

```
In [295]: import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
```

Cleaning Up Data

```
In [208]: # Much of the data was already cleaned in Excel beforehand
run = pd.read_csv("/Users/aiden/Desktop/RunningData.csv")
run = run.drop(run.columns[18:30], axis = 1)
run.head()
```

Out[208]:

	Activity Type	Date	Favorite	Title	Distance	Calories	Time	Avg HR	Max HR	Aerobic TE	Avg Cade
0	Running	5/17/2020 12:13	False	Centennial Running	10.03	946	0.051273	176	204	5.0	
1	Running	5/10/2020 15:00	False	Centennial Running	10.05	810	0.053403	156	196	4.0	
2	Running	5/4/2020 9:13	False	Centennial Running	5.02	460	0.026238	172	187	4.1	
3	Running	5/3/2020 9:56	False	Centennial Running	8.33	794	0.042188	179	199	5.0	
4	Running	4/30/2020 9:24	False	Littleton Running	5.00	484	0.025984	173	189	3.7	

```
In [94]: run.shape
```

Out[94]: (822, 17)

Question 1: What are the best predictors of average running pace?

```
In [182]: features = ["Distance", "Calories", "Time", "Avg HR", "Max HR", "Aerobic TE", "Aerobic TE", "Avg HR", "Calories", "Distance", "Elev Gain", "Elev Loss", "Avg Stride Length"]

pca = PCA()
pca.fit(run[features])
```

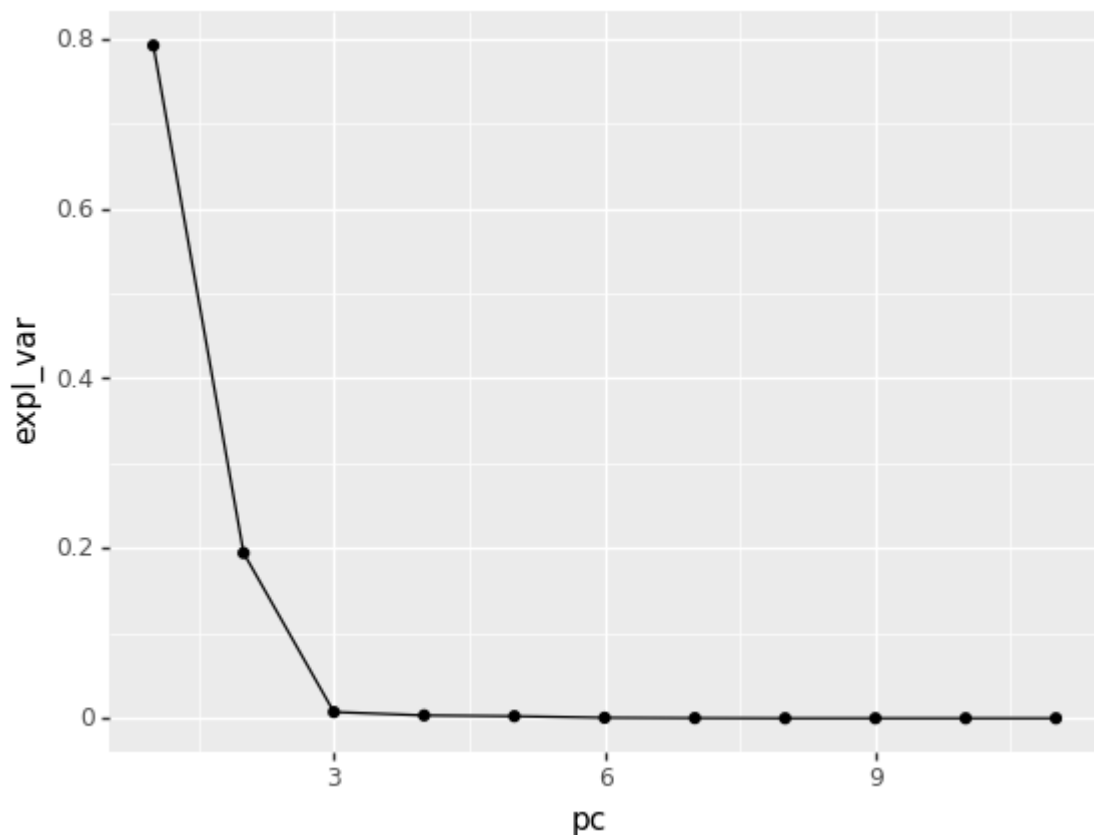
```
Out[182]: PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
            svd_solver='auto', tol=0.0, whiten=False)
```

```
In [183]: pcaDF = pd.DataFrame({"expl_var" : pca.explained_variance_ratio_, "pc": range(1,10)})
pcaDF.head()
```

```
Out[183]:
```

