Team Cauchy Distribution

Boston University HACKATHON

Samhita Surapaneni, Weichao Wu, Abhinav Tyagi, Xiaoyi Geng

INTRODUCTION

Our team was given he billionaires data set that contained information about 2000+ billionaires across the world as well as what industries they accumulated their wealth from, in addition to other vital metrics.

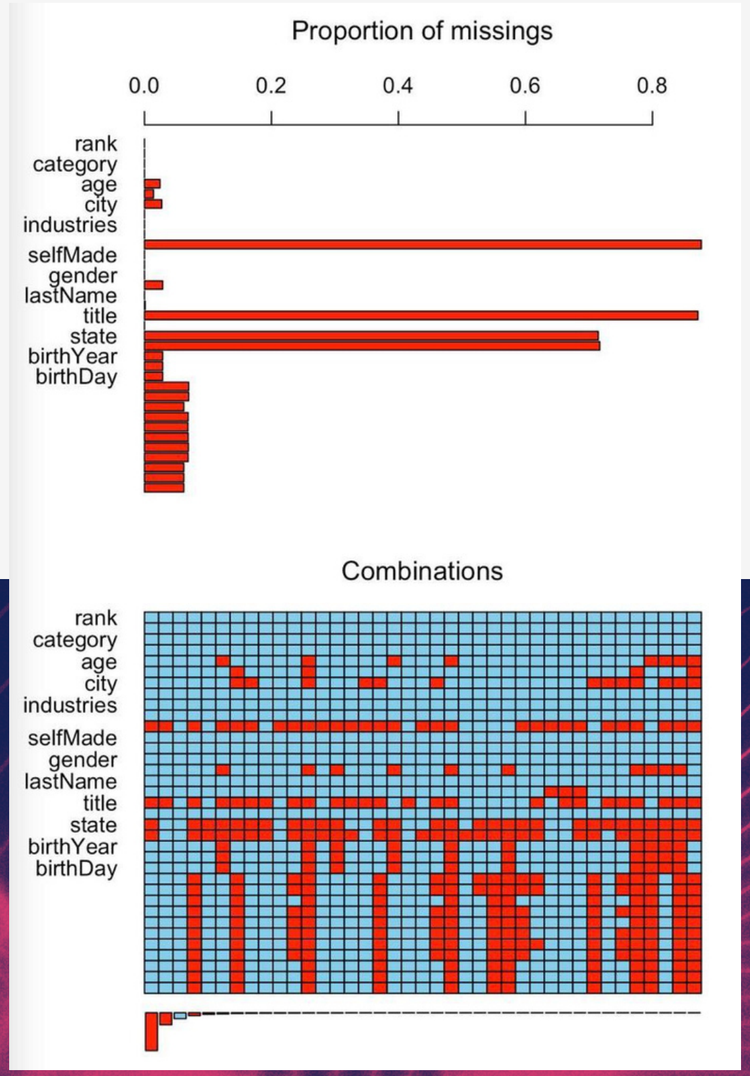
Team Cauchy Distribution took it upon ourselves to modify this data and present it in a way that tells a story to our audiences.

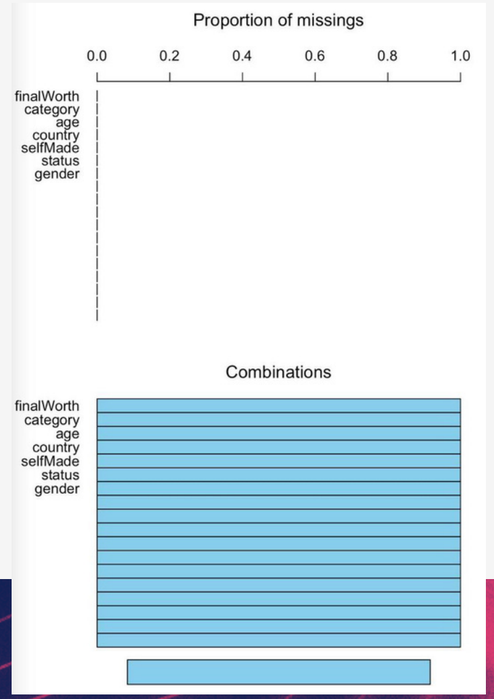
Data Cleaning

OVERVIEW

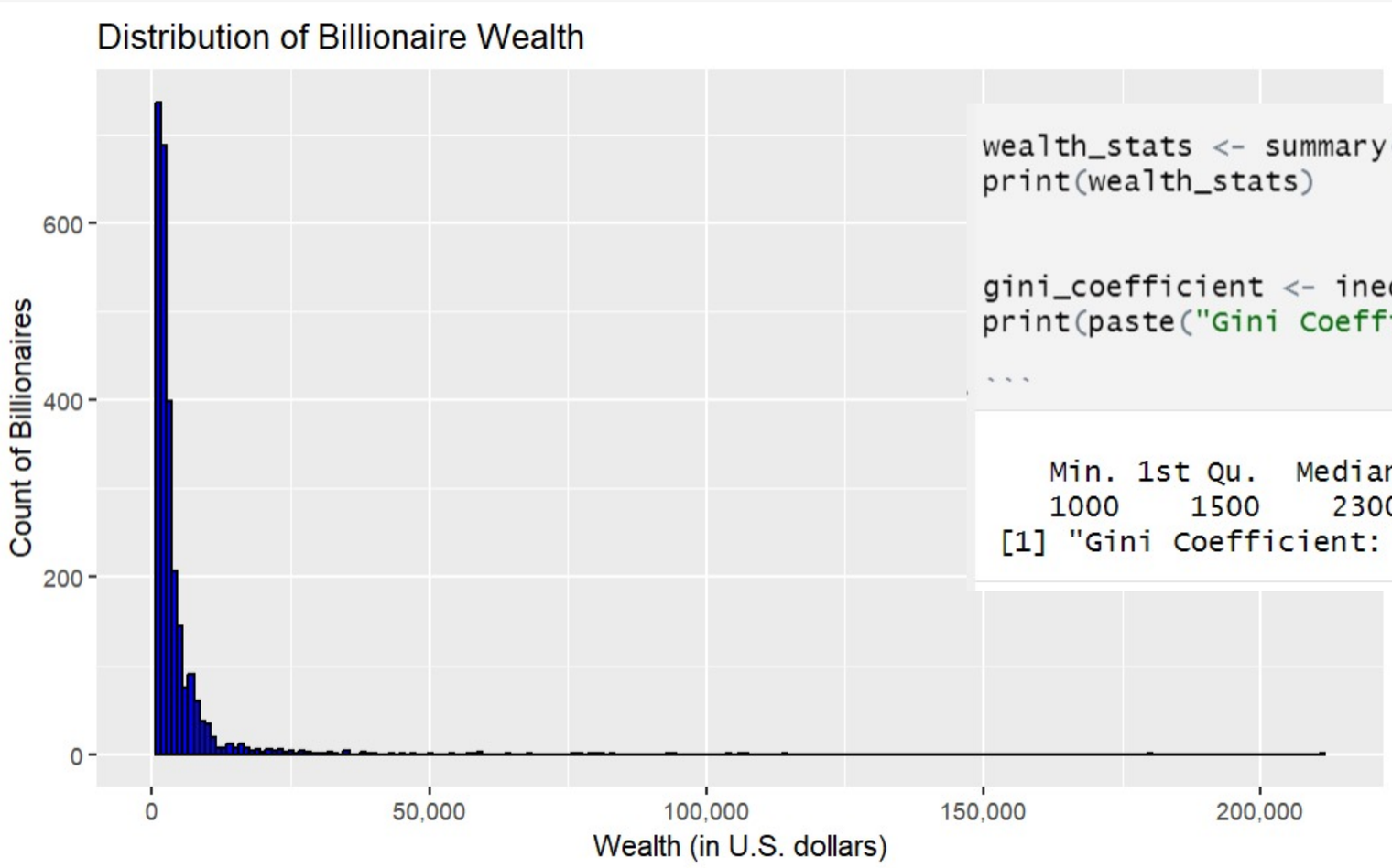
We went through numerous steps in the data cleaning process including checking for NANs, incomplete information, or impossible values.

rank	0
category	0
age	64
city	72
industries	0
organization	2315
status	0
birthDate	1245
firstName	3
date	0
residenceStateRegion	1893
birthMonth	76
cpi_country	184
gdp_country	164
gross_primary_education_enrollment_country	181
tax_revenue_country_country	183
population_country	164
longitude_country	164
finalWorth	0
personName	0
country	38
source	0
countryOfCitizenship	0
selfMade	0
gender	0
lastName	0
title	2301
state	1887
birthYear	76
birthDay	76
cpi_change_country	184
gross_tertiary_education_enrollment	182
life_expectancy_country	182
total_tax_rate_country	182
latitude_country	164
name_in_org	3





