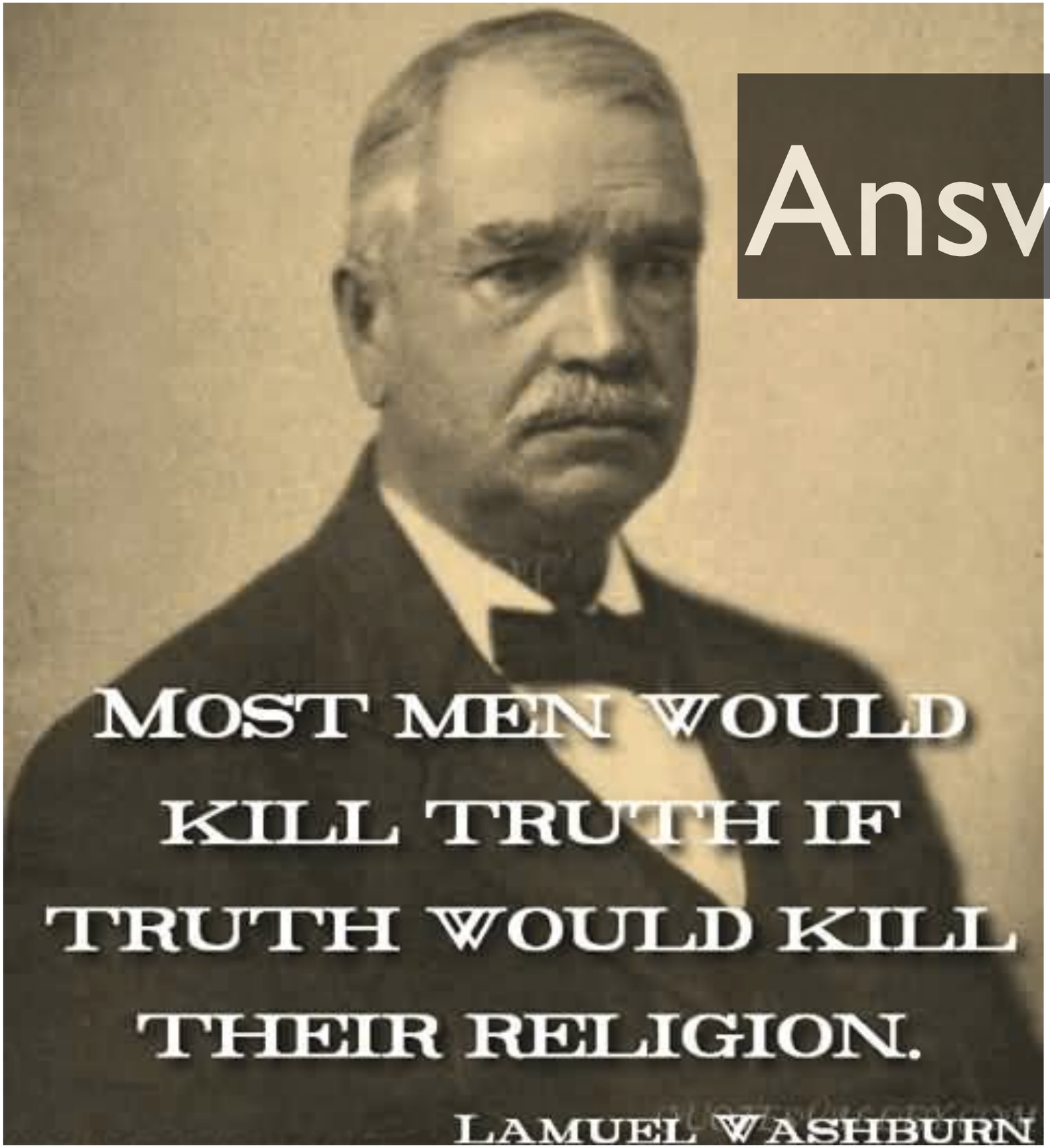
Mike Schoeffler

GA Data Science: DAT-NYC-30

Answerable?