	expl_var	pc	cum_var
0	0.791649	1	0.791649
1	0.194417	2	0.986066
2	0.007316	3	0.993382
3	0.003360	4	0.996742
4	0.002417	5	0.999158

```
In [184]: (ggplot(pcaDF, aes(x = "pc", y = "expl_var")) + geom_line() + geom_point())
```



```
Out[184]: <ggplot: (-9223371912710995116)>
```

```
In [203]: pcomps1 = pca.transform(run[features])
pcomps1 = pd.DataFrame(pcomps1[:,1])

pcomps2 = pca.transform(run[features])
pcomps2 = pd.DataFrame(pcomps2[:, 1:2])

pcomps3 = pca.transform(run[features])
pcomps3 = pd.DataFrame(pcomps3[:, 1:3])

pcomps4 = pca.transform(run[features])
pcomps4 = pd.DataFrame(pcomps4[:, 1:4])

pcomps5 = pca.transform(run[features])
pcomps5 = pd.DataFrame(pcomps5[:, 1:5])

pcomps6 = pca.transform(run[features])
pcomps6 = pd.DataFrame(pcomps6[:, 1:6])

pcomps7 = pca.transform(run[features])
pcomps7 = pd.DataFrame(pcomps7[:, 1:7])

pcomps8 = pca.transform(run[features])
pcomps8 = pd.DataFrame(pcomps8[:, 1:8])

pcomps9 = pca.transform(run[features])
pcomps9 = pd.DataFrame(pcomps9[:, 1:9])

pcomps10 = pca.transform(run[features])
pcomps10 = pd.DataFrame(pcomps10[:, 1:10])

#modeMod1
rr = Ridge()
rr.fit(run[features], run["Avg Pace"])
print("all data: ", rr.score(run[features], run["Avg Pace"]))

#modeMod1
rr1 = Ridge()
rr1.fit(pcomps1, run["Avg Pace"])
print("1 PCs:  ", rr1.score(pcomps1, run["Avg Pace"]))

#modeMod1
rr2 = Ridge()
rr2.fit(pcomps2, run["Avg Pace"])
print("2 PCs:  ", rr2.score(pcomps2, run["Avg Pace"]))

#modeMod1
rr3 = Ridge()
rr3.fit(pcomps3, run["Avg Pace"])
print("3 PCs:  ", rr3.score(pcomps3, run["Avg Pace"]))

#modeMod1
rr4 = Ridge()
rr4.fit(pcomps4, run["Avg Pace"])
print("4 PCs:  ", rr4.score(pcomps4, run["Avg Pace"]))

#modeMod1
```

```
rr5 = Ridge()
rr5.fit(pcomps5, run["Avg Pace"])
print("5 PCs:  ", rr5.score(pcomps5, run["Avg Pace"]))

#modeMod1
rr6 = Ridge()
rr6.fit(pcomps6, run["Avg Pace"])
print("6 PCs:  ", rr6.score(pcomps6, run["Avg Pace"]))

#modeMod1
rr7 = Ridge()
rr7.fit(pcomps7, run["Avg Pace"])
print("7 PCs:  ", rr7.score(pcomps7, run["Avg Pace"]))

#modeMod1
rr8 = Ridge()
rr8.fit(pcomps8, run["Avg Pace"])
print("8 PCs:  ", rr8.score(pcomps8, run["Avg Pace"]))

#modeMod1
rr9 = Ridge()
rr9.fit(pcomps9, run["Avg Pace"])
print("9 PCs:  ", rr9.score(pcomps9, run["Avg Pace"]))

#modeMod1
rr10 = Ridge()
rr10.fit(pcomps10, run["Avg Pace"])
print("10 PCs:  ", rr10.score(pcomps10, run["Avg Pace"]))
```

```
all data:  0.7637036677007885
1 PCs:    0.009229260081699109
2 PCs:    0.009229260081699109
3 PCs:    0.012016595101565164
4 PCs:    0.09674334259466677
5 PCs:    0.3228949734422327
6 PCs:    0.48864146383678964
7 PCs:    0.4918817846119433
8 PCs:    0.5258517930547029
9 PCs:    0.5333716532044763
10 PCs:   0.6498168285443847
```

```
In [186]: loadings = pd.DataFrame({"loading": pca.components_.flatten(),
                                "comp": np.repeat(range(1,12), 11 ,
                                axis=0), "variable":np.tile(features,11) })

loadings.head(60)
```

Out[186]:

	loading	comp	variable
0	0.009193	1	Distance
1	0.854445	1	Calories
2	0.000049	1	Time
3	0.020514	1	Avg HR
4	0.029426	1	Max HR
5	0.003082	1	Aerobic TE
6	-0.013926	1	Avg Run Cadence
7	-0.001247	1	Max Run Cadence
8	0.367175	1	Elev Gain
9	0.365417	1	Elev Loss
10	-0.000595	1	Avg Stride Length
11	-0.005817	2	Distance
12	-0.516821	2	Calories
13	-0.000028	2	Time
14	-0.021354	2	Avg HR
15	-0.023230	2	Max HR
16	-0.002329	2	Aerobic TE
17	0.011726	2	Avg Run Cadence
18	0.016633	2	Max Run Cadence
19	0.594961	2	Elev Gain
20	0.614385	2	Elev Loss
21	0.000403	2	Avg Stride Length
22	0.000926	3	Distance
23	0.008424	3	Calories
24	0.000004	3	Time
25	-0.021619	3	Avg HR
26	-0.021214	3	Max HR
27	-0.000293	3	Aerobic TE
28	-0.028878	3	Avg Run Cadence
29	-0.064236	3	Max Run Cadence
30	-0.712306	3	Elev Gain

	loading	comp	variable
31	0.697617	3	Elev Loss
32	-0.000717	3	Avg Stride Length
33	0.018600	4	Distance
34	0.049218	4	Calories
35	0.000092	4	Time
36	-0.660720	4	Avg HR
37	-0.619603	4	Max HR
38	-0.019492	4	Aerobic TE
39	0.210582	4	Avg Run Cadence
40	0.363027	4	Max Run Cadence
41	-0.010530	4	Elev Gain
42	-0.008543	4	Elev Loss
43	0.008735	4	Avg Stride Length
44	-0.008772	5	Distance
45	-0.004050	5	Calories
46	-0.000066	5	Time
47	0.272717	5	Avg HR
48	0.319557	5	Max HR
49	0.002009	5	Aerobic TE
50	0.390789	5	Avg Run Cadence
51	0.815216	5	Max Run Cadence
52	-0.060793	5	Elev Gain
53	0.047412	5	Elev Loss
54	0.013212	5	Avg Stride Length
55	0.009548	6	Distance
56	-0.011389	6	Calories
57	0.000107	6	Time
58	-0.111530	6	Avg HR
59	0.070967	6	Max HR

