Regression Analysis of Creative Jobs in US

Zoe Chen

2019/2/11

0. Introduction

In this data analysis case, we want to deal with an interesting topic: What decide a county in US to have more creative jobs than others. To be more specific, we will focus on using regression methods to understand how different predictors influence the number of creative jobs in each area.

1. Data Collection and Preprocessing

Importing Data Sets

One good news is that we can collect several useful county-level data sets from USDA(United States Department of Agriculture) website. Here we import five data sets which are `creative.df`, `education.df`, `pupulation.df`, `poverty.df` and `unemployment.df`. Now we don't have any idea that what predictors have more impact on number of creative jobs, but we will figure it out later.

```
knitr::opts_chunk$set(fig.width=10, fig.height=6)
creative.df = read.csv("./creativeclass200711.csv", skip=0, stringsAsFactors = F)
education.df = read.csv("./Education.csv", skip=4, stringsAsFactors = F)
population.df = read.csv("./PopulationEstimates.csv", skip=2, stringsAsFactors = F)
poverty.df = read.csv("./PovertyEstimates.csv", skip=3, stringsAsFactors = F)
unemployment.df = read.csv("./Unemployment.csv", skip=7, stringsAsFactors = F)
```

Getting Basic Understanding of Data Sets

For each data set, we want to first know more about their dimensions and their N/A ratios. The definition of N/A ratio here is simply total number of N/A divided by total cells that a dataframe has.

From the result table below we see that for data set dimensions, they have similar rows because they are all county-level data sets. As for columns, they have different numbers ranging from 20 up to 133.

Next, let's see the computed N/A ratio result for each data set. We can notice that `creative.df` is the most complete data set and it even doesn't contain any N/A value. On the other hand, 17.58% of data in `poverty.df` is N/A, which is the highest among these five dataframes. Thus we take a deeper look at this data set and we find out the data in certain columns is very scarce. That is the reason why N/A ratio in this data set turns out to be so high.

```
calcuate_na = function(df){
  total = nrow(df) * ncol(df)
  ratio = sum(is.na(df)) / total
  return(ratio)
}

na.ratio = c()
df.row = c()
```

```
df.col = c()
for (df in list(creative.df, education.df, population.df, poverty.df, unemployment.df)){
  ratio = calcuate_na(df)
  na.ratio = c(na.ratio, ratio)
  df.row = c(df.row, nrow(df))
  df.col = c(df.col, ncol(df))
}

df.names = c('creative.df', 'education.df', 'population.df', 'poverty.df',
             'unemployment.df')
ratio.df = data.frame(df.names, df.row, df.col, na.ratio)
print(ratio.df)
```

```
##          df.names df.row df.col    na.ratio
## 1     creative.df   3140     20 0.000000000
## 2    education.df   3283     47 0.009183349
## 3   population.df   3273    133 0.021701826
## 4      poverty.df   3194     34 0.175798372
## 5 unemployment.df   3275     52 0.002853787
```