Exploratory Data

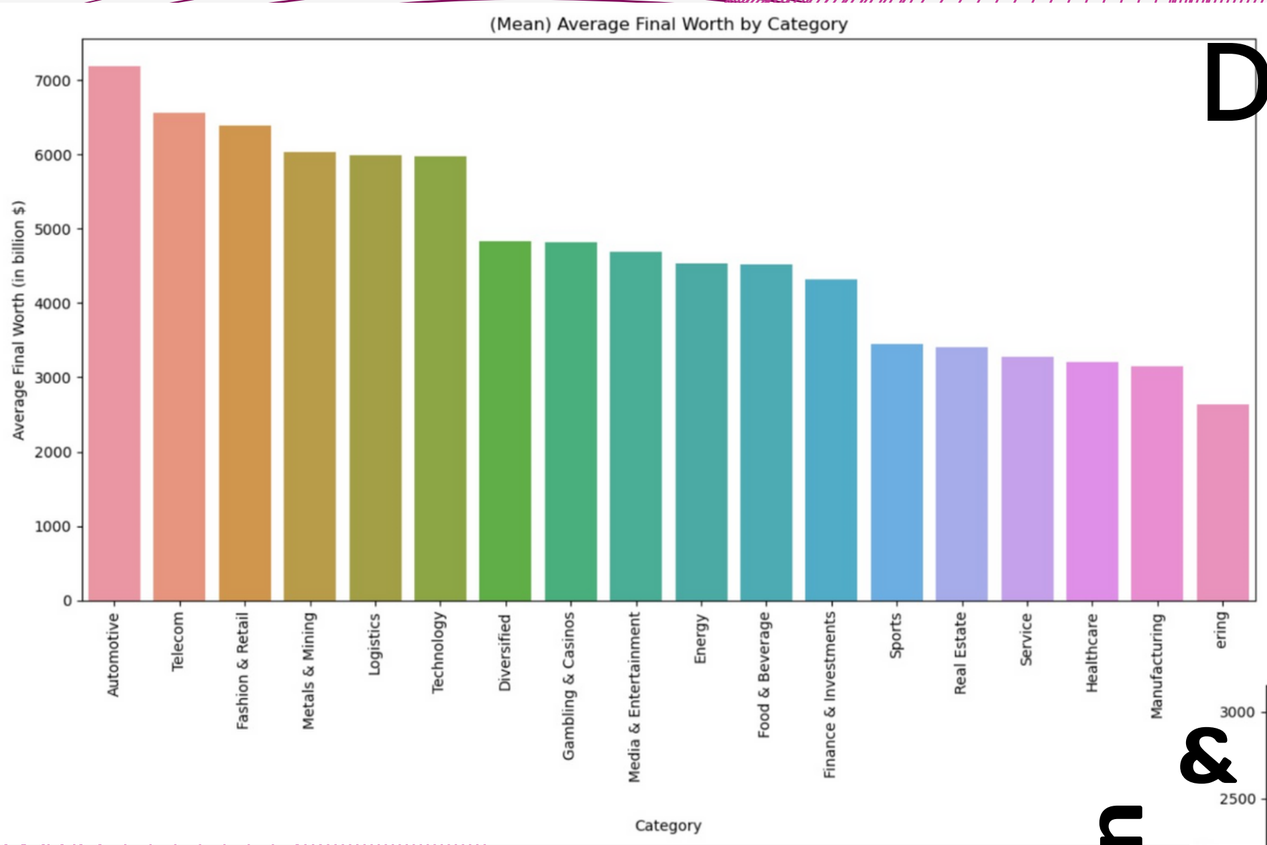


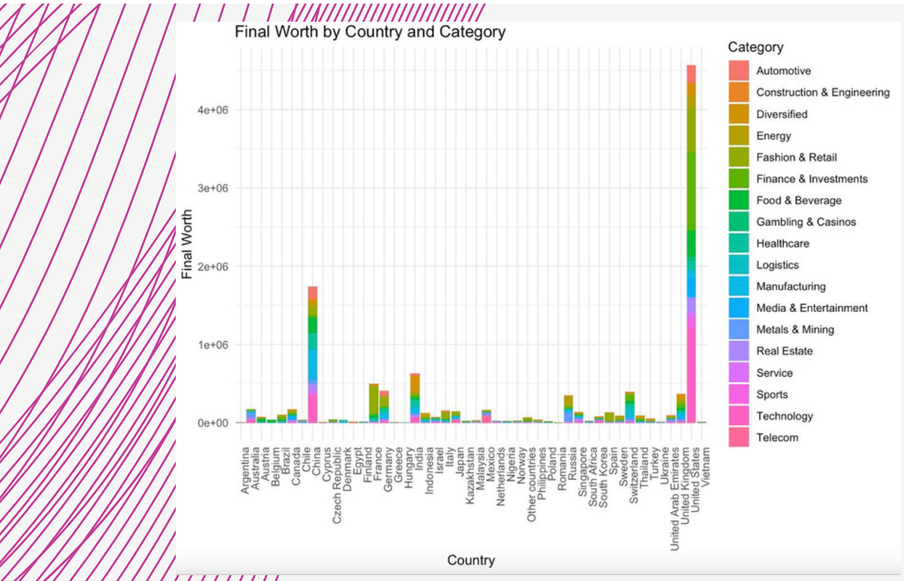

```
wealth_stats <- summary(df$finalWorth)
print(wealth_stats)


gini_coefficient <- ineq::Gini(df$finalWorth)
print(paste("Gini Coefficient:", gini_coefficient))
```