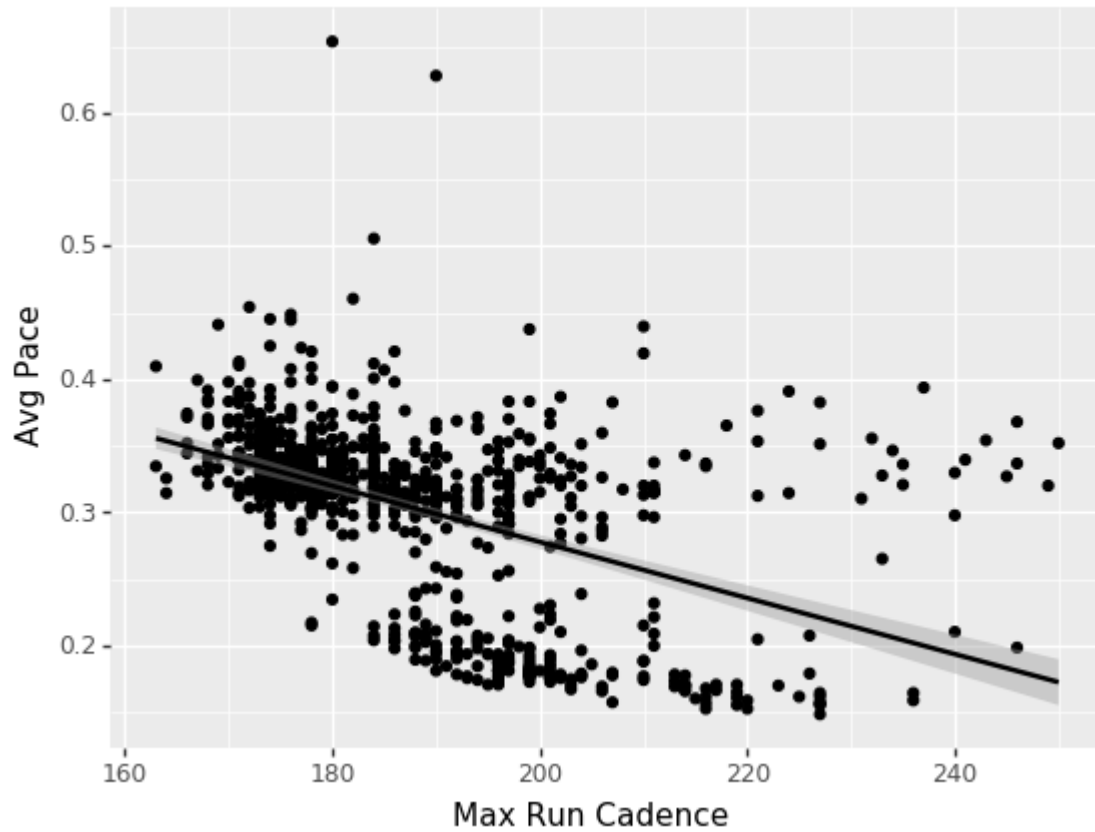
In [187]: `loadings.tail(60)`

Out[187]:

	loading	comp	variable
61	-8.896320e-01	6	Avg Run Cadence
62	4.361595e-01	6	Max Run Cadence
63	-3.821055e-03	6	Elev Gain
64	-1.759479e-03	6	Elev Loss
65	-2.257693e-02	6	Avg Stride Length
66	1.877379e-02	7	Distance
67	-6.193060e-03	7	Calories
68	1.341217e-04	7	Time
69	-6.887082e-01	7	Avg HR
70	7.119732e-01	7	Max HR
71	-6.275990e-04	7	Aerobic TE
72	9.695584e-02	7	Avg Run Cadence
73	-9.472669e-02	7	Max Run Cadence
74	1.514717e-03	7	Elev Gain
75	-2.804885e-03	7	Elev Loss
76	4.408200e-04	7	Avg Stride Length
77	9.806861e-01	8	Distance
78	-1.084989e-02	8	Calories
79	3.704607e-03	8	Time
80	3.034826e-02	8	Avg HR
81	2.457924e-03	8	Max HR
82	-1.924635e-01	8	Aerobic TE
83	3.629332e-03	8	Avg Run Cadence
84	-2.050375e-03	8	Max Run Cadence
85	4.112213e-04	8	Elev Gain
86	1.576157e-04	8	Elev Loss
87	1.168008e-02	8	Avg Stride Length
88	1.929800e-01	9	Distance
89	-4.944037e-03	9	Calories
90	8.668218e-04	9	Time
91	-7.125455e-03	9	Avg HR
92	-1.304944e-02	9	Max HR
93	9.779117e-01	9	Aerobic TE
94	1.525765e-02	9	Avg Run Cadence

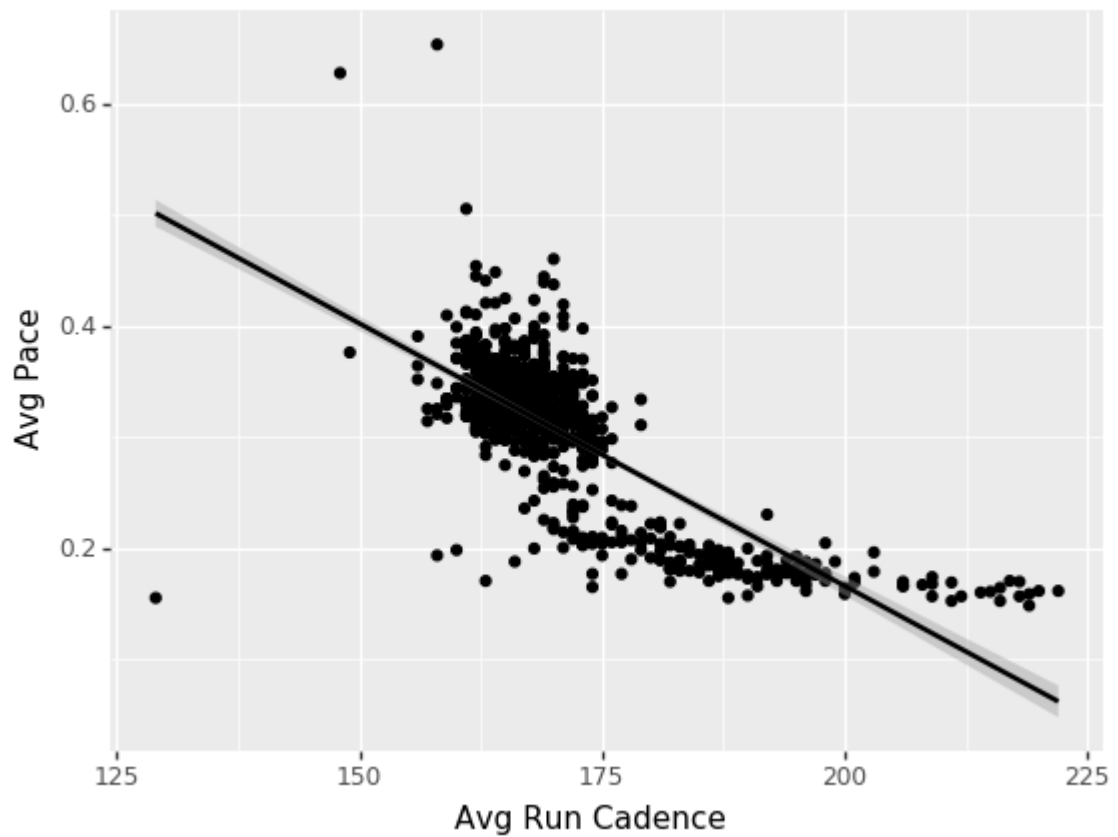
	loading	comp	variable
95	1.065845e-03	9	Max Run Cadence
96	5.905409e-05	9	Elev Gain
97	3.072025e-04	9	Elev Loss
98	-7.724725e-02	9	Avg Stride Length
99	3.641017e-03	10	Distance
100	-1.636847e-04	10	Calories
101	-1.721154e-03	10	Time
102	-9.489182e-04	10	Avg HR
103	1.458783e-03	10	Max HR
104	7.842693e-02	10	Aerobic TE
105	-2.611789e-02	10	Avg Run Cadence
106	-4.013909e-03	10	Max Run Cadence
107	2.767398e-04	10	Elev Gain
108	-9.825268e-05	10	Elev Loss
109	9.965599e-01	10	Avg Stride Length
110	-3.800552e-03	11	Distance
111	-1.516111e-05	11	Calories
112	9.999913e-01	11	Time
113	7.367117e-05	11	Avg HR
114	-2.227557e-05	11	Max HR
115	1.055401e-06	11	Aerobic TE
116	1.808727e-05	11	Avg Run Cadence
117	-1.283860e-05	11	Max Run Cadence
118	-2.032220e-06	11	Elev Gain
119	-4.359067e-08	11	Elev Loss
120	1.741406e-03	11	Avg Stride Length

```
In [190]: (ggplot(run, aes("Max Run Cadence", "Avg Pace")) + geom_point() + geom_smooth(me
```