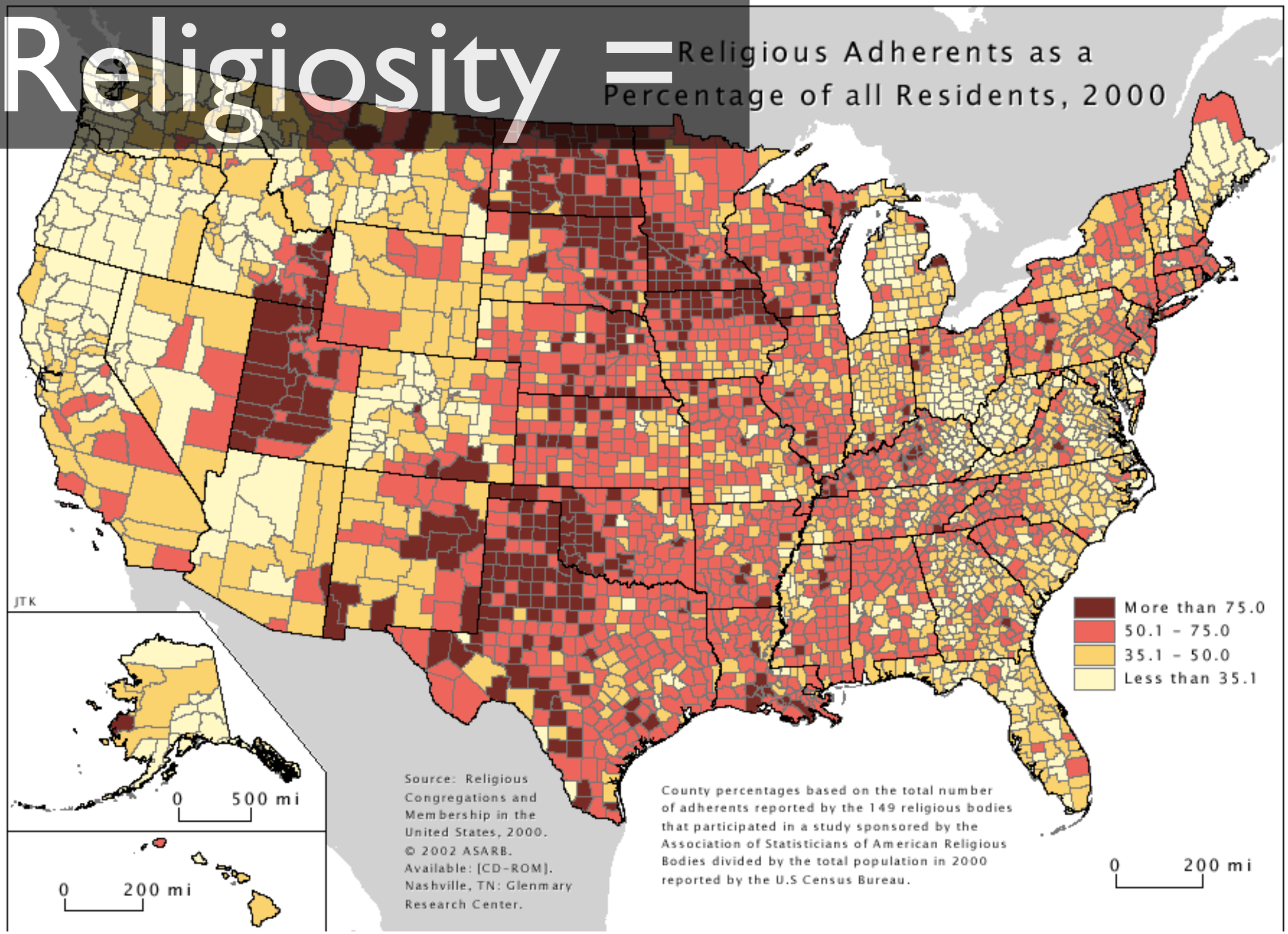
Mike Schoeffler GA Data Science: DAT-NYC-30