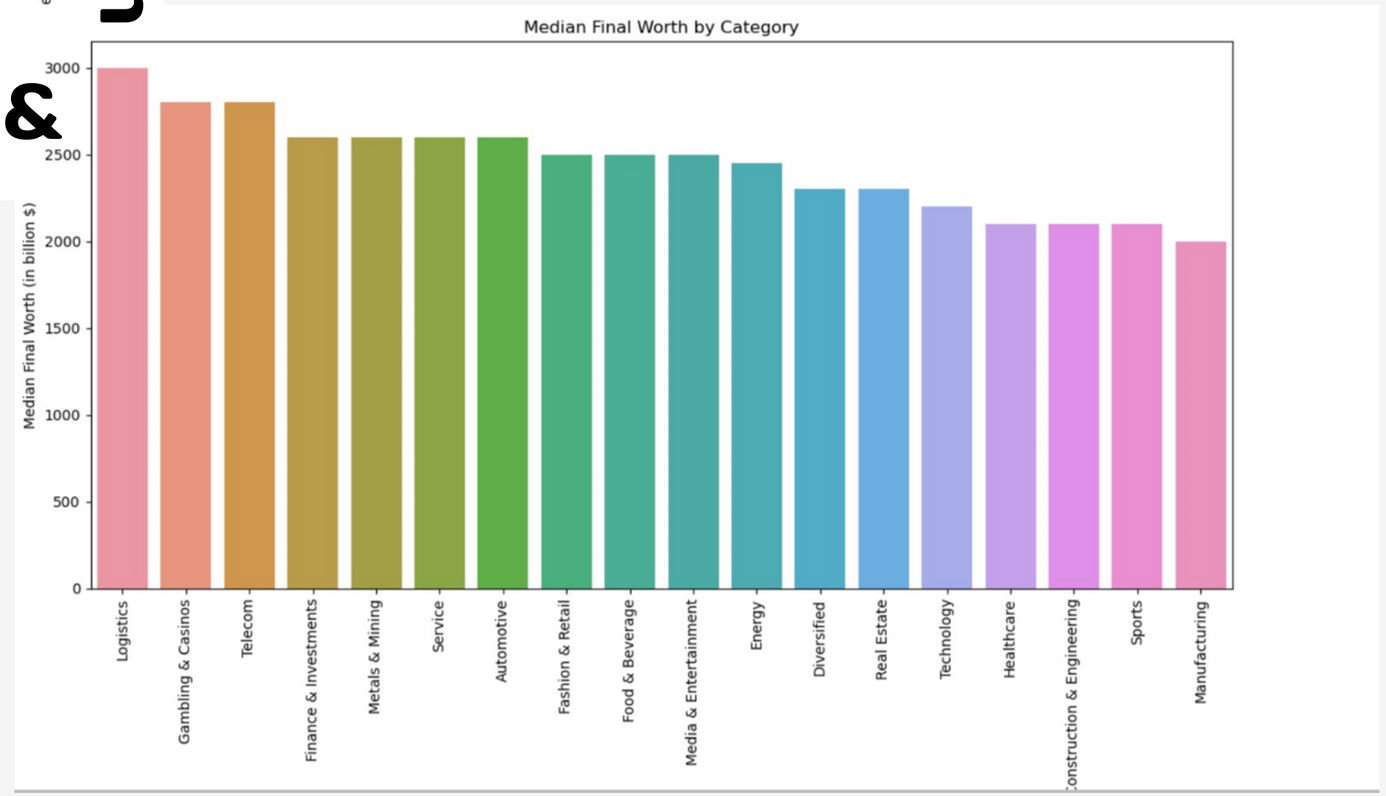
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1000    1500    2300    4624    4200  211000
[1] "Gini Coefficient: 0.549214527471524"
```

DESCRIPTIVE ANALYTICS

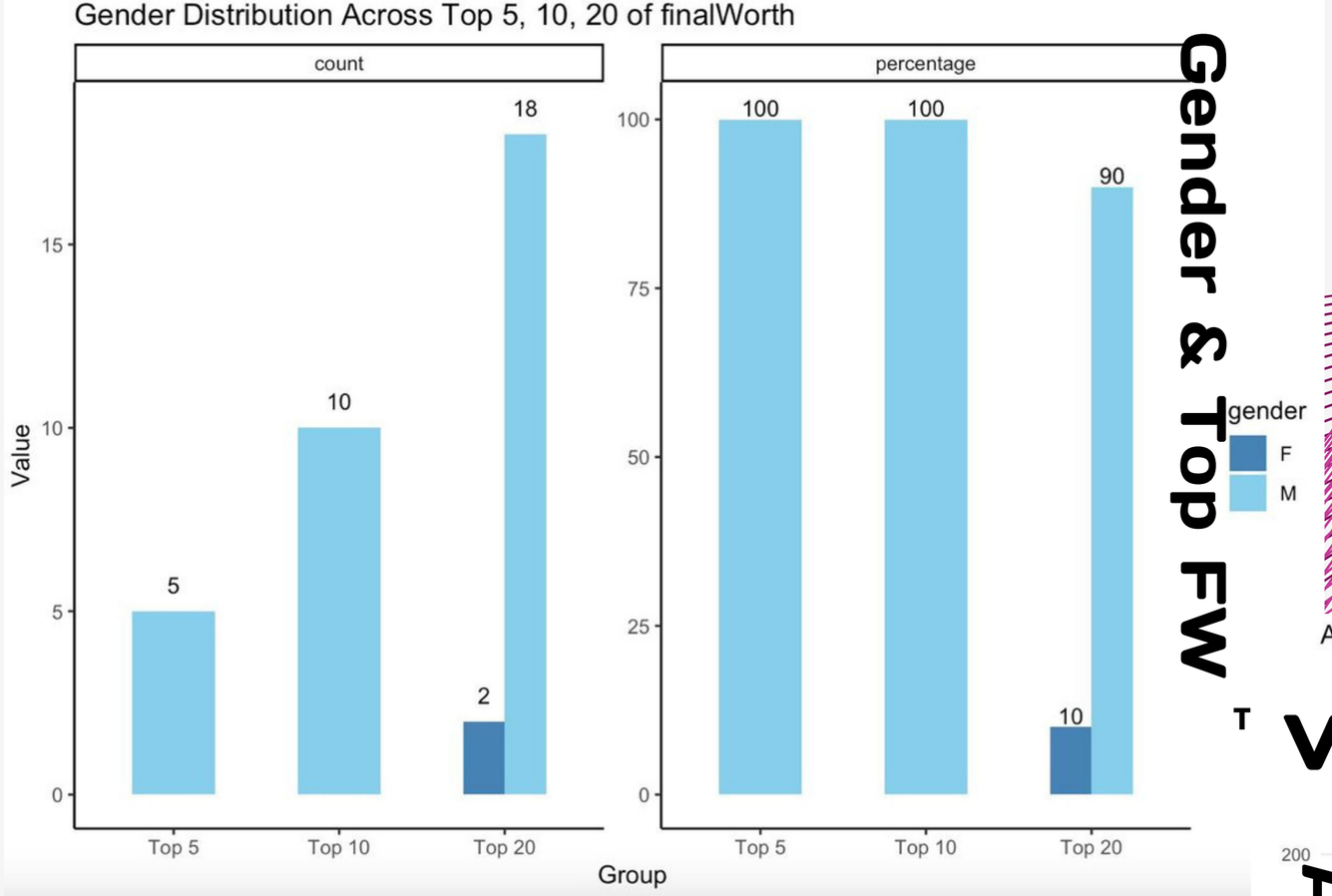
Mean & Median

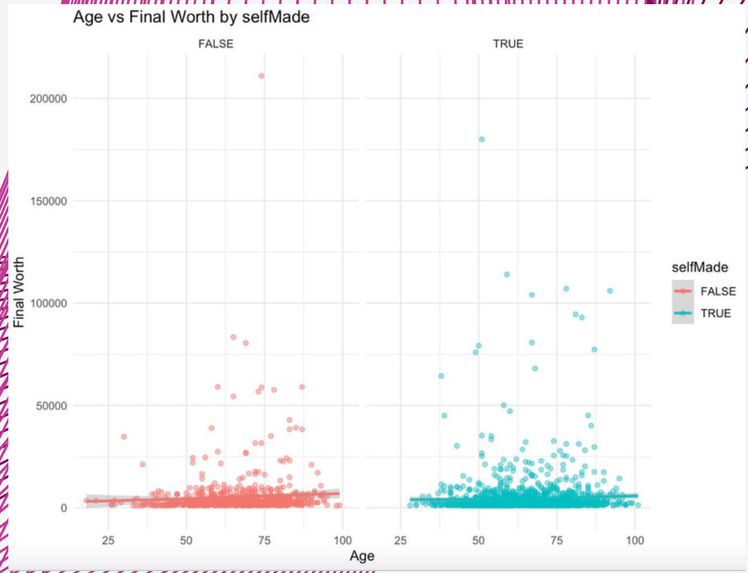


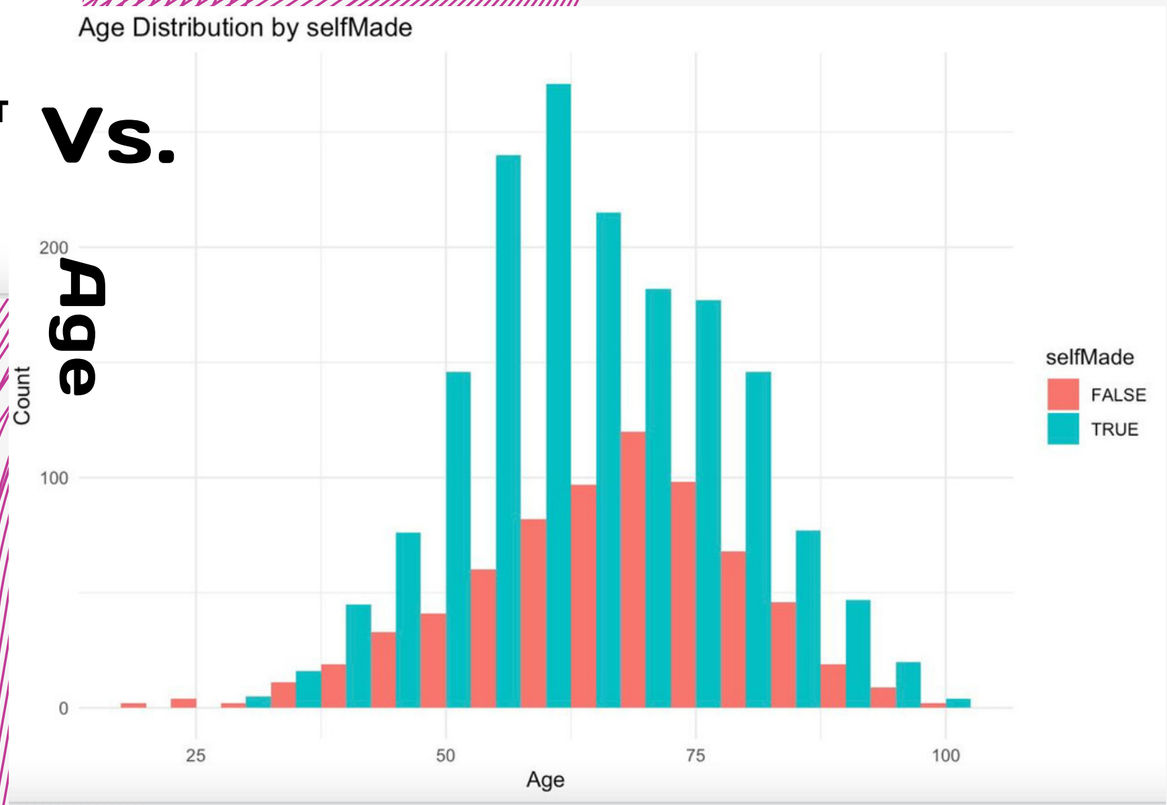




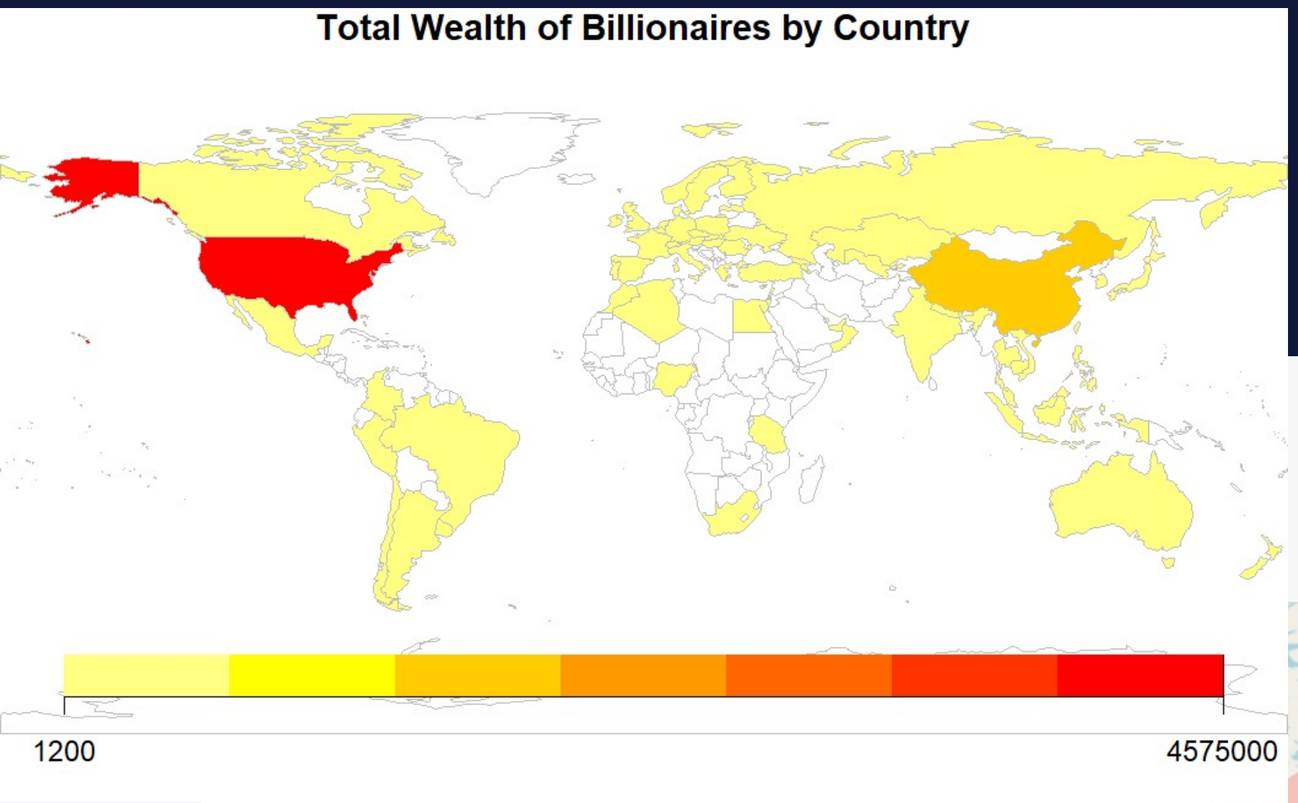
Gender & Top FW vs. Age

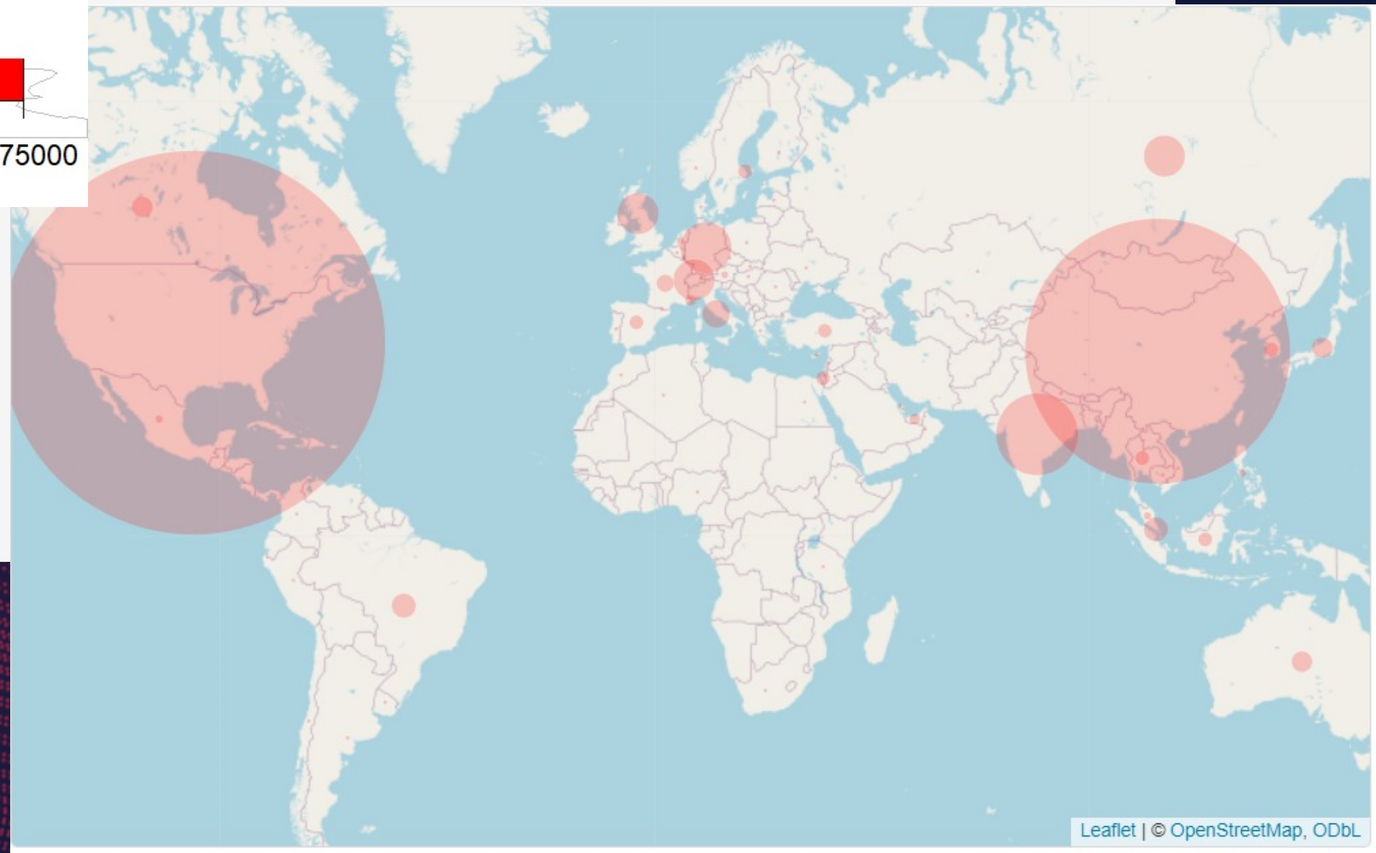




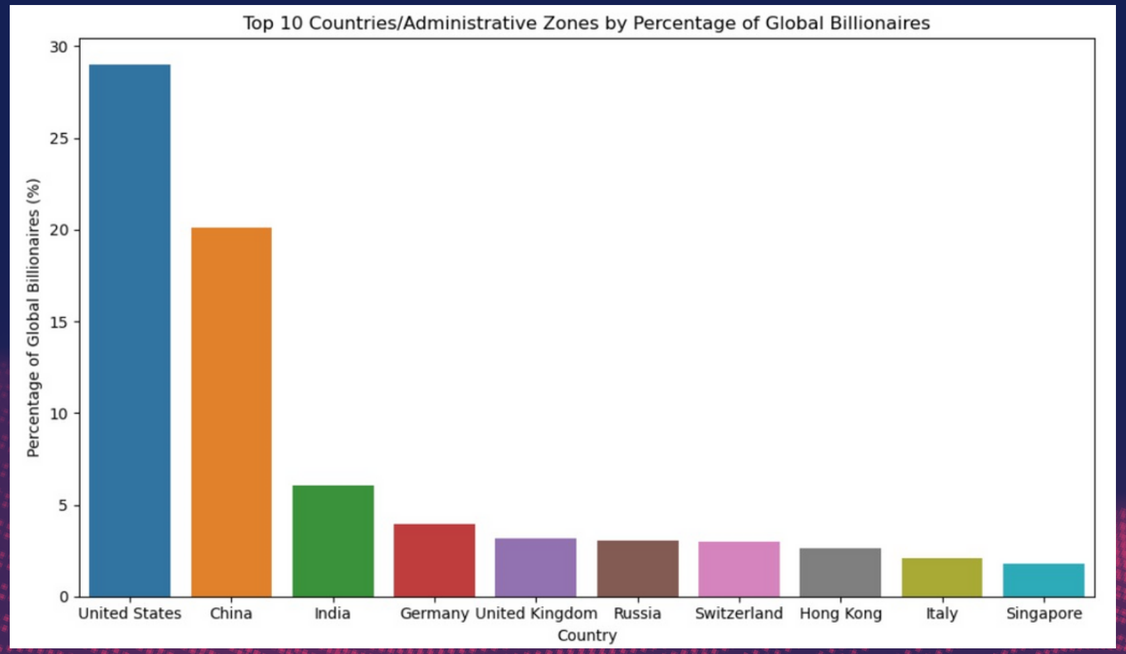