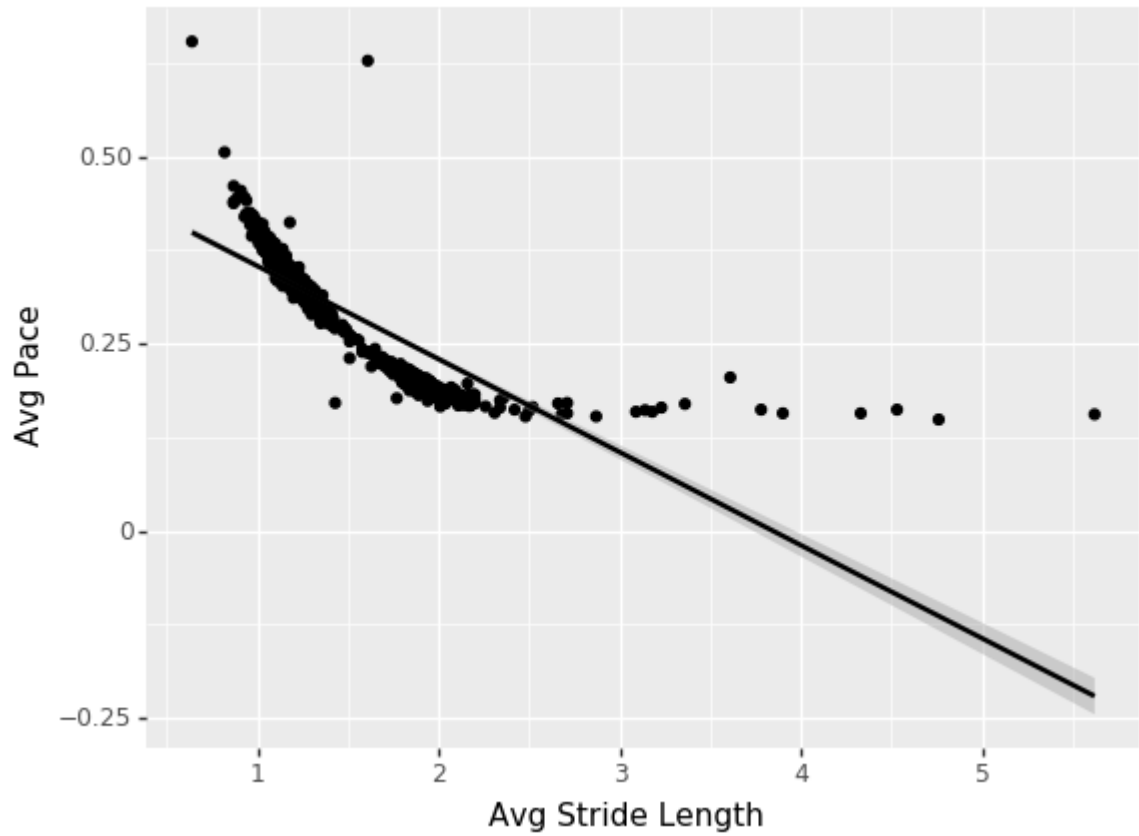
```
Out[190]: <ggplot: (-9223371912712494340)>
```

```
In [204]: (ggplot(run, aes("Avg Run Cadence", "Avg Pace")) + geom_point() + geom_smooth(me
```



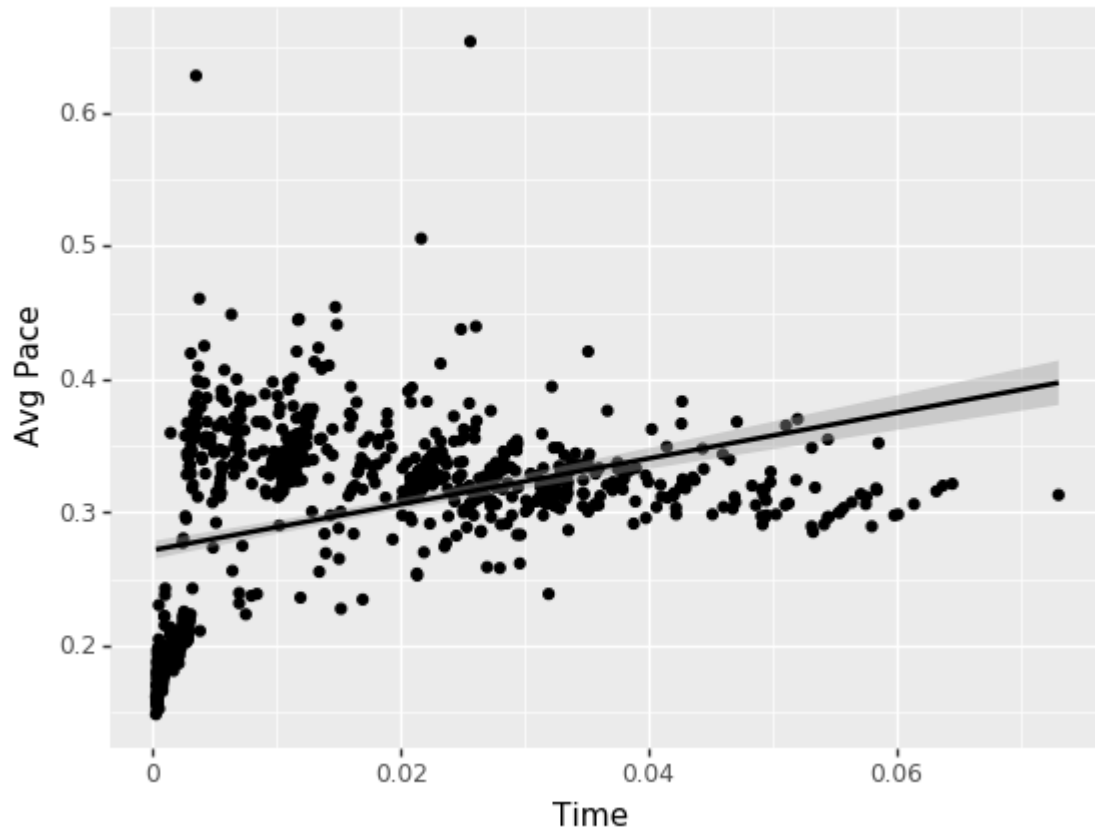
```
Out[204]: <ggplot: (-9223371912711199604)>
```

```
In [191]: (ggplot(run, aes("Avg Stride Length", "Avg Pace")) + geom_point() + geom_smooth(r
```



```
Out[191]: <ggplot: (-9223371912709910868)>
```

```
In [205]: (ggplot(run, aes("Time", "Avg Pace")) + geom_point() + geom_smooth(method = "lm"
```



```
Out[205]: <ggplot: (-9223371912711224780)>
```