Merging Data Sets

In this part, we merge `creative.df`, `education.df`, `population.df`, `poverty.df`, `unemployment.df` into another dataframe only with those columns that are meaningful and reasonable to the response variable. The merged data set contains totally 3133 rows and 39 columns. Also, during the merging process, we automatically omit those rows that have N/A value, so in this case we won't deal with N/A values. Next, to make it easier to understand, let's also change the columns into other more reasonable and readable names. Below is the columns in the new dataframe. Our response variable is `total.creative`.

```
merged.df = merge(x=creative.df[, c(1:6, 8, 10)], y=education.df[, c(1, 40:47)],
                  by.x = 'FIPS', by.y = 'FIPS.Code')
merged.df = merge(x=merged.df, y=population.df[, c(1, 11:12, 27:28, 51:52, 59:60)],
                  by.x = 'FIPS', by.y = 'FIPS')
merged.df = merge(x=merged.df, y=poverty.df[, c(1, 8, 11, 14, 17, 26)], by.x = 'FIPS',
                  by.y = 'FIPStxt')
merged.df = merge(x=merged.df, y=unemployment.df[, c(1, 8:9, 12:13, 16:17, 20:21, 24:25)],
                  by.x = 'FIPS', by.y = 'FIPStxt')

new.df = merged.df
colnames(new.df) = c('FIPS', 'state', 'state.abr', 'country', 'metro',
                     'total.emp', 'total.creative', 'percent.creative',
                     'less.than.highschool', 'highschool.only', 'college',
                     'bachelor.higher', 'percent.less.than.highschool',
                     'percent.highschool.only', 'percent.college',
                     'percent.bachelor.higher', 'pop.2010', 'pop.2011',
                     'birth.2010', 'birth.2011', 'international.mig.2010',
                     'international.mig.2011', 'domestic.mig.2010',
                     'domestic.mig.2011', 'all.pov.2016', 'percent.all.pov.2016',
                     '0.17.pov.2016', 'percent.0.17.pov.2016', 'median.income.2016',
                     'employed.2007', 'unemployed.2007', 'employed.2008',
                     'unemployed.2008', 'employed.2009', 'unemployed.2009',
                     'employed.2010', 'unemployed.2010', 'employed.2011',
```

```
                     'unemployed.2011')

colnames(new.df)
```

```
##  [1] "FIPS"                          "state"
##  [3] "state.abr"                     "country"
##  [5] "metro"                         "total.emp"
##  [7] "total.creative"                "percent.creative"
##  [9] "less.than.highschool"          "highschool.only"
## [11] "college"                       "bachelor.higher"
## [13] "percent.less.than.highschool" "percent.highschool.only"
## [15] "percent.college"               "percent.bachelor.higher"
## [17] "pop.2010"                      "pop.2011"
## [19] "birth.2010"                    "birth.2011"
## [21] "international.mig.2010"        "international.mig.2011"
## [23] "domestic.mig.2010"             "domestic.mig.2011"
## [25] "all.pov.2016"                  "percent.all.pov.2016"
## [27] "0.17.pov.2016"                 "percent.0.17.pov.2016"
## [29] "median.income.2016"            "employed.2007"
## [31] "unemployed.2007"               "employed.2008"
## [33] "unemployed.2008"               "employed.2009"
## [35] "unemployed.2009"               "employed.2010"
## [37] "unemployed.2010"               "employed.2011"
## [39] "unemployed.2011"
```

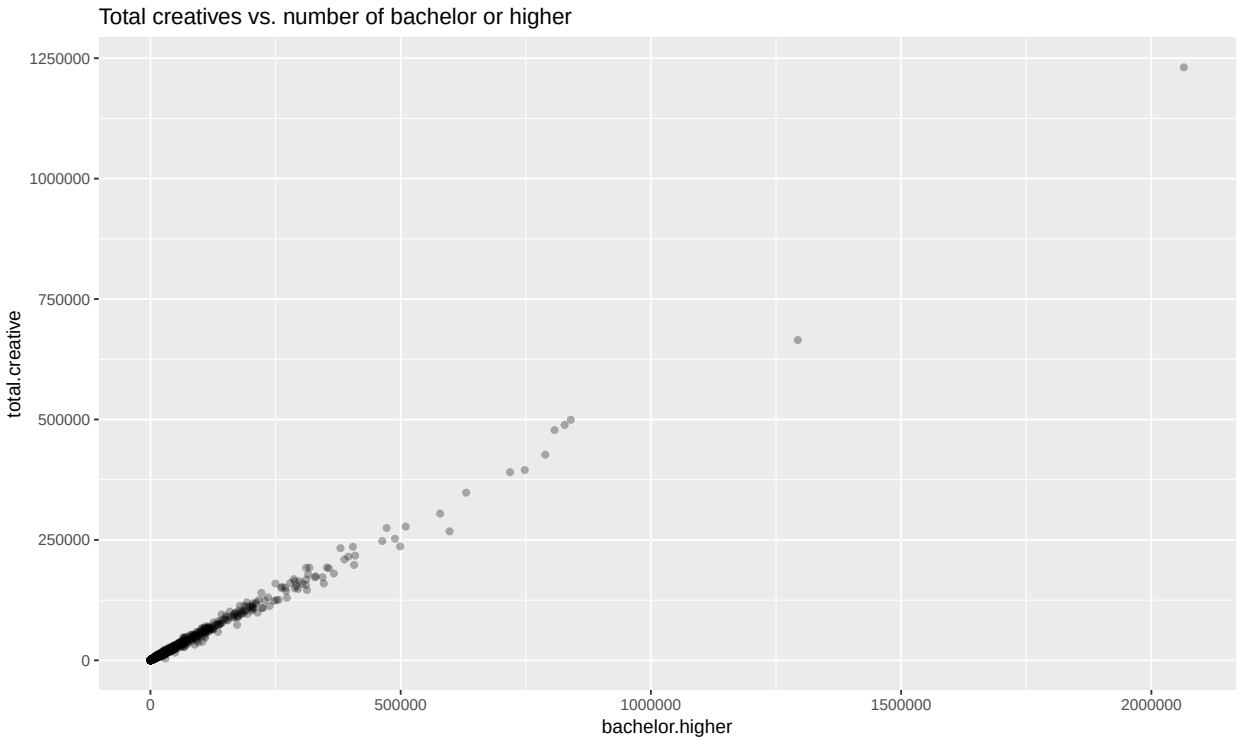
2. Doing Preliminary Analysis and Building Hypotheses