P2P historical data +

Mike Schoeffler

GA Data Science: DAT-NYC-30

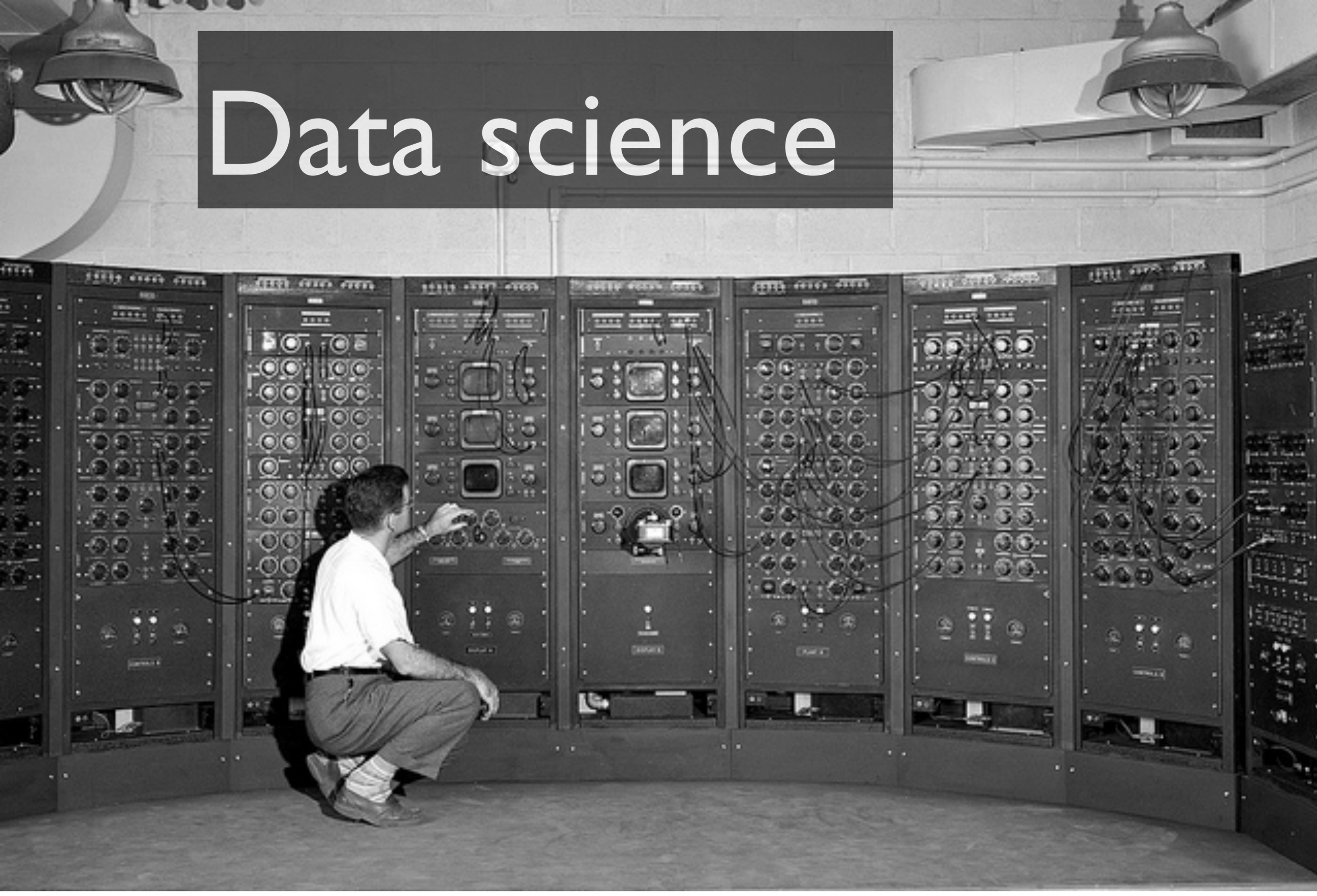
Religiosity =



Mike Schoeffler

GA Data Science: DAT-NYC-30

Data science



Mike Schoeffler

GA Data Science: DAT-NYC-30

Data prep

```
# map zip codes to county codes
zip_counties = pd.read_excel('ZIP_COUNTY_122015.xlsx', skiprows = 0, header = 0)
zip_counties['COUNTY'] = zip_counties['COUNTY'].astype(str).str.zfill(5)
zip_counties['ZIP'] = zip_counties['ZIP'].astype(str).str.zfill(5)
zip_counties.sample()
```

	ZIP	COUNTY	RES_RATIO	BUS_RATIO	OTH_RATIO	TOT_
39293	71456	22069	1	1	1	1

Leading zeros

	State	State_Code	County_Code	County	H1	County_Code_Full	County_Full
0	AL	01	001	Autauga County	H1	01001	Autauga, AL
1	AL	01	00	[illegible]		01003	Baldwin, AL

String manipulation

```
zip_to_adherents = pd.merge(religion, national_county, how='left', left_on='County or Equivalent', right_on='County_Full')
zip_to_adherents = pd.merge(zip_to_adherents, zip_coun[illegible] how='right', left_on='County_Code_Full', right_on='COUNTY')
zip_to_adherents.head()
```

Left join

	County or Equivalent	Population	PopRank	Adherents	AdhRank	Congregations	ConRank	Adherents %	Adh% Rank	Congregations Per 10K People	Con Per 10K Rank	State	State_Code
0	Autauga, AL	54571	917	36938	691	106	831	67.7	546	19	1799	AL	01