Question 1 Conclusions

```
In [ ]: # I performed a PCA and found that 98.6% of the total variance in the data
# was explained by the first two components. That showed me that I had a good
# model, but it did not answer my original question, because it did not tell
# me which components explained the variance of Average Pace specifically,
# and what variables those components were made up of. So then I fit a ridge
# regression model for predicting Average Pace using the first component,
# the first two components, etc. all the way up to using all of the components,
# and printed out the score for how well each model predicted Average Pace.
# By looking at the differences between the scores, I determined that the
# components best at predicting Average Pace were the 5th, 6th, 10th, and 11th.
# Then I found the Loadings for each variable for each component so I could deter
# which variables were the most influential in the 5th, 6th, 10th, and 11th compo
# Here were the most significant variables and their Loadings:
# 5th component: Avg HR(.27), Max HR(.32), Avg Cadence(.39), Max Cadence(.82)
# 6th component: Avg HR(-.11), Avg Cadence(-.88), Max Cadence(-.44)
# 10th component: Avg Stride Length(-.99)
# 11th component: Time(-.99)

# The 5th and 6th components caused the greatest increases on predictive score,
# and the variables with the highest Loadings in these two components were
# Avg Cadence and Max Cadence. Because of this, I am lead to believe that
# these two variables are the best predictors of Avg Pace, followed by
# Avg Stride Length and Time.

# These results are supported by my background knowledge that pace
# is a measure of speed, and speed can be calculated by multiplying
# cadence with stride length.
```

Question 2: Is run location correlated with average pace (and other variables?)

```
In [243]: features = ["Avg Pace", "Distance", "Calories", "Time", "Avg HR", "Aerobic TE",
                    "Avg Stride Length", "Location"]
X = run[features]

z = StandardScaler()
X[features] = z.fit_transform(X)

km = KMeans(n_clusters = 5)
km.fit(X)

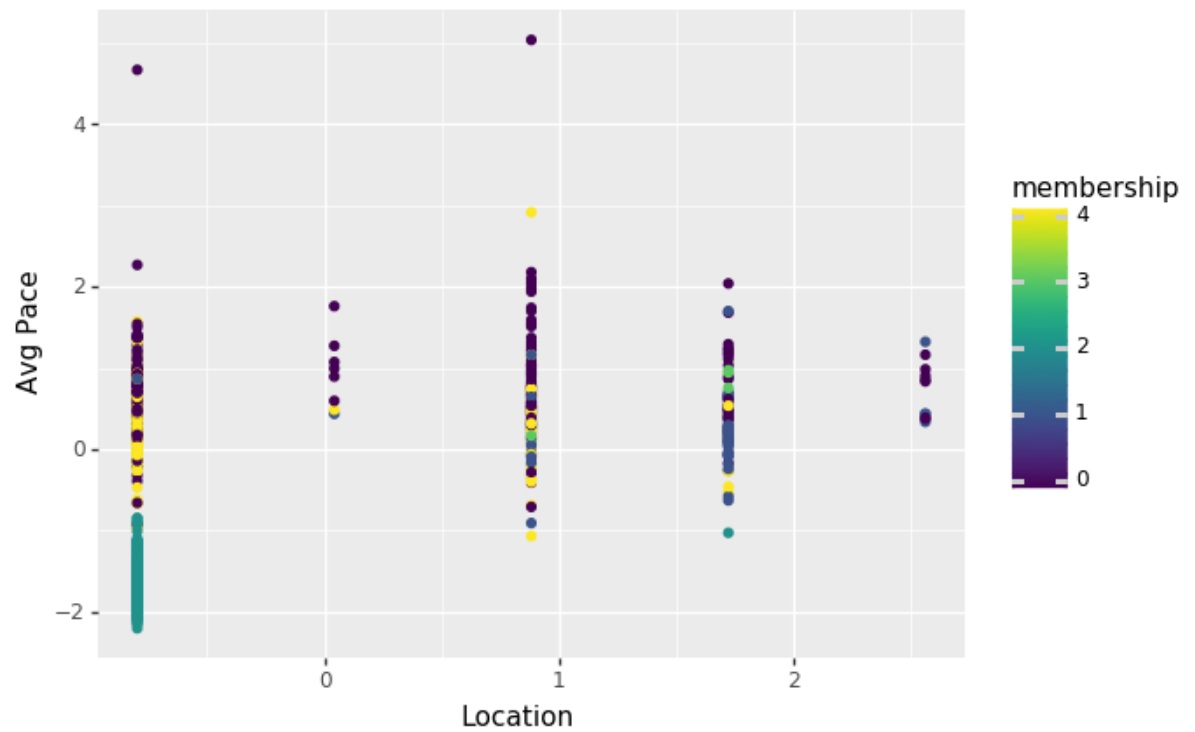
membership = km.predict(X)

X["cluster"] = membership

silhouette_score(X, membership)
```