We can make some hypotheses based on simple preliminary plots. Here we can pick some predictors based on our domain knowledge that may be possible to be significant independent variables. Let's choose `bachelor.higher`, `international.mig.2010` and `all.pov.2016` and draw the scatter plots respectively. From the plots we can also make some hypotheses. After building the models, we can come back and check these hypotheses again.

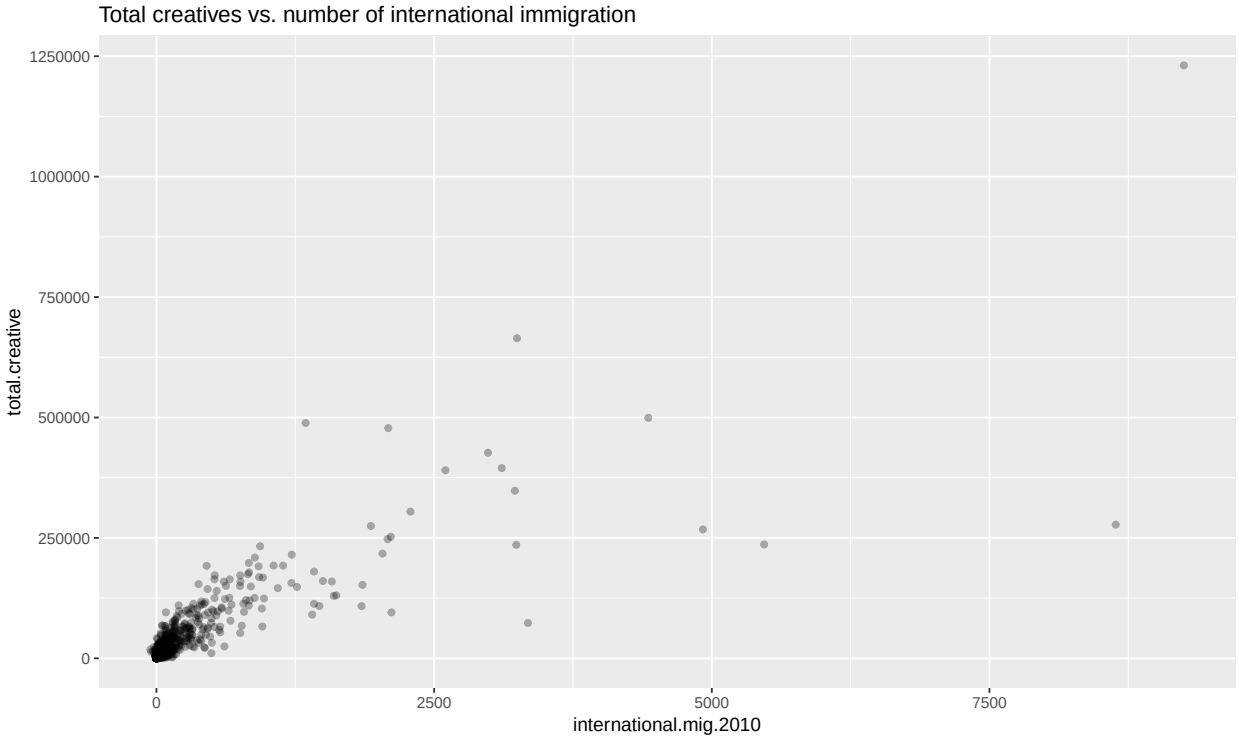
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