1	Autauga, AL	54571	917	36938	691	106	831	67.7	546	19	1799	AL	01
2	Autauga,	54571	917	36938	691	106	831	67.7	546	19	1799	AL	01

Map y

```
# map function failed for unknown reason
past_loans['default'] = np.where(past_loans['loan_status']=='Fully Paid', 0,
    np.where(past_loans['loan_status']=='Charged Off', 1,
    np.where(past_loans['loan_status']=='Current', 0.05,
    np.where(past_loans['loan_status']=='Late (31-120 days)', 0.76,
    np.where(past_loans['loan_status']=='In Grace Period', 0.28,
    np.where(past_loans['loan_status']=='Late (16-30 days)', 0.59,
    np.where(past_loans['loan_status']=='Default', 0.9,
    np.where(past_loans['loan_status']=='Does not meet the credit policy. Status:Fully Paid', 0,
    np.where(past_loans['loan_status']=='Does not meet the credit policy. Status:Charged Off', 1, 999)))))))))
```

ıome_ownership	annual_inc	verification_status	issue_d	loan_status	desc	purpose	title	zip_code	addr_state	dti	delinq_2yrs
RENT	24000	Verified	2011-12-01	Fully Paid	Borrower added on 12/22/11 > I need to upgra...	credit_card	Computer	86001	AZ	27.65	0
RENT	30000	Source Verified	2011-12-01	Charged Off	Borrower added on 12/22/11 > I plan	car	bike	30901	GA	1.00	0

106 Columns

870,000 Rows

Examine field types and amount of data

Is there any missing data? Unfortunately the null counts disappear with too much data.