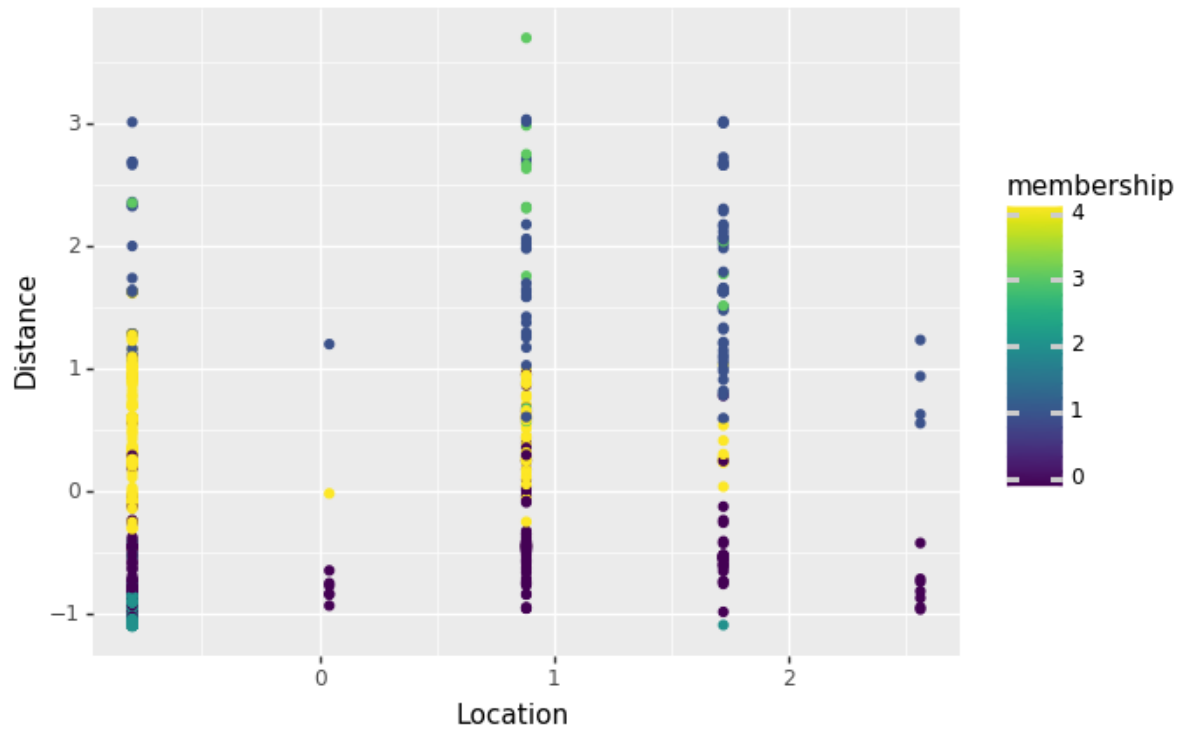
Out[243]: 0.48649471315303733

```
In [244]: (ggplot(X, aes("Location", "Avg Pace", color = "membership")) + geom_point())
```



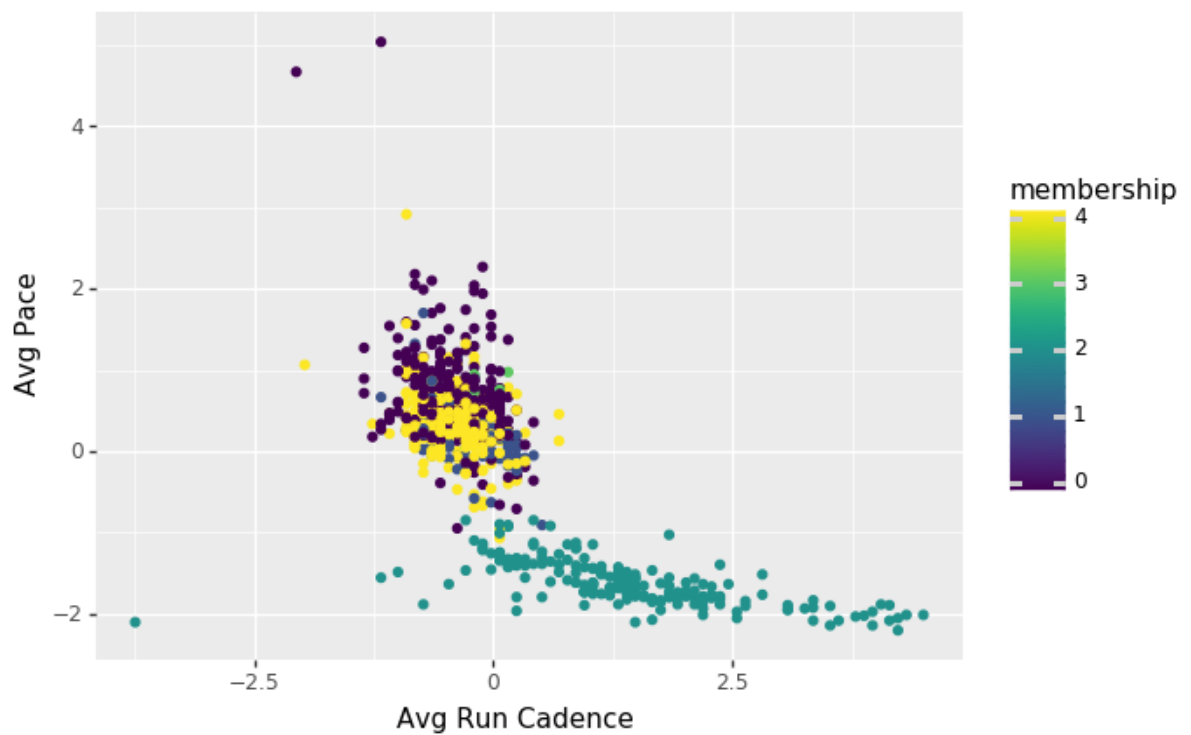
```
Out[244]: <ggplot: (-9223371912706904172)>
```

```
In [249]: (ggplot(X, aes("Location", "Distance", color = "membership")) + geom_point())
```