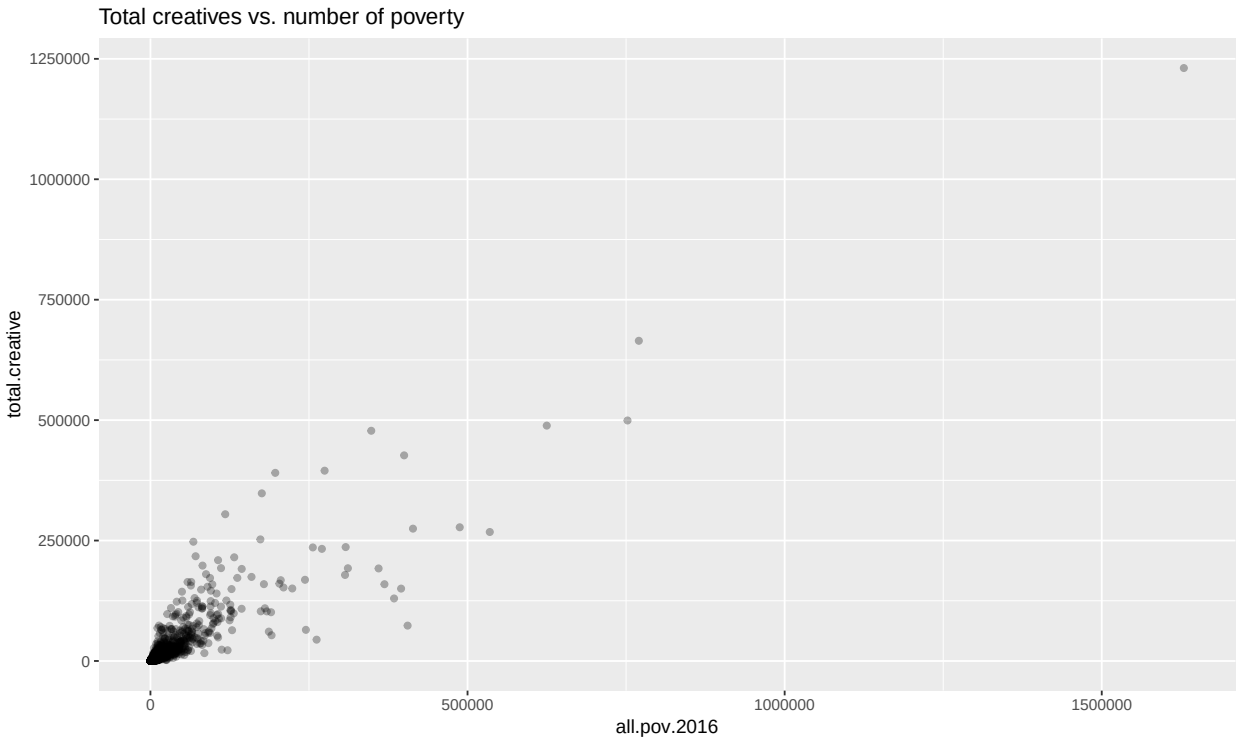
```
ggplot(new.df, aes(x=bachelor.higher, y=total.creative)) + geom_point(alpha = 0.3) +
  ggtitle('Total creatives vs. number of bachelor or higher')
```




```
ggplot(new.df, aes(x=international.mig.2010, y=total.creative)) + geom_point(alpha = 0.3) +
  ggtitle('Total creatives vs. number of international immigration')
```



```
ggplot(new.df, aes(x=all.pov.2016, y=total.creative)) + geom_point(alpha = 0.3) +
  ggtitle('Total creatives vs. number of poverty')
```



3. Splitting Data Sets

Then let's split 80% percent of data into training set and the rest into test set.

```
new.df$metro = as.factor(new.df$metro)
set.seed(111)
sample_row = sample(nrow(new.df), size = .8*nrow(new.df), replace = F)
train.df = new.df[sample_row,]
test.df = new.df[-sample_row,]
dim(train.df)
```

```
## [1] 2506   39
```

```
dim(test.df)
```

```
## [1] 627  39
```