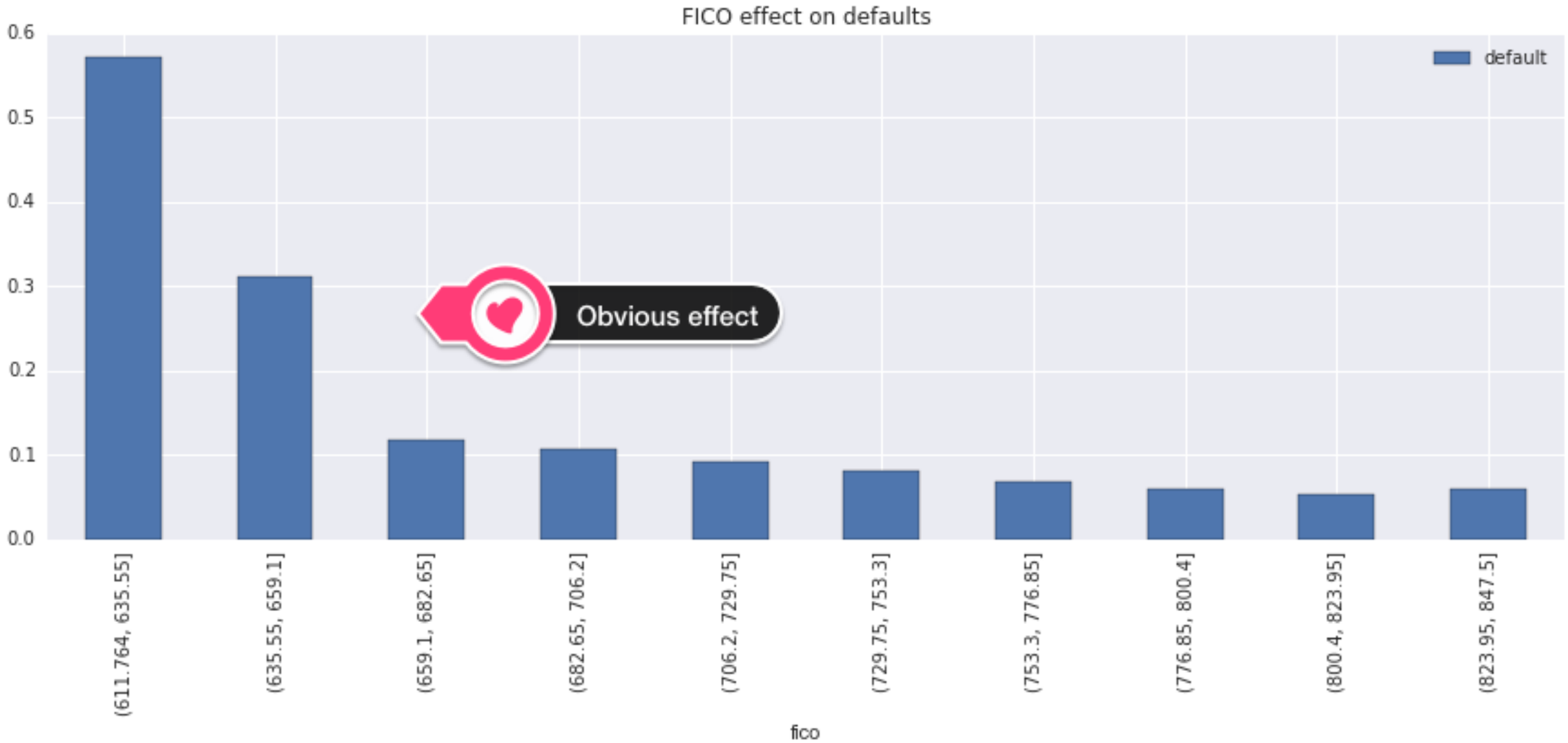
```
: past_loans.info()
<class 'pandas.core.frame.DataFrame'>
Index: 887383 entries, 1077501 to 36271262
Columns: 104 entries, member_id to fico
dtypes: category(7), datetime64[ns](4), float64(84), object(9)
memory usage: 669.4+ MB
```

