.08e-4 0.00441  3.00e-4  9.75e-4 -0.00314
##  2 -0.00375 -0.000527 -0.00579 -8.73e-4 0.00433 -3.38e-4 -7.90e-4 -0.00520
##  3 -0.00272 -0.00162  -0.00773 -2.49e-3 0.00416 -1.75e-4 -7.08e-4 -0.00462
##  4 -0.00118  0.000380 -0.00361  6.87e-4 0.00484  2.62e-4  1.50e-3 -0.00336
##  5 -0.00173  0.000432 -0.00480  6.99e-4 0.00744 -4.98e-4  1.43e-3 -0.00252
##  6 -0.00285  0.00137  -0.00425  2.08e-3 0.00598 -3.67e-4  2.67e-3 -0.00349
##  7 -0.00318 -0.000792 -0.00707 -3.08e-3 0.00343 -1.92e-3  7.81e-4 -0.00568
##  8 -0.00288  0.00155  -0.00393 -4.82e-4 0.00547  8.59e-5  2.54e-3 -0.00127
##  9 -0.00381  0.00458  -0.00964  8.43e-4 0.00625  1.66e-3  1.62e-3 -0.00190
## 10 -0.00422 -0.00220  -0.00465 -7.82e-4 0.00374 -1.87e-3  2.53e-4 -0.00314
## # ... with 70 more rows, and 5 more variables: V9 <dbl>, V10 <dbl>,
## #   V11 <dbl>, RegionID <int>, Direction <fct>
```

```
set.seed(1234)
test_random_forest_df <- final_df2 %>%
  group_by(Direction) %>%
  sample_frac(.2) %>%
  ungroup()

train_random_forest_df <- final_df2 %>%
  anti_join(test_random_forest_df, by="RegionID")

rf2 <- randomForest(Direction~., data=train_random_forest_df %>% select(-RegionID))
rf2
```

```
##
## Call:
##  randomForest(formula = Direction ~ ., data = train_random_forest_df %>%      sele
ct(-RegionID))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 40.62%
## Confusion matrix:
##      down up class.error
## down    5 18   0.7826087
## up      8 33   0.1951220
```

```
test_predictions <- predict(rf2, newdata=test_random_forest_df %>% select(-RegionID))
broom::tidy(table(pred=test_predictions, observed=test_random_forest_df$Direction))
```

```
##   pred observed Freq
## 1 down     down    3
## 2   up     down    3
## 3 down       up    0
## 4   up       up   10
```

Cross Validation

```
set.seed(1234)

# create the cross-validation partition
result_df <- createFolds(final_df1$Direction, k=10) %>%
  # fit models and gather results
  purrr::imap(function(test_indices, fold_number) {
    # split into train and test for the fold
    train_df1 <- final_df1 %>%
      select(-RegionID) %>%
      slice(-test_indices)

    train_df2 <- final_df2 %>%
      select(-RegionID) %>%
      slice(-test_indices)


    test_df1 <- final_df1 %>%
      select(-RegionID) %>%
      slice(test_indices)


    test_df2 <- final_df2 %>%
      select(-RegionID) %>%
      slice(test_indices)

    # fit the two models
    rf1 <- randomForest(Direction~., data=train_df1, ntree=500)
    rf2 <- randomForest(Direction~., data=train_df2, ntree=500)

    # gather results
    test_df1 %>%
      select(observed_label = Direction) %>%
      mutate(fold=fold_number) %>%
      mutate(prob_positive_rf1 = predict(rf1, newdata=test_df1, type="prob")[,"up"])
%>%
      # add predicted labels for rf1 using a 0.5 probability cutoff
      mutate(predicted_label_rf1 = ifelse(prob_positive_rf1 > 0.5, "up", "down")) %>%
      mutate(prob_positive_rf2 = predict(rf2, newdata=test_df2, type="prob")[, "up"])
%>%
      # add predicted labels for rf2 using a 0.5 probability cutoff
      mutate(predicted_label_rf2 = ifelse(prob_positive_rf2 > 0.5, "up", "down"))
}) %>%
  purrr::reduce(bind_rows)

  # gather res

result_df
```

```
## # A tibble: 80 x 6
##    observed_label fold  prob_positive_r~ predicted_label~ prob_positive_r~
##    <fct>          <chr>            <dbl> <chr>                       <dbl>
##  1 up             Fold~            0.472 down                        0.602
##  2 up             Fold~            0.614 up                          0.776
##  3 up             Fold~            0.748 up                          0.782
##  4 down           Fold~            0.394 down                        0.492
##  5 down           Fold~            0.376 down                        0.414
##  6 up             Fold~            0.394 down                        0.342
##  7 down           Fold~            0.496 down                        0.506
##  8 up             Fold~            0.890 up                          0.806
##  9 up             Fold~            0.746 up                          0.750
## 10 up             Fold~            0.788 up                          0.744
## # ... with 70 more rows, and 1 more variable: predicted_label_rf2 <chr>
```

Error Rates

```
result_df %>%
  mutate(error_rf1 = observed_label != predicted_label_rf1,
         error_rf2 = observed_label != predicted_label_rf2) %>%
  group_by(fold) %>%
  summarize(big_rf = mean(error_rf1), small_rf = mean(error_rf2)) %>%
  tidyr::gather(model, error, -fold) %>%
  lm(error~model, data=.) %>%
  broom::tidy()
```