4. Building Simple Linear Regrsssion Models

We first build 5 simple linear regression models. Surprisingly, from the fitted model we find that in some models, even one independent variable can result in very high R-adjusted value.

```
simple.1 = lm(total.creative ~ total.emp, data = train.df)
simple.2 = lm(total.creative ~ bachelor.higher, data = train.df)
simple.3 = lm(total.creative ~ international.mig.2010, data = train.df)
simple.4 = lm(total.creative ~ all.pov.2016, data = train.df)
simple.5 = lm(total.creative ~ median.income.2016, data = train.df)
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 3.5.2
```

```
i = 1
for (model in list(simple.1, simple.2, simple.3, simple.4, simple.5)){
  train.prediction = predict(model, train.df)
  test.prediction = predict(model, newdata = test.df)
  train.rmse = rmse(train.df$total.creative, train.prediction)
  test.rmse = rmse(test.df$total.creative, test.prediction)
  i = i+1
}

summary(simple.1)
```

```
##
## Call:
## lm(formula = total.creative ~ total.emp, data = train.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -78794   -855    201    765 157195
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.187e+03  1.594e+02  -7.446 1.31e-13 ***
## total.emp    2.825e-01  9.872e-04 286.106  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7634 on 2504 degrees of freedom
## Multiple R-squared:  0.9703, Adjusted R-squared:  0.9703
## F-statistic: 8.186e+04 on 1 and 2504 DF,  p-value: < 2.2e-16
```

```
summary(simple.2)
```

```
##
## Call:
## lm(formula = total.creative ~ bachelor.higher, data = train.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -59040   -154      3    340  75589
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.814e+01  7.072e+01   1.105    0.269
```

```
## bachelor.higher 5.595e-01  8.647e-04 646.999   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3417 on 2504 degrees of freedom
## Multiple R-squared:  0.9941, Adjusted R-squared:  0.9941
## F-statistic: 4.186e+05 on 1 and 2504 DF,  p-value: < 2.2e-16
```

```
summary(simple.3)
```

```
##
## Call:
## lm(formula = total.creative ~ international.mig.2010, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -648187   -4956   -4046   -1225  339943
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5465.944    468.801   11.66   <2e-16 ***
## international.mig.2010  106.552      1.302   81.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23120 on 2504 degrees of freedom
## Multiple R-squared:  0.7278, Adjusted R-squared:  0.7277
## F-statistic:  6695 on 1 and 2504 DF,  p-value: < 2.2e-16
```

```
summary(simple.4)
```

```
##
## Call:
## lm(formula = total.creative ~ all.pov.2016, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -234326   -2841   -1504    -805  214473
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  884.58773  376.18800   2.351   0.0188 *
## all.pov.2016   0.75673    0.00679 111.445   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18150 on 2504 degrees of freedom
## Multiple R-squared:  0.8322, Adjusted R-squared:  0.8321
## F-statistic: 1.242e+04 on 1 and 2504 DF,  p-value: < 2.2e-16
```

```
summary(simple.5)
```

```
##
## Call:
## lm(formula = total.creative ~ median.income.2016, data = train.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -85596  -11921   -4357    2948 1205534
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -4.325e+04  3.333e+03  -12.97   <2e-16 ***
## median.income.2016  1.119e+00  6.527e-02   17.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 41920 on 2504 degrees of freedom
## Multiple R-squared:  0.1051, Adjusted R-squared:  0.1047
## F-statistic:   294 on 1 and 2504 DF,  p-value: < 2.2e-16
```