Condensed info

Summary

Mike Schoeffler GA Data Science: DAT-NYC-30

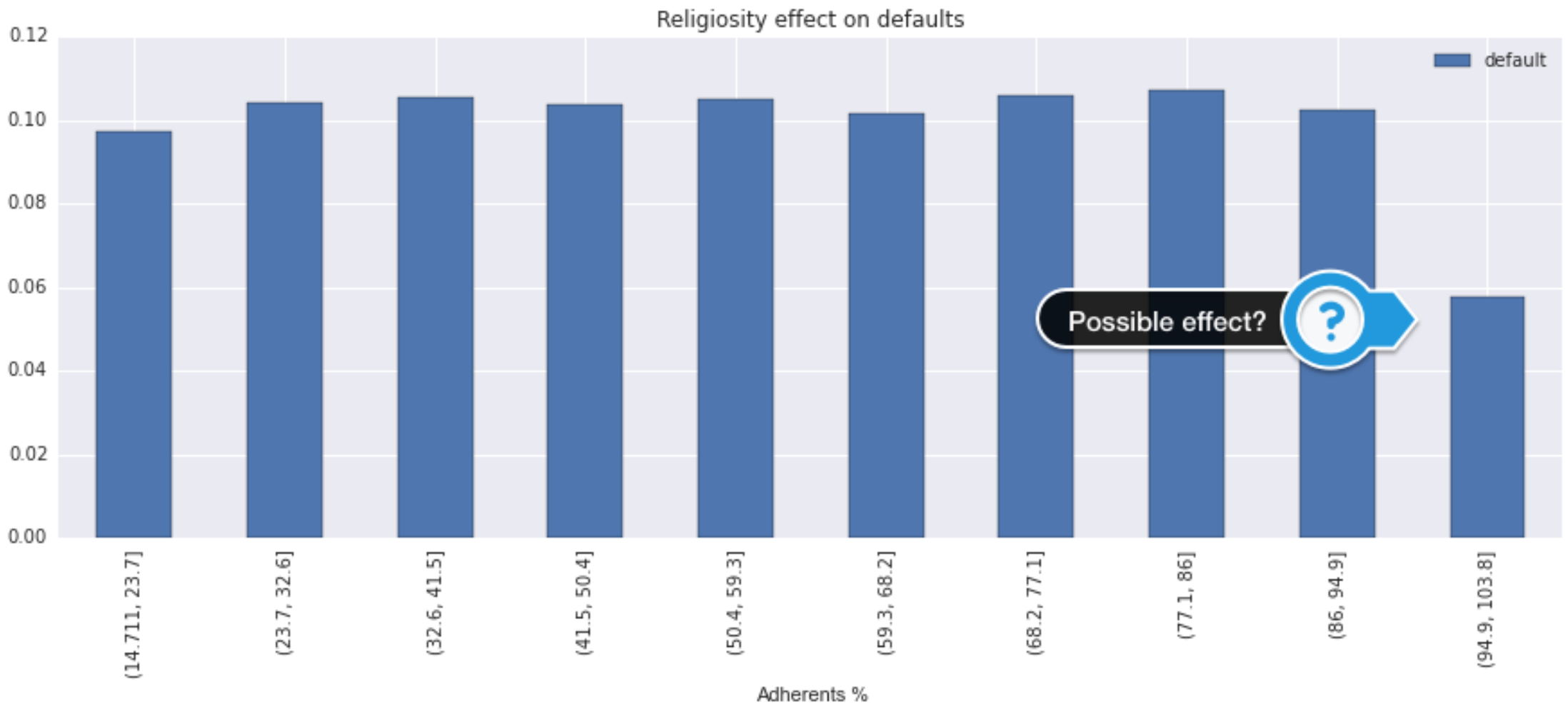
FICO



Mike Schoeffler

GA Data Science: DAT-NYC-30

Religion


Religiosity effect on defaults
default
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Possible effect?
(14.711, 23.7]
(23.7, 32.6]
(32.6, 41.5]
(41.5, 50.4]
(50.4, 59.3]
(59.3, 68.2]
(68.2, 77.1]
(77.1, 86]
(86, 94.9]
(94.9, 103.8]
Adherents %

