Contents

1 Executive Summary

Mixed effect models are a type of regression model that take into account both
(1) variation that is explained by the independent variables of interest (like age) – fixed effects, and
(2) variation that is not explained by the independent variables of interest (like county)– random effects. Since the model includes a mixture of fixed and random effects, it's called a mixed model.

2 Introduction

Capturing the average time spent on a website

Improving a website's bounce rate and average time on a page will ultimately improve SEO (search engine optimization), The goal in this project is to find out if "Age" of a website visitor has any relationship with the average time spent on the website. For this purpose we have decided to perform a linear regression model and explain its findings.

3 Importing Libraries

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
from math import sqrt
import warnings
warnings.filterwarnings('ignore')
```

4 Loading Data

In [3]:
```
data=pd.read_csv("Downloads/data.csv")
```

In [4]:
```
data.head(5)
```

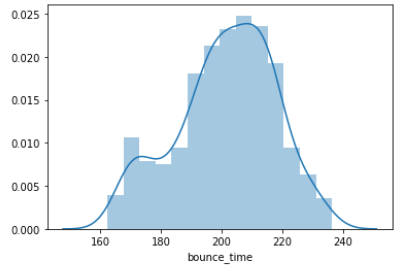
Out[4]:

	bounce_time	age	county	location
0	165.548520	16	devon	a
1	167.559314	34	devon	a
2	165.882952	6	devon	a
3	167.685525	19	devon	a
4	169.959681	34	devon	a

5 EDA

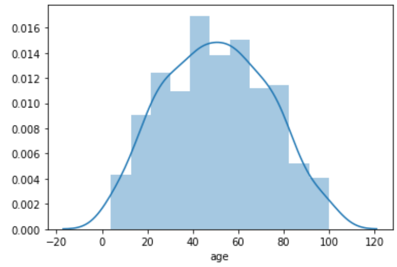
In [10]:
```
sns.distplot(data.bounce_time)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1416a1cee08>



In [13]:
```
sns.distplot(data.age)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1416a83e908>



In [15]:
```
from sklearn import preprocessing
```

In [16]:
```
data["age_scaled"] = preprocessing.scale(data.age.values)
```

Contents

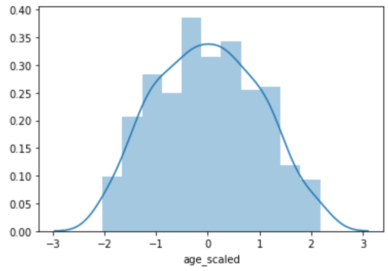
In [17]:
```
data.head(5)
```

Out[17]:

	bounce_time	age	county	location	age_scaled
0	165.548520	16	devon	a	-1.512654
1	167.559314	34	devon	a	-0.722871
2	165.882952	6	devon	a	-1.951423
3	167.685525	19	devon	a	-1.381024
4	169.959681	34	devon	a	-0.722871

In [19]:
```
sns.distplot(data.age_scaled)
```

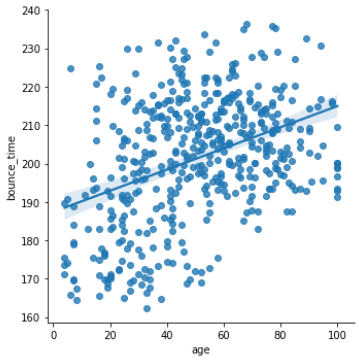
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x14168e8fb88>



first we check if the "bounce time" is dependent on the "age", later we fill fit a linear regression mdoel

In [21]:
```
sns.lmplot(x="age" , y= "bounce_time" , data = data)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x1416ab0ef88>



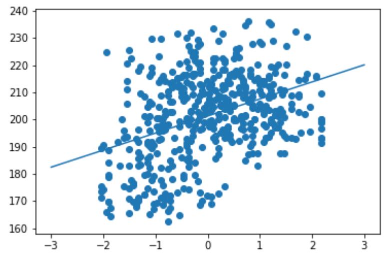
6 Construct a Linear Regression Model

In [25]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
x= data.age_scaled
y= data.bounce_time
model.fit(x[:, np.newaxis], y)
xfit = np.linspace(-3, 3, 1000)
yfit = model.predict(xfit[:, np.newaxis])
plt.plot(xfit, yfit)
plt.scatter(x, y)
```

Out[25]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [29]:
```
xfit = np.linspace(-3, 3, 1000)
yfit = model.predict(xfit[:, np.newaxis])
plt.plot(xfit, yfit)
plt.scatter(x, y)
```

Out[29]: <matplotlib.collections.PathCollection at 0x1416bda86c8>



In [35]:
```
print (model.coef_[0])
print (model.intercept_)
```