5. Building Multiple Linear Regrssion Models

Now let's try to build multiple linear regression models and apply feature selection function. Based on the previous results of simple linear regression models, we can assume that the multiple regression models will also have strong fit. It turns out that it's true. From the computation of RMSE on both training set and test set, since they don't have huge difference, the model isn't overfitting.

```
m1 = lm(total.creative ~.
        -FIPS -state -state.abr -country -percent.creative
        -percent.less.than.highschool -percent.highschool.only -percent.college -percent.bachelor.highe
        - pop.2011 -percent.all.pov.2016 -percent.0.17.pov.2016 -employed.2007 -employed.2008
        -employed.2009 -employed.2010 -employed.2011, data = train.df)

summary(m1)
```

```
##
## Call:
## lm(formula = total.creative ~ . - FIPS - state - state.abr -
##     country - percent.creative - percent.less.than.highschool -
##     percent.highschool.only - percent.college - percent.bachelor.higher -
##     pop.2010 - pop.2011 - percent.all.pov.2016 - percent.0.17.pov.2016 -
##     employed.2007 - employed.2008 - employed.2009 - employed.2010 -
##     employed.2011, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19263.9   -294.8    -67.8    296.0  24824.1
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             9.436e+01  1.586e+02   0.595  0.55198
## metro1                  1.828e+02  8.474e+01   2.157  0.03107 *
## total.emp               2.501e-01  6.232e-03  40.128  < 2e-16 ***
```

```
## less.than.highschool   -2.423e-02  8.094e-03  -2.993  0.00279 **
## highschool.only        -1.966e-01  5.810e-03 -33.845  < 2e-16 ***
## college                -6.477e-02  6.742e-03  -9.607  < 2e-16 ***
## bachelor.higher         3.082e-01  5.481e-03  56.226  < 2e-16 ***
## birth.2010             -2.069e+01  1.316e+00 -15.718  < 2e-16 ***
## birth.2011              4.723e+00  3.643e-01  12.964  < 2e-16 ***
## international.mig.2010 -3.632e+00  1.422e+00  -2.554  0.01070 *
## international.mig.2011  4.271e-01  3.377e-01   1.265  0.20613
## domestic.mig.2010       1.357e-01  1.239e-01   1.095  0.27357
## domestic.mig.2011       1.890e-01  3.890e-02   4.859 1.25e-06 ***
## all.pov.2016           -6.278e-02  1.020e-02  -6.152 8.91e-10 ***
## `0.17.pov.2016`         4.142e-02  2.953e-02   1.402  0.16091
## median.income.2016      3.787e-04  3.341e-03   0.113  0.90976
## unemployed.2007        -5.184e-01  5.359e-02  -9.674  < 2e-16 ***
## unemployed.2008         3.619e-02  7.844e-02   0.461  0.64456
## unemployed.2009         2.310e-01  7.178e-02   3.219  0.00130 **
## unemployed.2010         9.046e-01  1.211e-01   7.470 1.11e-13 ***
## unemployed.2011        -9.057e-01  8.233e-02 -11.001  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1664 on 2485 degrees of freedom
## Multiple R-squared:  0.9986, Adjusted R-squared:  0.9986
## F-statistic: 8.864e+04 on 20 and 2485 DF,  p-value: < 2.2e-16
```

```
library(MASS)
m2.stepwise = stepAIC(m1, direction = 'both', trace = F)
summary(m2.stepwise)
```

```
##
## Call:
## lm(formula = total.creative ~ metro + total.emp + less.than.highschool +
##     highschool.only + college + bachelor.higher + birth.2010 +
##     birth.2011 + international.mig.2010 + domestic.mig.2011 +
##     all.pov.2016 + `0.17.pov.2016` + unemployed.2007 + unemployed.2009 +
##     unemployed.2010 + unemployed.2011, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19081.7   -290.1    -68.1    297.5  24520.4
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             108.760097  42.761588   2.543 0.011038 *
## metro1                  183.604119  78.786264   2.330 0.019864 *
## total.emp                 0.249527   0.005771  43.238  < 2e-16 ***
## less.than.highschool     -0.020708   0.007601  -2.724 0.006487 **
## highschool.only          -0.195436   0.005601 -34.891  < 2e-16 ***
## college                  -0.065243   0.006392 -10.208  < 2e-16 ***
## bachelor.higher           0.310356   0.005256  59.043  < 2e-16 ***
## birth.2010              -20.421112   1.241562 -16.448  < 2e-16 ***
## birth.2011                4.617614   0.341094  13.538  < 2e-16 ***
## international.mig.2010   -1.921650   0.295303  -6.507 9.21e-11 ***
## domestic.mig.2011         0.219954   0.030954   7.106 1.56e-12 ***
```

```
## all.pov.2016            -0.065073   0.009847  -6.609 4.73e-11 ***
## `0.17.pov.2016`          0.050876   0.028326   1.796 0.072600 .
## unemployed.2007         -0.498881   0.045269 -11.020  < 2e-16 ***
## unemployed.2009          0.214384   0.061400   3.492 0.000488 ***
## unemployed.2010          0.940041   0.112169   8.381  < 2e-16 ***
## unemployed.2011         -0.915588   0.072870 -12.565  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1664 on 2489 degrees of freedom
## Multiple R-squared:  0.9986, Adjusted R-squared:  0.9986
## F-statistic: 1.109e+05 on 16 and 2489 DF,  p-value: < 2.2e-16
```

```
train.prediction = predict(m2.stepwise, train.df)
test.prediction = predict(m2.stepwise, newdata = test.df)
train.rmse = rmse(train.df$total.creative, train.prediction)
test.rmse = rmse(test.df$total.creative, test.prediction)
print(paste('RMSE on training set:', train.rmse))
```

```
## [1] "RMSE on training set: 1657.95234008369"
```

```
print(paste('RMSE on test set:', test.rmse))
```

```
## [1] "RMSE on test set: 1926.5550960853"
```

6. Conclusion

Based on the results we see that there is strong relationship between the response variable, the number of creative jobs, with other variables. Besides, the model can be generalized to unseen data well too. To sum up, we can tell that the most important predictors are total number of employment, education background distribution, number of immigration and employment rate.