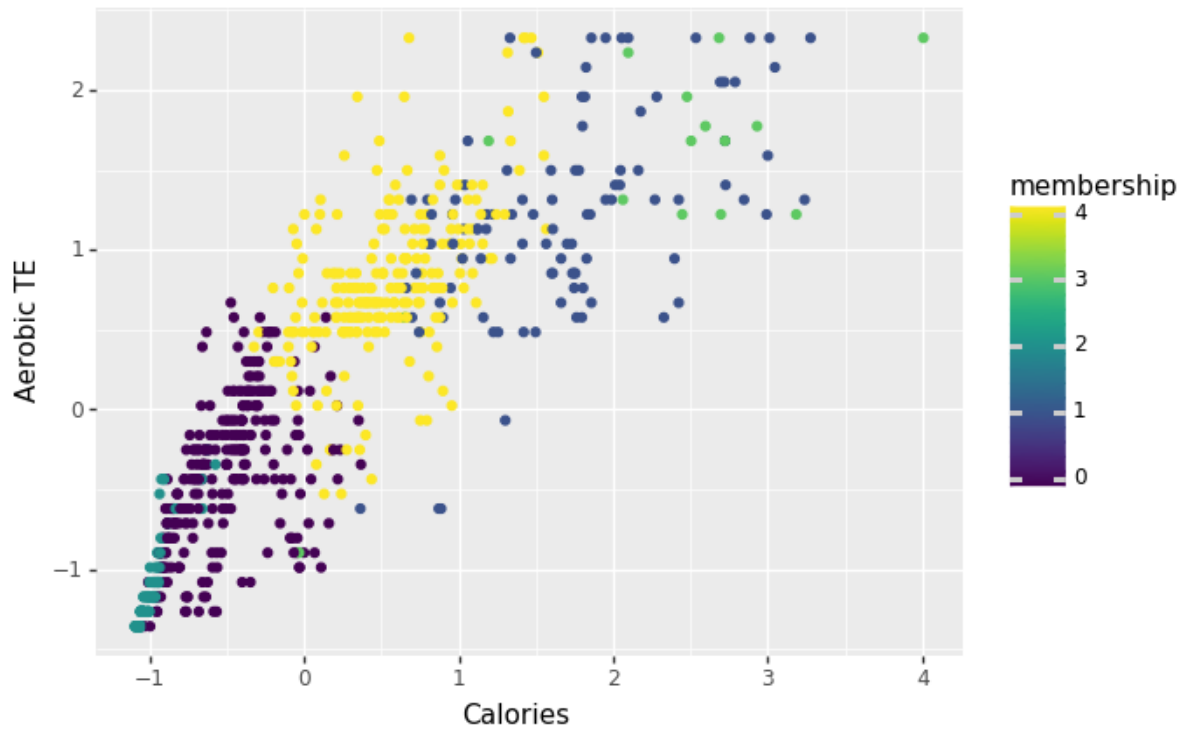
```
Out[249]: <ggplot: (-9223371912706914352)>
```

```
In [248]: (ggplot(X, aes("Avg Run Cadence", "Avg Pace", color = "membership"))) + geom_point()
```



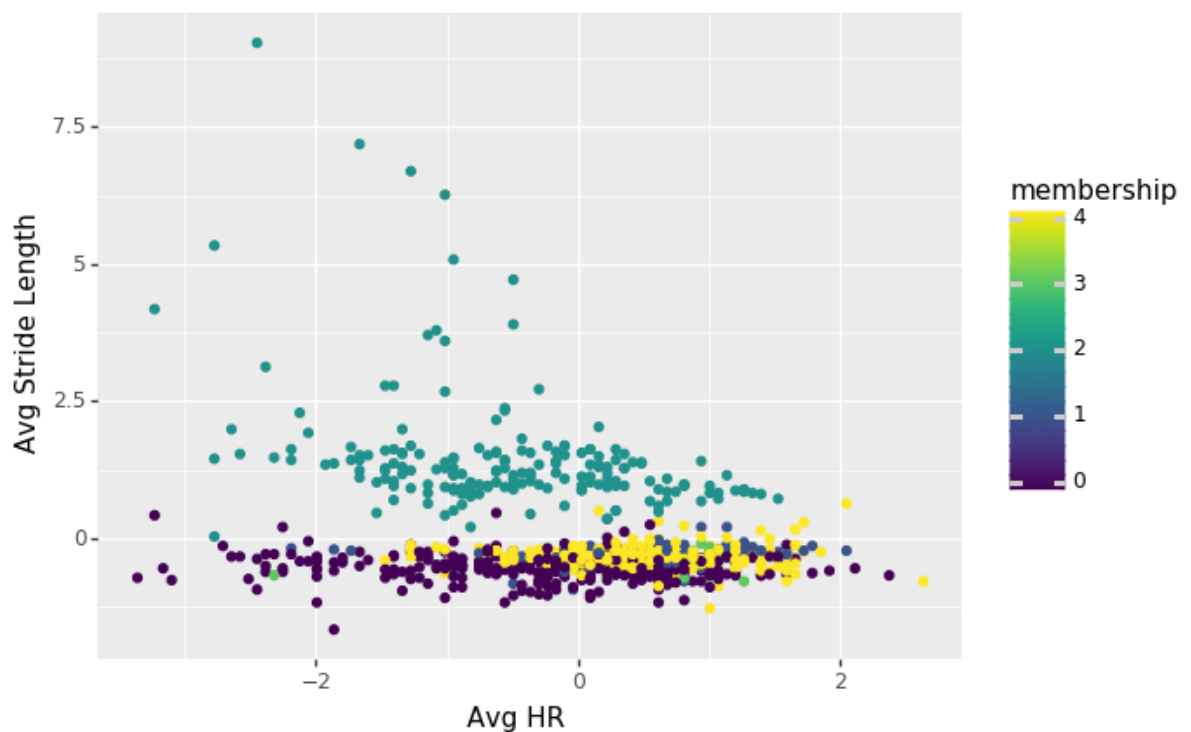
```
Out[248]: <ggplot: (-9223371912707992804)>
```

```
In [250]: (ggplot(X, aes("Calories", "Aerobic TE", color = "membership")) + geom_point())
```