Create dependent and independent data

```
import numpy as np
full_data = full_data[np.isfinite(full_data['Adherents %'])]
X = full_data[['dti', 'fico', 'Adherents %', 'loan_amnt', 'annual_inc', 'ranked_sub_grade', 'employment']]
#X = full_data[['Adherents %']]
y = full_data['default']
```

Split into training and test data

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
# 10 cross validation iterations with 20% test / 80% train
from sklearn.cross_validation import ShuffleSplit
cv = ShuffleSplit(X_train.shape[0], n_iter=10, test_size=0.2, random_state=0)
```

Set up regression

```
# Drop all features
lr_none = LogisticRegression(C= .00001, class_weight=None, penalty="l1")
lr_none.fit(X_train_std, y_train.apply(np.round))

# Drop the least predictive feature (turns out to be adherents)
lr_all_but_adherents = LogisticRegression(C= .05, class_weight=None, penalty="l1")
lr_all_but_adherents.fit(X_train_std, y_train.apply(np.round))

# include all features
lr_all = LogisticRegression(C= .5, class_weight=None, penalty=
lr_all.fit(X_train_std, y_train.apply(np.round))
```

Coefficients

```
pd.DataFrame({'features': X_train.columns, 'coefficients': lr_none.coef_[0]})
```

	coefficients	features
0	0	dti
1	0	fico
2	0	Adherents %
3	0	loan_amnt
4	0	annual_inc
5	0	ranked_sub_grade
6	0	employment

```
pd.DataFrame({'features': X_train.columns, 'coefficients': lr_all_but_adherents.coef_[
```

	coefficients	features
0	0.053639	dti
1	-0.128155	fico
2	0.000000	Adherents %
3	0.001957	loan_amnt
4	-0.280748	annual_inc
5	0.311255	ranked_sub_grade
6	0.035417	employment

```
pd.DataFrame({'features': X_train.columns, 'coefficien
```

	coefficients	features
0	0.059471	dti
1	-0.160048	fico
2	-0.010151	Adherents %
3	0.034970	loan_amnt
4	-0.330274	annual_inc
5	0.302917	ranked_sub_grade
6	0.052459	employment

Classification report

```
from sklearn.metrics import classification_report
print classification_report(lr_none.predict(X_test_std), y_test.apply(np.round), digits = 5)
```

```
             precision    recall  f1-score   support

        0.0    1.00000   0.92613   0.96165    217543
        1.0    0.00000   0.00000   0.00000         0

avg / total    1.00000   0.92613   0.96165    217543
```


```
print classification_report(lr_all_but_adherents.predict(X_test_std), y_test.apply(np.round), digits = 5)
```

```
             precision    recall  f1-score   support

        0.0    1.00000   0.92613   0.96165    217543
        1.0    0.00000   0.00000   0.00000         0