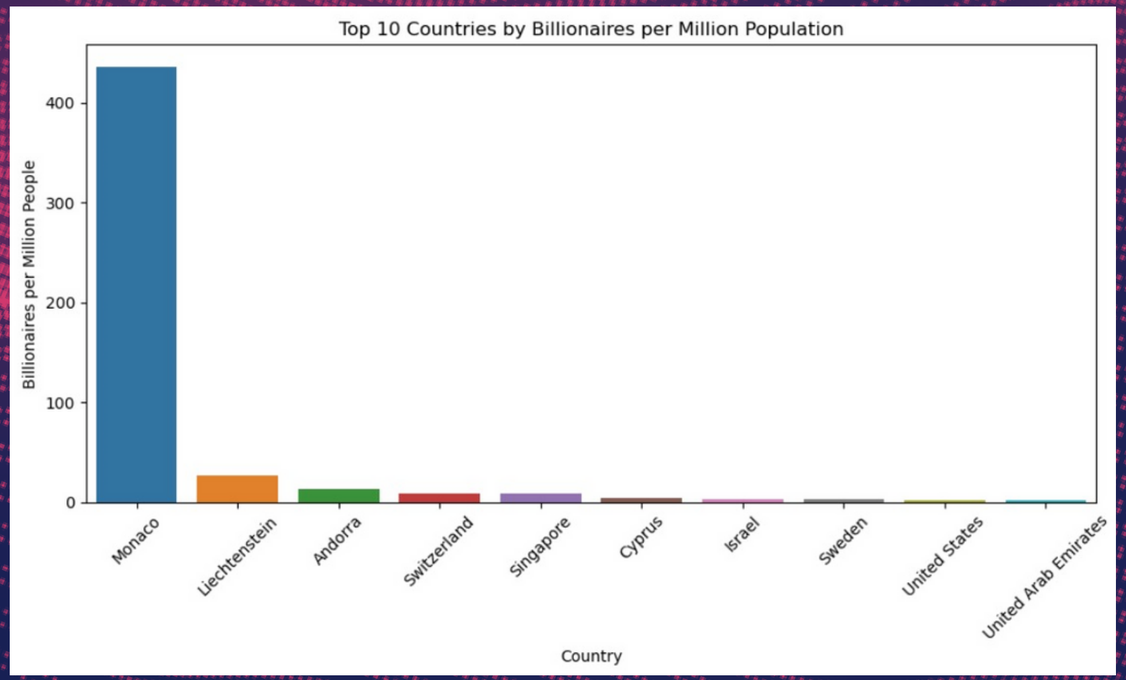
GEOGRAPHIC MAPPING

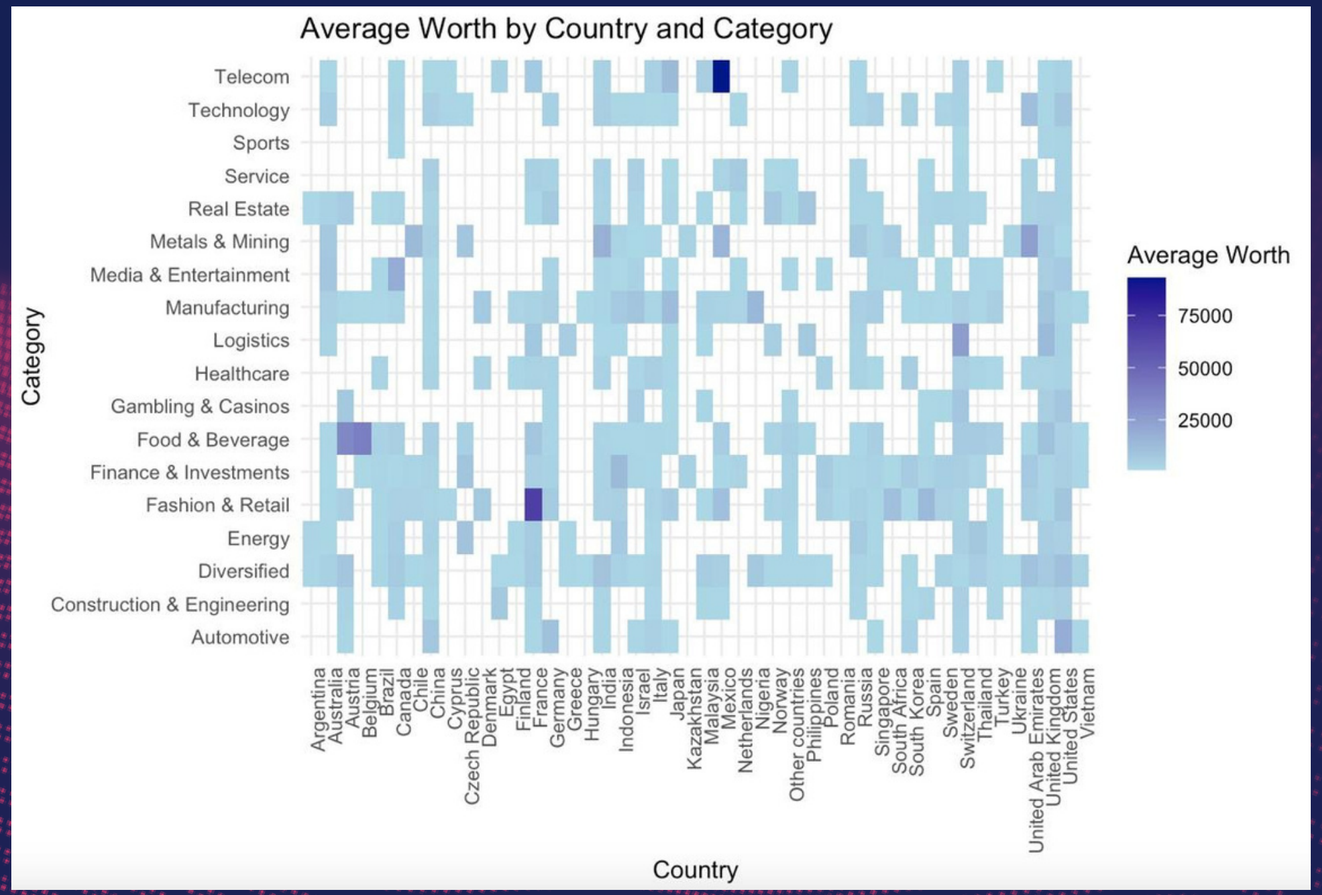




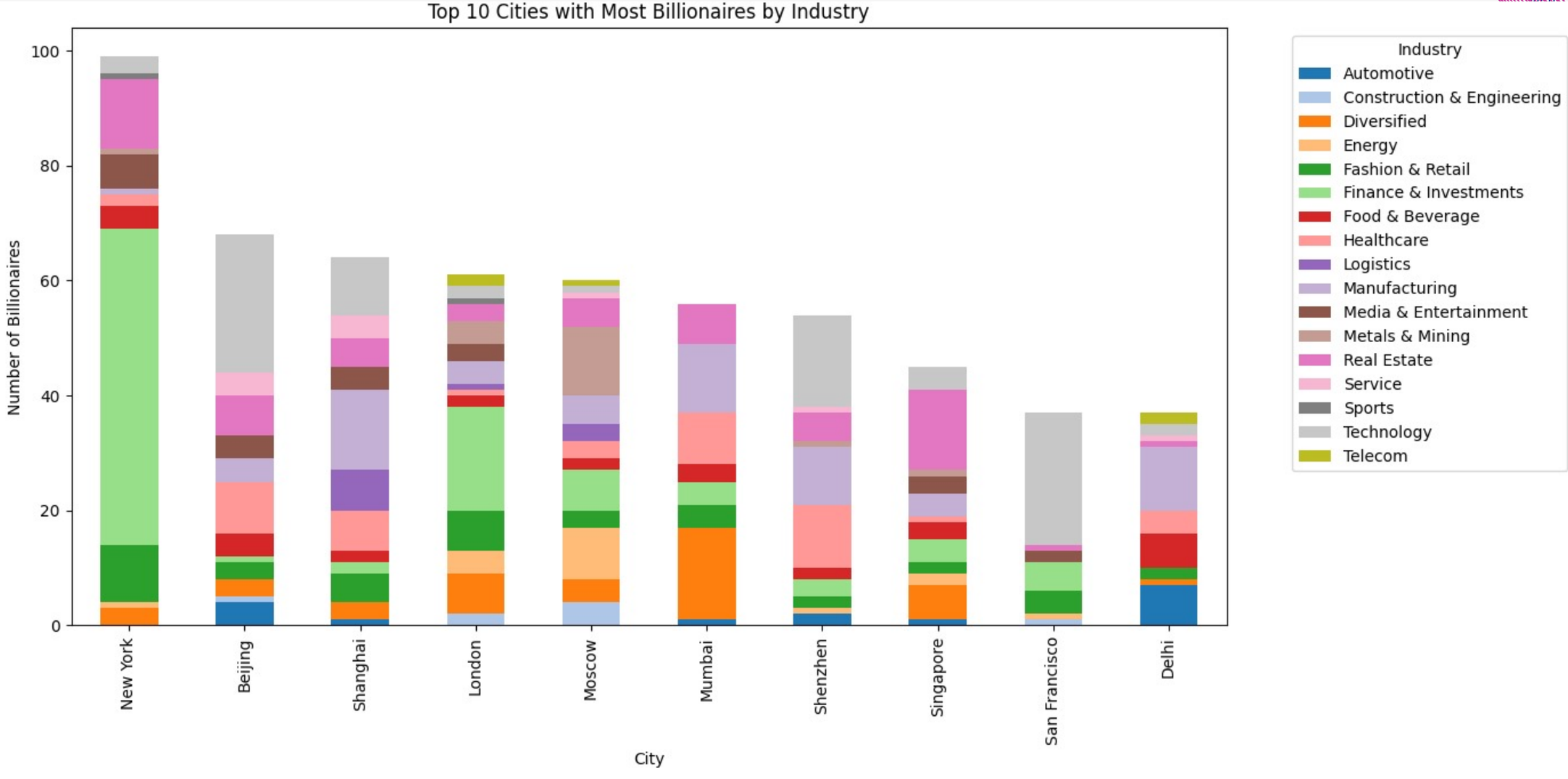
Billionaires to Countries



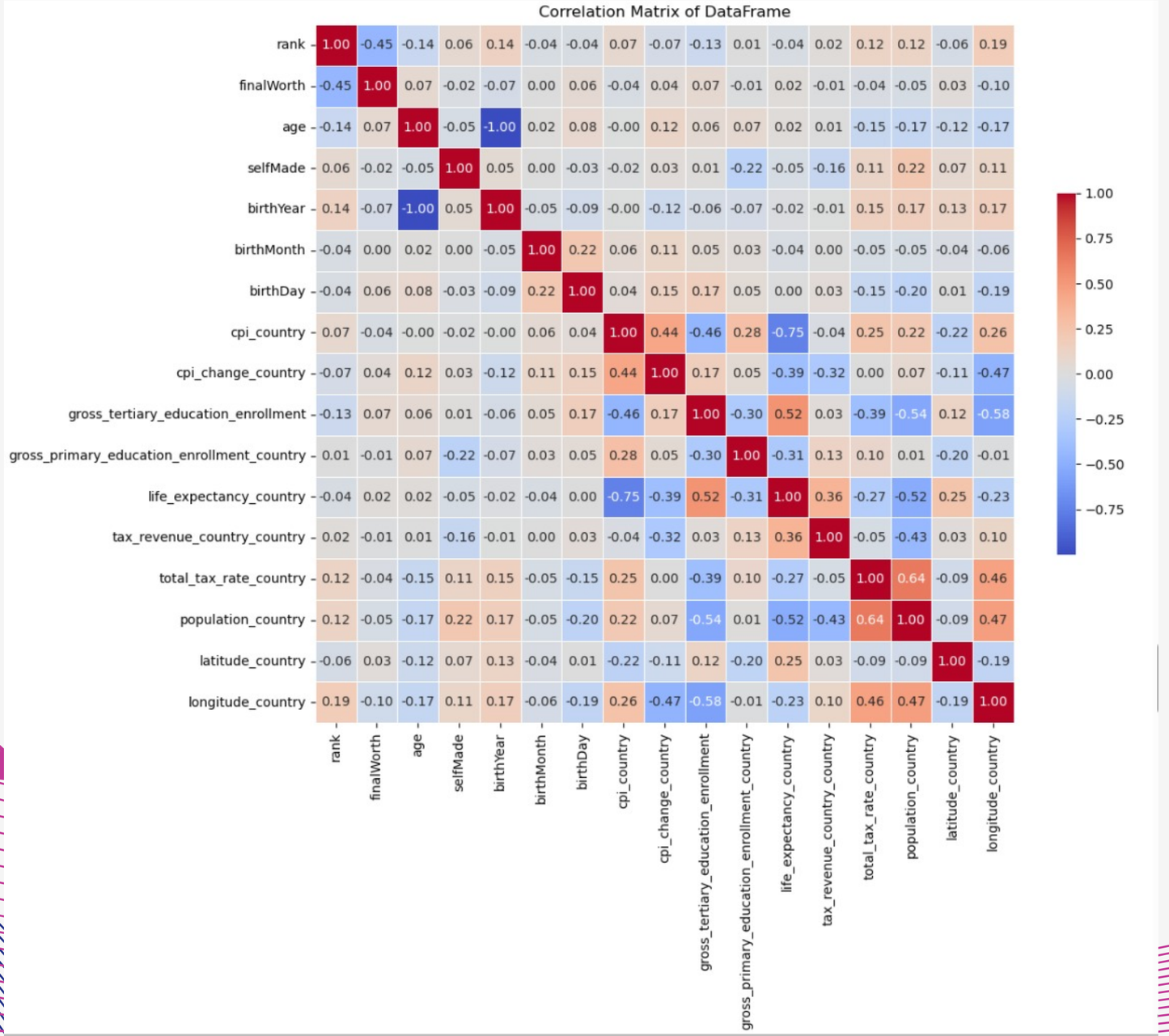




BILLIONAIRES GROUPED BY CITY AND INDUSTRY



Regression Analysis



LOGISTIC REGRESSION

$$log(\frac{P}{1-P}) = \beta_0 + \beta_1\ Category\ + \beta_2\ Gender\ + \beta_3\ cpi\_country\ + \beta_4\ cpi\_change\_country\ + \beta_5\ gdp\_country\ + \beta_6\ life\_expectancy\_country$$

Multiple interations of insignificant variables removal
Method: **backwards variable removal**
Interpretation: unit change in x leads to unit change in log odd ratio.
What about the P in the ratio?

$$Odd\ Ratio = \frac{\text{prob of the selected event}}{\text{prob of the other event}}$$

Prediction: threshold 0.5
Accuracy : 0.7602
Sensitivity : 0.3709
Specificity : 0.9260
Precision: 0.6810
Note that the selfMade = FALSE is the positive case!