```
##            term    estimate  std.error  statistic      p.value
## 1   (Intercept)  0.40119048 0.04820755  8.3221505 1.390436e-07
## 2 modelsmall_rf -0.03789683 0.06817577 -0.5558694 5.851441e-01
```

AUROC Curve

```
library(ROCR)
```

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

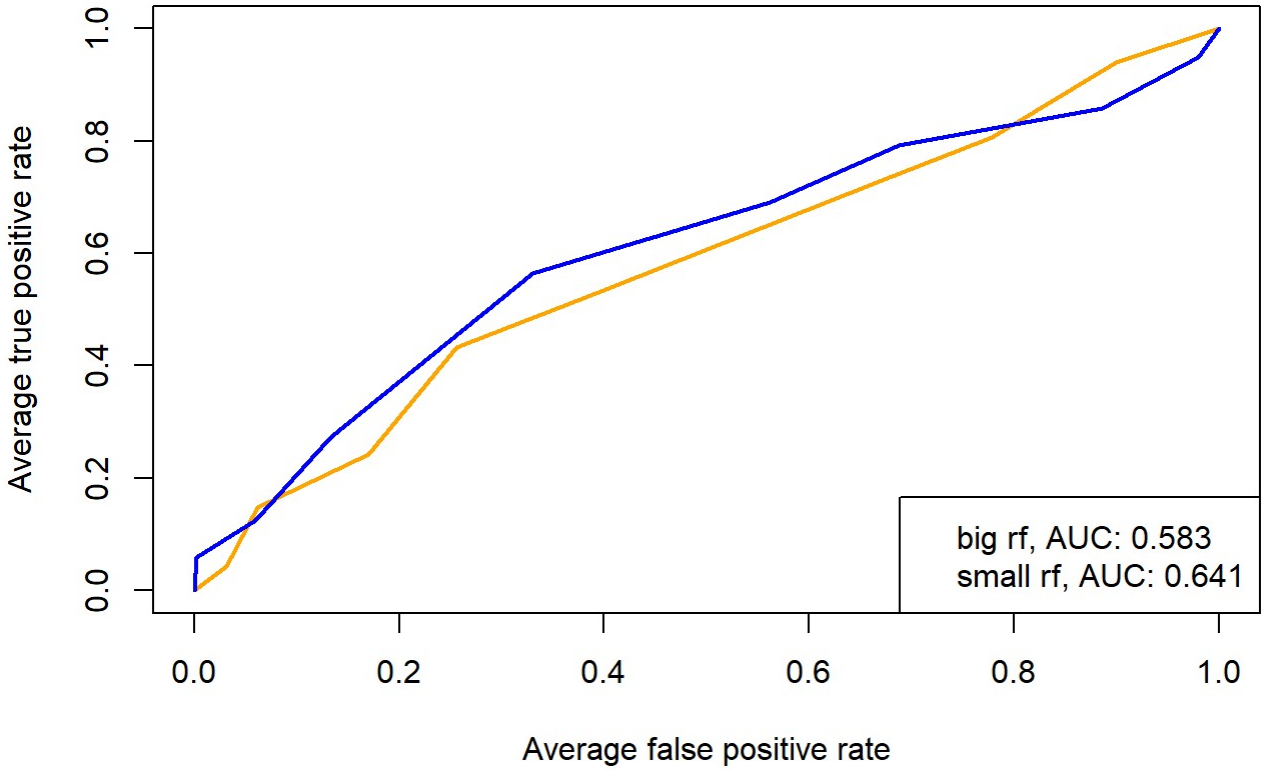
```
## Loading required package: gplots
```

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

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
# 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::predicti
on function
predictions_rf1 <- split(result_df$prob_positive_rf1, result_df$fold) %>% prediction(
labels)

# do the same for the second RF
predictions_rf2 <- split(result_df$prob_positive_rf2, result_df$fold) %>% prediction(
labels)

# compute average AUC for the first RF
mean_auc_rf1 <- predictions_rf1 %>%
  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 RF
mean_auc_rf2 <- predictions_rf2 %>%
  performance(measure="auc") %>%
  slot("y.values") %>% unlist() %>%
  mean()

# plot the ROC curve for the first RF
predictions_rf1 %>%
  performance(measure="tpr", x.measure="fpr") %>%
  plot(avg="threshold", col="orange", lwd=2)

# plot the ROC curve for the second RF
predictions_rf2 %>%
  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("big", "small"), "rf, AUC:", round(c(mean_auc_rf1, mean_auc_rf2
), digits=3)),
       col=c("orange", "blue"))
```



The question I chose to analyze is if a dataset with standardized affordability has better performance in predictions using Random Forest classification with 500 trees. Interpretation: With the results, we can see that the error rate when you standardize the affordability is 0.31 while the error rate when you don't standardize is 0.25. Thus, by only analyzing error rates, standardizing the data did not prove to improve performance. Looking at the hypothesis testing between the two approaches, we can see that the p-value is greater than 0.05 so the error difference is not significant. In addition, the AUROC curve plot shows that the non-standardized line is above the standardized line which means not standardizing performs better. The average true positive rate is higher when the false positive rate is low