```
6.279602007970821
201.31646151854164
```

Contents

In [44]:
```
y_predict = model.predict (x.values.reshape(-1,1))
RMSE = sqrt(((y-y_predict)**2).values.mean())
results = pd.DataFrame()
results["Method"] = ["Linear Regression"]
results["RMSE"] = RMSE
results
```

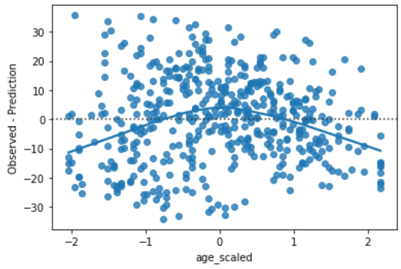
Out[44]:

	Method	RMSE
0	Linear Regression	14.928334

6.1 Residual Plot

We Regress y on x and then draw a scatterplot of the residuals

In [47]:
```
ax = sns.residplot(x = "age_scaled", y= "bounce_time", data = data, lowess = True)
ax.set(ylabel='Observed - Prediction')
plt.show()
```



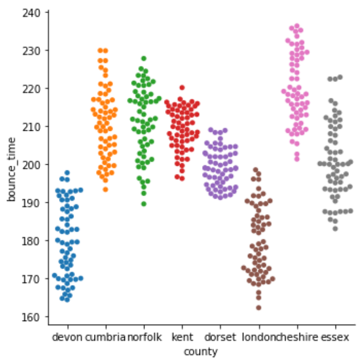
Main observation: there are more positive residuals than negative residuals at the highest and lowest predicted value ranges

6.2 Check in the Independance

we compare the bounce times for each county

In [48]:
```
sns.catplot(x="county", y="bounce_time", data=data, kind = "swarm")
```

Out[48]: <seaborn.axisgrid.FacetGrid at 0x1416bc7fcc8>



Conclusion: there is substantial grouping so our data is not independent, and thus it is inappropriate to use a linear model for this data.

Solution: seperate linear regression models for each county but we have to estimate a slope and intercept parameter for each regression and also it effectively reducing our sample sizes for each category.

7 Modelling (treating) County as a Fixed Effect

In [62]:
```
counties= data.county.unique()
```

In [53]:
```
data_new = pd.concat([data, pd.get_dummies(data.county)], axis = 1)
```

In [54]:
```
data_new
```

Out[54]:

	bounce_time	age	county	location	age_scaled	cheshire	cumbria	devon	dorset	essex	kent	london	norfolk
0	165.548520	16	devon	a	-1.512654	0	0	1	0	0	0	0	0
1	167.559314	34	devon	a	-0.722871	0	0	1	0	0	0	0	0
2	165.882952	6	devon	a	-1.951423	0	0	1	0	0	0	0	0
3	167.685525	19	devon	a	-1.381024	0	0	1	0	0	0	0	0
4	169.959681	34	devon	a	-0.722871	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	211.153312	82	essex	c	1.383217	0	0	0	0	1	0	0	0
476	213.577174	59	essex	c	0.374050	0	0	0	0	1	0	0	0
477	207.625105	69	essex	c	0.812818	0	0	0	0	1	0	0	0
478	198.252773	75	essex	c	1.076080	0	0	0	0	1	0	0	0
479	208.055977	66	essex	c	0.681188	0	0	0	0	1	0	0	0

480 rows × 13 columns

Contents

In [73]:
```
model = LinearRegression(fit_intercept=True)
x = data_new.loc[:,np.concatenate((["age_scaled"],counties))]
y = data.bounce_time
```

In [74]:
```
model.fit(x, y)
```

Out[74]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [75]:
```
y_predict = model.predict(x)
RMSE = sqrt(((y-y_predict)**2).values.mean())
results.loc[1] = ["Fixed", RMSE]
results
```

Out[75]:

	Method	RMSE
0	Linear Regression	14.928334
1	Fixed	8.563396

7.1 Checking the Coefficients

coefficients of age and the counties

In [76]:
```
pd.DataFrame.from_records(list(zip(np.concatenate((["age_scaled"],counties)), model.coef_)))
```

Out[76]:

	0	1
0	age_scaled	0.048782
1	devon	-21.381957
2	cumbria	9.391460
3	norfolk	9.824419
4	kent	7.938668
5	dorset	-2.079637
6	london	-21.437323
7	cheshire	18.372916
8	essex	-0.628546

Conclusion: The residuals are much better than before (more evenly distributed with respect to age). Coefficient for the gradient given to age is substnantially smaller, and is likely no longer significant.

Solution: we need to control for the variation between the different counties, we have to treat our counties as random effects. So in this new model we treat age (what we are interested in) as a fixed effect, and county and location as a random effect.

8 Build a Mixed Effect Model

First we look at how the bounce time relates to the scaled ages, while controlling for the impact of counties by allowing for a random intercept for each country (each county has its own random intercept, but that the slopes are still the same with respect to age)

In [78]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
md = smf.mixedlm("bounce_time ~ age_scaled", data, groups=data["county"])
mdf = md.fit()
print(mdf.summary())
```

```
          Mixed Linear Model Regression Results
=========================================================
Model:            MixedLM Dependent Variable: bounce_time
No. Observations: 480     Method:             REML
No. Groups:       8       Scale:              74.7350
Min. group size:  60      Log-Likelihood:     -1733.0397
Max. group size:  60      Converged:          Yes
Mean group size:  60.0
---------------------------------------------------------
             Coef.  Std.Err.   z    P>|z|  [0.025  0.975]
---------------------------------------------------------
Intercept   201.316    5.175 38.902 0.000 191.174 211.459
age_scaled    0.136    0.612  0.221 0.825  -1.065   1.336
Group Var   212.999   13.382
=========================================================
```