```
          Reference
Prediction FALSE TRUE
     FALSE    79   37
     TRUE    134  463
```

New individual:

```
new_individual <- data.frame(
  category = "Technology",
  gender = "M",
  cpi_country = 100,
  cpi_change_country = 1.5,
  gdp_country = 50000,
  life_expectancy_country = 75
)
```

Result:

```
  1
TRUE
Levels: FALSE TRUE
```

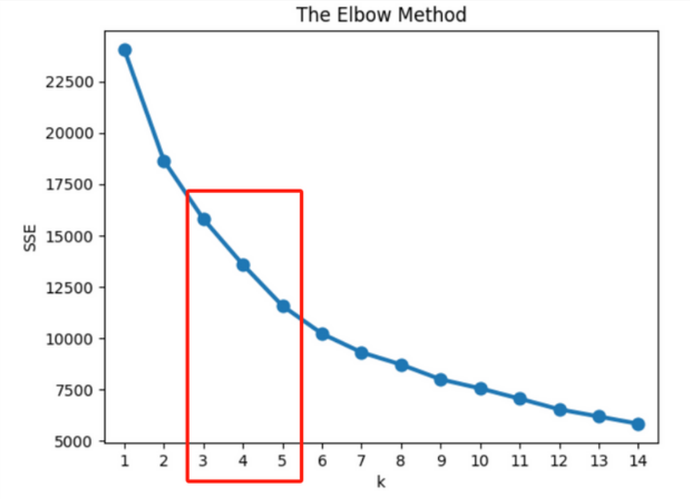
Clusters ANALYSIS

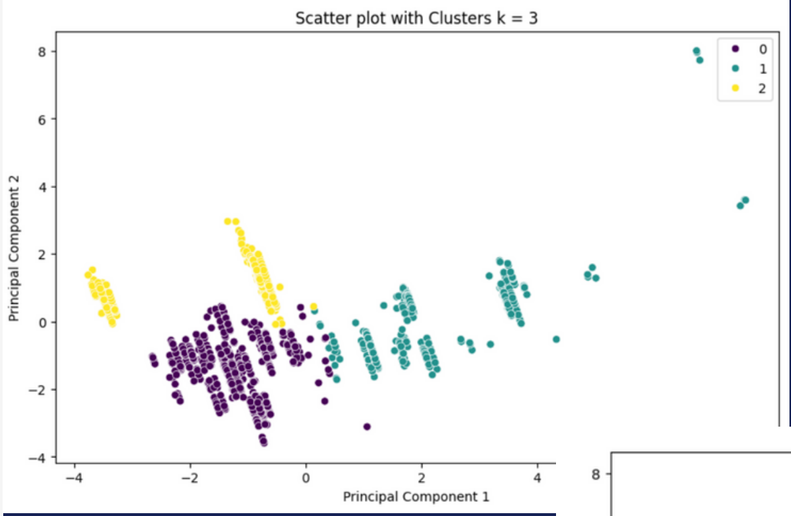
Additional data manipulation:

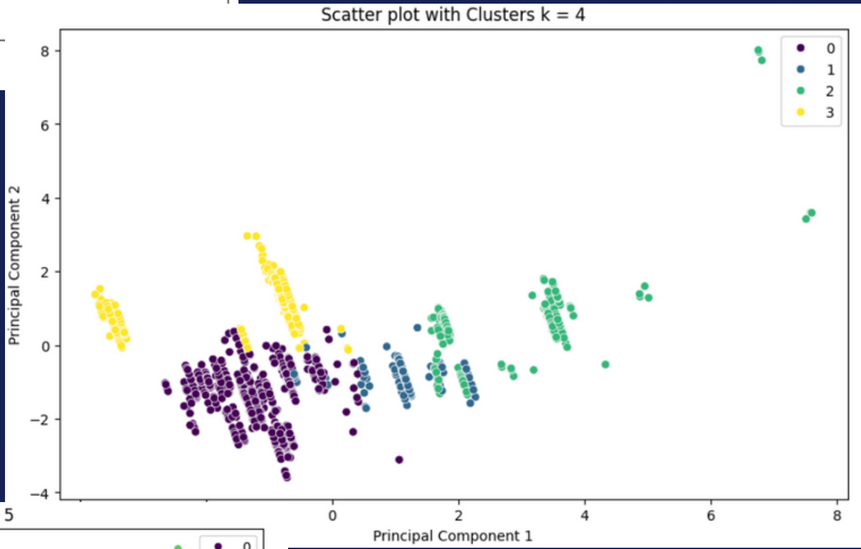
- $ and comma removed
- Additional populational info: for missing values
- Normalized: large scale (gdp and finalWorth

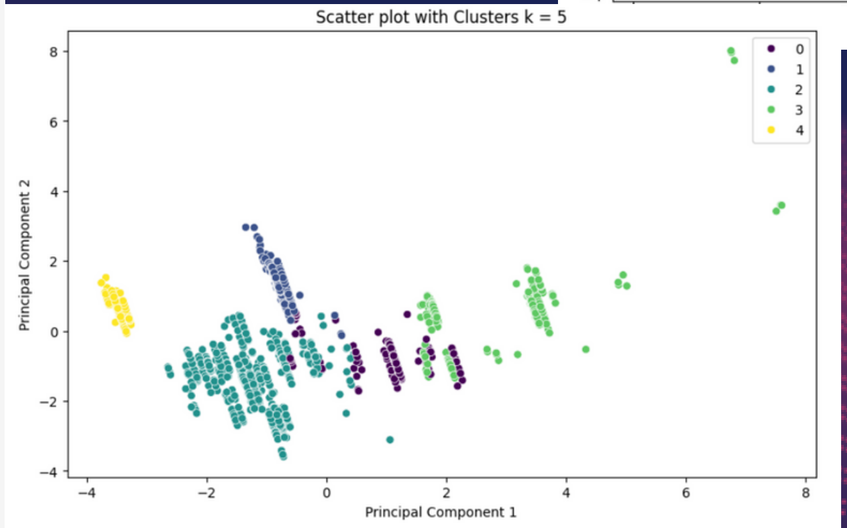
Additional metric: billionaires_per_million

- # of billionaires within per million population









CLUSTERS ANALYSIS

	finalWorth	age	selfMade	cpi_country	cpi_change_country	gross_tertiary_education_enrollment	life_expectancy_country	tax_revenue_country_country	total_tax_rate_country	billionaires_per_million	
	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	count
Cluster											
0	-0.01	0.08	-0.32	-0.52	-0.89	0.21	0.96	1.30	-0.16	-0.24	599
1	-0.10	-0.21	0.22	0.79	0.03	-0.92	-0.90	-0.32	0.85	-0.67	915
2	0.11	0.17	-0.01	-0.46	0.56	0.80	0.28	-0.54	-0.76	0.84	893

CLUSTER 0
CLASSICAL HEIRS

BASIC: AVERAGE, MID-AGE, INHERITED
COUNTRIES: LOW CPI AND KEEP DECREASING; DECENT EDUCATION; HIGH LIFE EXPECTANCY, HIGH TAX RATE
OPPORTUNITY: NORMAL

CLUSTER 1
GRASSROOTS PIONEERS

BASIC: NOT THAT WEALTHY, YOUNG, START-UP:
COUNTRIES: HIGH CPI AND KEEP AT THE CONSTANT; LOW EDUCATION; LOW LIFE EXPECTANCY, RELATIVELY HIGH TAX RATE;
OPPORTUNITY: LOW

CLUSTER 2
RAPID GROWING STARS

BASIC: WEALTHY, OLD, SELFMADE-BALANCE
COUNTRIES: LOW CPI BUT FAST INCREASE; HIGH EDUCATION; DECENT LIFE EXPECTANCY, LOW TAX RATE
OPPORTUNITY: RELATIVELY HIGH

RANDOM FOREST