avg / total    1.00000   0.92613   0.96165    217543
```

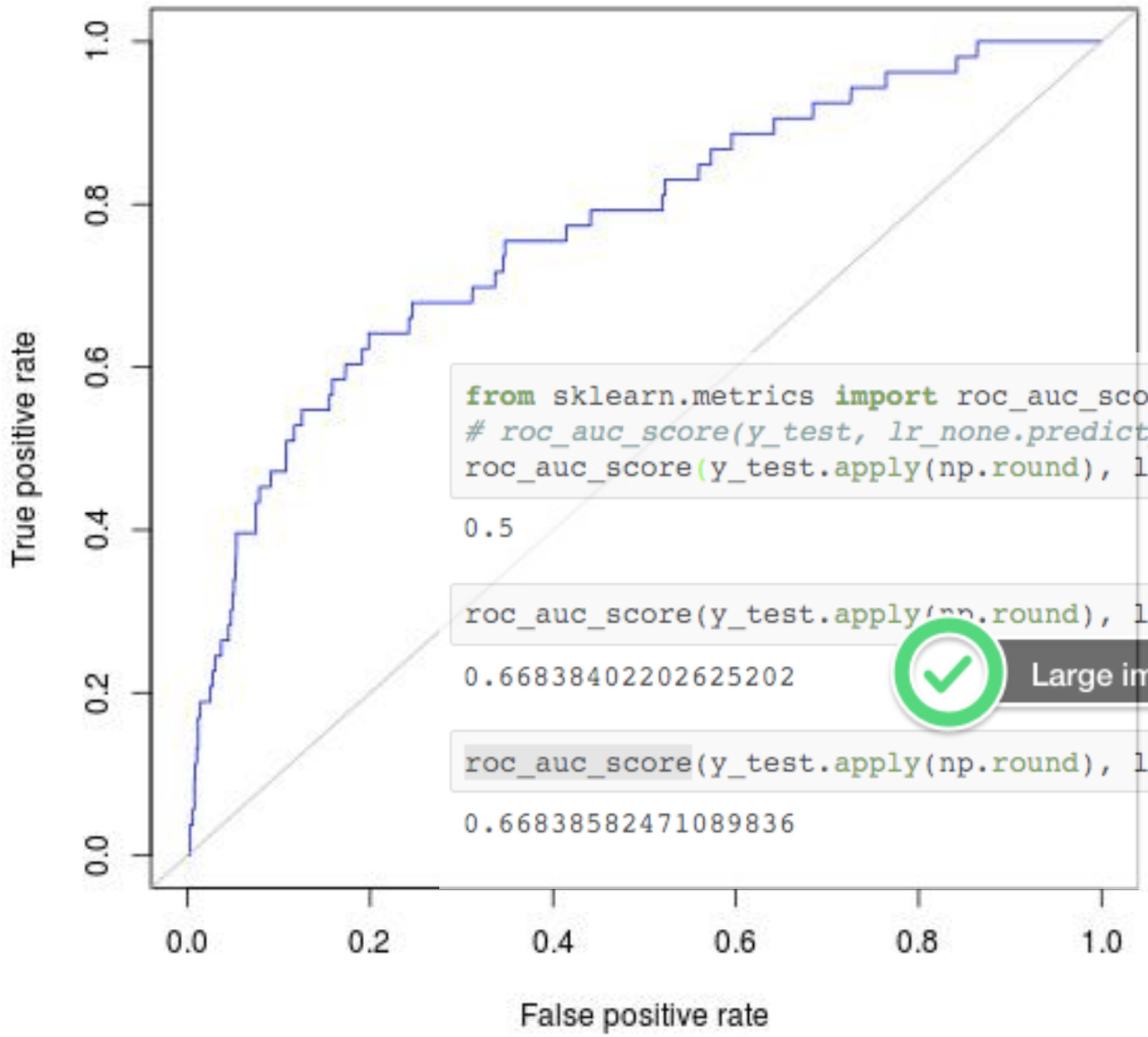

```
# test if we simply predict 0 (no default) every time
junk = np.empty(217543)
junk.fill(0)
print classification_report(junk, y_test.apply(np.round), digits = 5)
```

```
             precision    recall  f1-score   support

        0.0    1.00000   0.92613   0.96165    217543
        1.0    0.00000   0.00000   0.00000         0

avg / total    1.00000   0.92613   0.96165    217543
```

ROC Area Under Curve




```
from sklearn.metrics import roc_auc_score
# roc_auc_score(y_test, lr_none.predict_proba(X_test_std)[:,1])
roc_auc_score(y_test.apply(np.round), lr_none.predict_proba(X_test_std)[:,1])
```

```
0.5
```

```
roc_auc_score(y_test.apply(np.round), lr_all_but_adherents.predict_proba(X_test_std)[:,1])
```

```
0.66838402202625202
```

Large improvement

```
roc_auc_score(y_test.apply(np.round), lr_all.predict_proba(X_test_std)[:,1])
```

```
0.66838582471089836
```

Small improvement

Next steps

- Investigate better metrics
- Add more factors
- Aim for prediction
  - Nearest neighbors
  - ARIMA