In [79]:
```
y_predict = mdf.fittedvalues
RMSE = sqrt(((y-y_predict)**2).values.mean())
results.loc[2] = ["Mixed", RMSE]
results
```

Out[79]:

	Method	RMSE
0	Linear Regression	14.928334
1	Fixed	8.563396
2	Mixed	8.563948

Conclusion: The residuals plot looks alomst identical to the previous one where we were treating the county as a fixed effect.

Solution: To ensure that each county has its own random slope we need to include this in our random effects forumla

Contents

In [87]:
```
md = smf.mixedlm("bounce_time ~ age_scaled", data, groups=data["county"], re_formula="~age_scaled")
mdf = md.fit()
print(mdf.summary())
```

```
               Mixed Linear Model Regression Results
====================================================================
Model:                  MixedLM     Dependent Variable:     bounce_time
No. Observations:       480         Method:                 REML
No. Groups:             8           Scale:                  72.8722
Min. group size:        60          Log-Likelihood:         -1733.3946
Max. group size:        60          Converged:              Yes
Mean group size:        60.0
--------------------------------------------------------------------
                        Coef.  Std.Err.   z    P>|z|  [0.025  0.975]
--------------------------------------------------------------------
Intercept               202.140    8.356 24.190 0.000 185.762 218.518
age_scaled                0.161    1.196  0.134 0.893  -2.184   2.505
Group Var               558.143
Group x age_scaled Cov  -51.614
age_scaled Var            8.621
====================================================================
```

Conclusion: The mixed model with the random slopes is now performing much better, with the residuals much better ditributed. Crucially though, we can see that age does not impact the bounce rate.

9 The Nested Random Effects

The random effects are sometimes nested. For example, there is nothing important about the locations a, b, and c that link location a in one county (London) with that in others (Essex). Therefore explicitly nest these two features.

In [81]:
```
data["location_county"] = data["location"] + "_" + data["county"]
```

In [82]:
```
data.head()
```

Out[82]:

	bounce_time	age	county	location	age_scaled	location_county
0	165.548520	16	devon	a	-1.512654	a_devon
1	167.559314	34	devon	a	-0.722871	a_devon
2	165.882952	6	devon	a	-1.951423	a_devon
3	167.685525	19	devon	a	-1.381024	a_devon
4	169.959681	34	devon	a	-0.722871	a_devon

In [88]:
```
md = smf.mixedlm("bounce_time ~ age_scaled", data, groups=data["location_county"], re_formula="~age_scaled")
mdf = md.fit()
print(mdf.summary())
```

```
               Mixed Linear Model Regression Results
====================================================================
Model:                  MixedLM     Dependent Variable:     bounce_time
No. Observations:       480         Method:                 REML
No. Groups:             24          Scale:                  23.7942
Min. group size:        20          Log-Likelihood:         -1504.9078
Max. group size:        20          Converged:              No
Mean group size:        20.0
--------------------------------------------------------------------
                        Coef.  Std.Err.   z    P>|z|  [0.025  0.975]
--------------------------------------------------------------------
Intercept               201.491    3.448 58.441 0.000 194.734 208.249
age_scaled                0.151    0.393  0.385 0.700  -0.618   0.920
Group Var               282.769   21.275
Group x age_scaled Cov   -8.285    2.478
age_scaled Var            0.386    0.494
====================================================================
```

In [85]:
```
y_predict = mdf.fittedvalues
RMSE = sqrt(((y-y_predict)**2).mean())
results.loc[3] = ["Nested_Mixed", RMSE]
results
```

Out[85]:

	Method	RMSE
0	Linear Regression	14.928334
1	Fixed	8.563396
2	Mixed	8.563948
3	Nested_Mixed	4.764192

10 Conclusion

We showed that by paying attension to a number of assumptions about the data (homoscedastic and Independence), the interpretaions of the obtained results out of a simple linear regression model could be different. That is to say such investigations can guide us towards building fixed or mixed effect models (sometimes called "multilevel models" or "hierarchical models").