```
> print(model)
Random Forest

1667 samples
  17 predictor
   2 classes: 'FALSE', 'TRUE'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1334, 1333, 1334, 1334, 1333
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   1    0.7000593  0.0000000
   2    0.7354426  0.2247879
   3    0.7486486  0.3156814
   4    0.7618475  0.3731837
   5    0.7840499  0.4469434
   6    0.7900451  0.4696026
   7    0.7924493  0.4713331
   8    0.7894517  0.4640817
   9    0.7912517  0.4726915
  10    0.7930517  0.4711066
  11    0.7900487  0.4642492
  12    0.7912427  0.4701162
  13    0.7924385  0.4736841
  14    0.7960421  0.4836252
  15    0.7960421  0.4855587
  16    0.7942421  0.4812569
  17    0.7918433  0.4718703

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 14.
```

MTRY choice: 14
With the highest Accuracy

	Reference	
Prediction	FALSE	TRUE
FALSE	97	32
TRUE	116	468

Accuracy : 0.7924
Sensitivity : 0.4554
Specificity : 0.9360
Precision: 0.7519

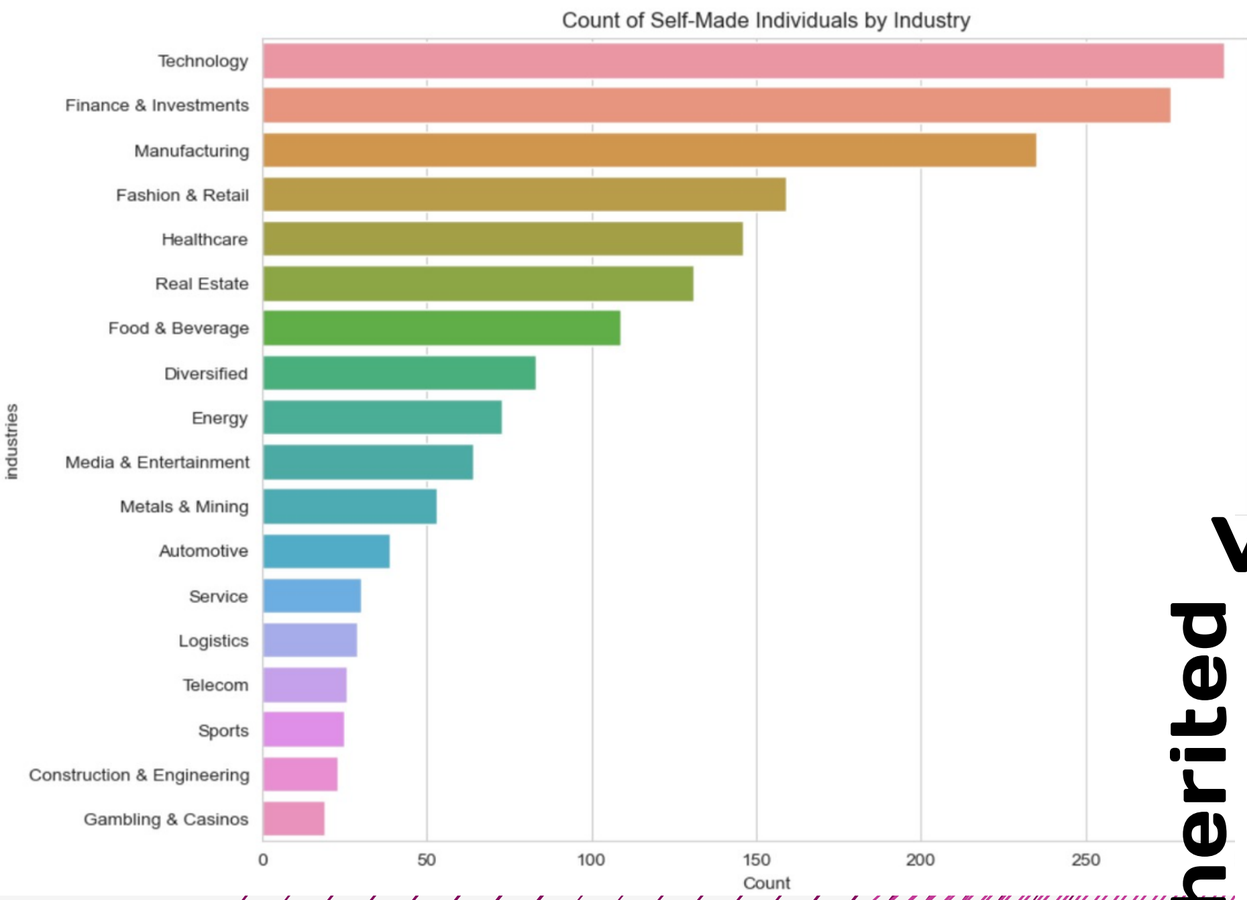
Note that the selfMade = FALSE is the positive case!

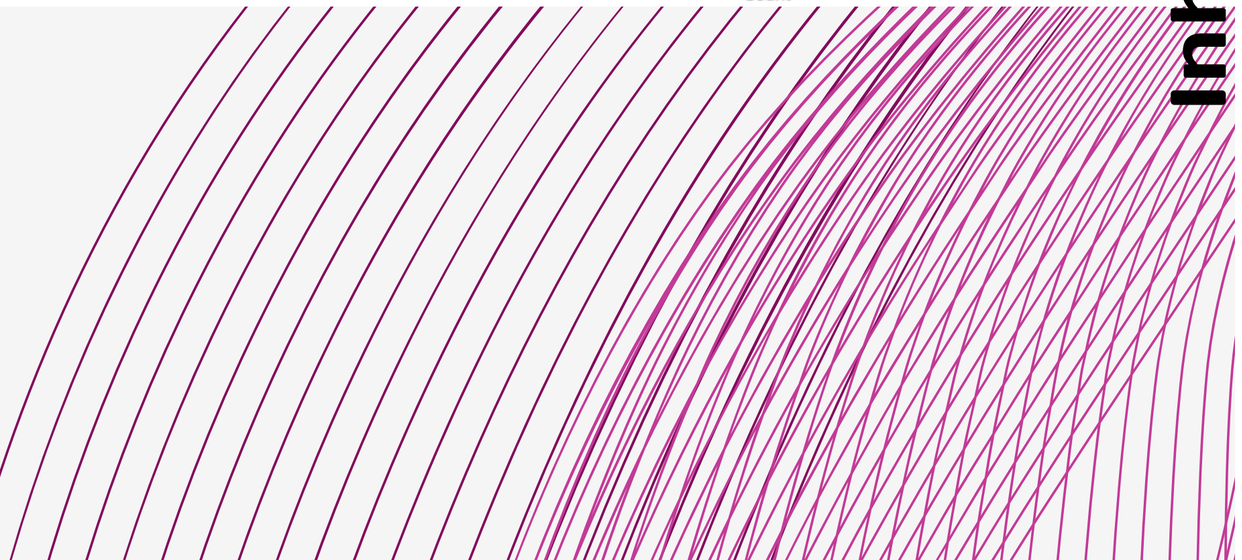
Cauchy Distribution's

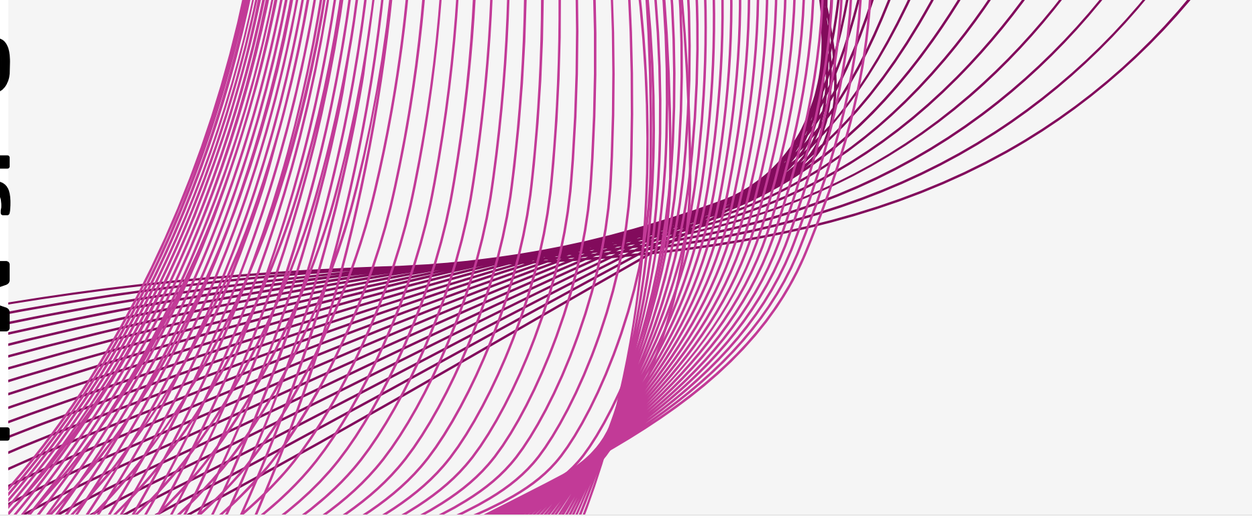
FORTUNE CHANGER APP

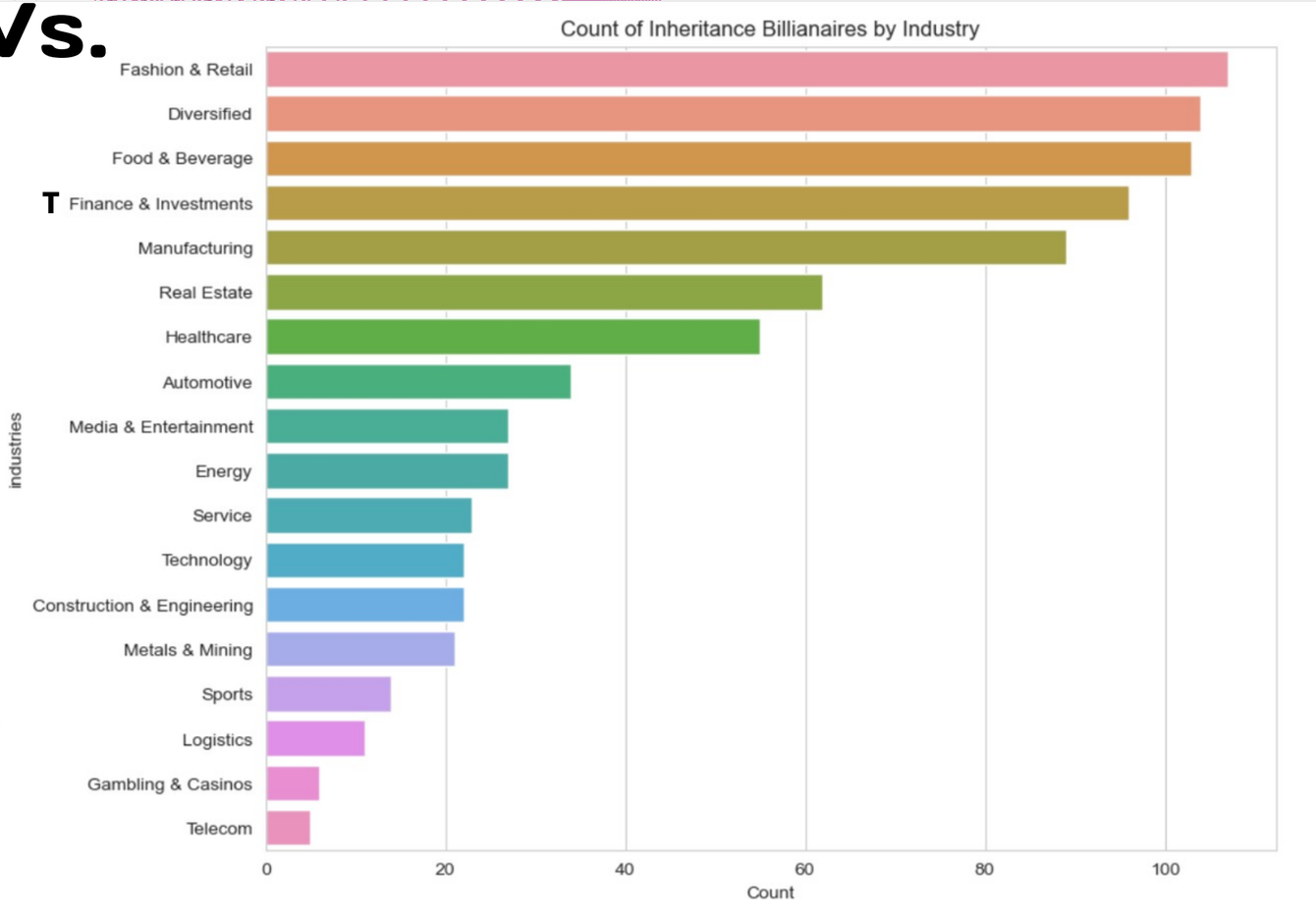
```
WELCOME TO CUACHY DISTRIBUTION fORTUNE CHANGER APP!
What is your name? Alex
In which country do you reside? India
What city do you live in? delhi
Please select an industry by entering the corresponding number:
1: Automotive
2: Technology
3: Finance & Investments
4: Media & Entertainment
5: Telecom
6: Fashion & Retail
7: Food & Beverage
8: Diversified
9: Logistics
10: Manufacturing
11: Real Estate
12: Energy
13: Metals & Mining
14: Service
15: Healthcare
16: Gambling & Casinos
17: Construction & Engineering
18: Sports
Your choice:  17
Are your parents billionaires? (yes/no) no
Based on your input alex, if you move from India to the United States, the cha
nces of you becoming a billionaire will increase by 19.99 times. Switching to
Technology would it up by 12.695652173913043 times. If you move to the United
States and switch to Technology, you'd improve your chances of being a billion
aire by 253.82 times
```

Self-Made Vs. Inherited









Thank you!

Backup slides

Logistic regression

```r
#Logistic regression
set.seed(699)

# create the model
glm_model <- glm(selfMade ~ category+gender+cpi_country+cpi_change_country+
                   gdp_country+life_expectancy_country, data = training_set, family = binomial)

# summary
summary(glm_model)

# predictions
glm_predictions <- predict(glm_model, newdata = testing_set, type = "response")
glm_predictions <- factor(ifelse(glm_predictions > 0.5, "TRUE", "FALSE"),levels=c("FALSE","TRUE"))

# confusion matrix
confusionMatrix(data = glm_predictions, reference = testing_set$selfMade)
```

Random Forest

K-nn

```
#remove $ sign and comma in a column
df['gdp_country'] = df['gdp_country'].str.replace('$', '').str.replace(',', '') # remove $ and , by "nothing"
df['gdp_country'] = df['gdp_country'].astype(float) # change it to float
```

```
<ipython-input-11-244f68849c2b>:2: FutureWarning: The default value of regex will change from True to False in a future
  df['gdp_country'] = df['gdp_country'].str.replace('$', '').str.replace(',', '') # remove $ and , by "nothing"
```

```
[12] # Create a dictionary from the counts DataFrame for easy mapping
counts_dict = counts.set_index('country')['population_country'].to_dict()
counts_dict1 = counts.set_index('country')['billionaires_per_million'].to_dict()
# Fill missing values in 'population' column using the counts dictionary
df['population_country'] = df['country'].map(counts_dict)
df['billionaires_per_million'] = df['country'].map(counts_dict1)
df.head()
```

```
[19] # calling the value of df4 without rows with NA

df3 = df3.dropna()


# normalization
df3_standard = (df3 - df3.mean()) / df3.std()
df3_standard
```

```
sse = {}
for k in range(1, 15):    #Note -- we can adjust this range!
# Initialize KMeans with k clusters
    kmeans = KMeans(n_clusters=k, random_state=654)
# Fit KMeans on the normalized dataset
    kmeans.fit(df3_standard)
    sse[k] = kmeans.inertia_
# Add the plot title "The Elbow Method"
plt.title('The Elbow Method')
# Add X-axis label "k"
plt.xlabel('k')
# Add Y-axis label "SSE"
plt.ylabel('SSE')
sns.pointplot(x=list(sse.keys()), y=list(sse.values()));
```

K-nn

```
kmeans = KMeans(n_clusters=3, random_state=654)
kmeans.fit(df3_standard)
cluster_labels = kmeans.labels_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py
  warnings.warn(
```

```
[29] kmeans6 = df3_standard.assign(Cluster = cluster_labels)
kmeans6.groupby(['Cluster']).agg({
'finalWorth': 'mean',
'age': 'mean',
'selfMade': 'mean',
'cpi_country': 'mean',
'cpi_change_country': 'mean',
'gross_tertiary_education_enrollment': 'mean',
'life_expectancy_country': 'mean',
'tax_revenue_country_country': 'mean',
'total_tax_rate_country': 'mean',
'billionaires_per_million': ['mean', 'count'],
}).round(2)
```

```
[30] # Step 2: Reduce data to 2D using PCA for visualization
pca = PCA(n_components=2)
principal_components = pca.fit_transform(df3_standard)

# Step 3: Plotting
plt.figure(figsize=(10, 6))
sns.scatterplot(x=principal_components[:, 0], y=principal_components[:, 1], hue=cluster_labels, palette='viridis')
plt.title('Scatter plot with Clusters k = 3')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```