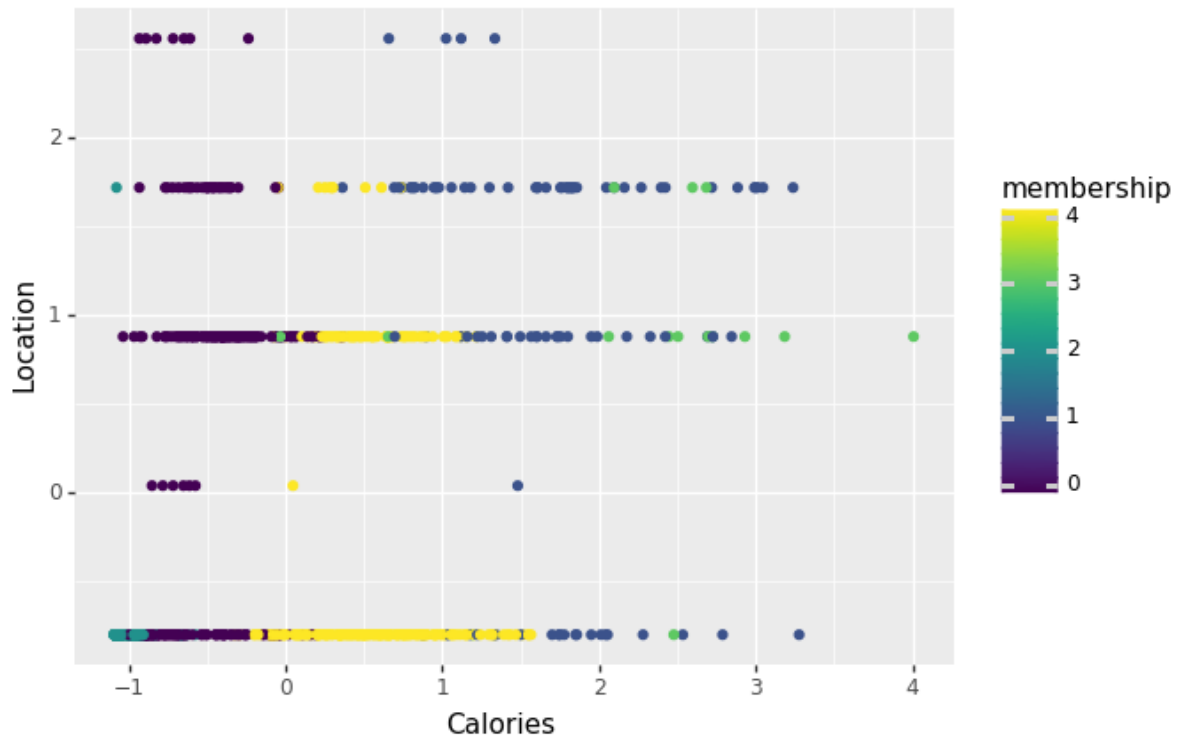
```
Out[250]: <ggplot: (-9223371912709865360)>
```

```
In [251]: (ggplot(X, aes("Avg HR", "Avg Stride Length", color = "membership")) + geom_point())
```



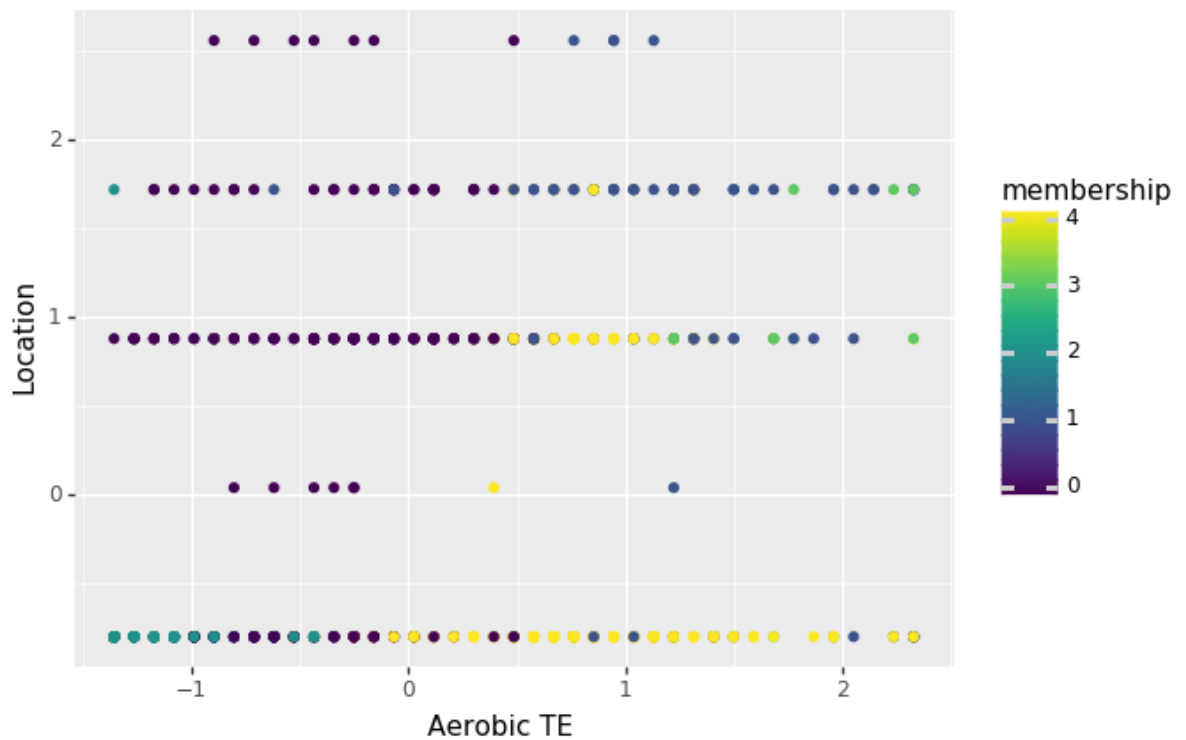
```
Out[251]: <ggplot: (-9223371912711191476)>
```

```
In [252]: (ggplot(X, aes("Calories", "Location", color = "membership")) + geom_point())
```



```
Out[252]: <ggplot: (-9223371912711266456)>
```

```
In [253]: (ggplot(X, aes("Aerobic TE", "Location", color = "membership")) + geom_point())
```



```
Out[253]: <ggplot: (-9223371912712416652)>
```

Question 2 Conclusions

```
In [ ]: # I made a K-Means model using Location and many other variables. I thought
# that Location might be a good indicator of many of the other variables,
# so the model might cluster based on Location. But, when I graphed the
# clusters, there was a good mix of some members of every cluster at
# each different running location. So the model must have used variables
# other than Location to classify. That left me with the question, what
# variables did it use? I looked at a couple graphs of the clusters in
# terms of other variables, and it appears that Aerobic TE and Calories
# may have actually been some of the most important variables in the
# classification. So to answer the original question, it does not
# appear that Location is correlated very highly with Avg Pace
# or other variables. Instead, Calories and Aerobic TE are much
# better indicators of the run as a whole.
```

Question 3: Can I predict Avg HR using other data from the run?

```
In [298]: predictors = ["Avg Pace", "Distance", "Calories", "Time", "Aerobic TE", "Avg Run
                "Avg Stride Length", "Location"]

X_train, X_test, y_train, y_test = train_test_split(run[predictors], run["Avg HR"],
X_train.head()

zscore = StandardScaler()
zscore.fit(X_train)
Xz_train = zscore.transform(X_train)
Xz_test = zscore.transform(X_test)
```

```
In [299]: lr = LinearRegression()
lr.fit(Xz_train, y_train)
```

```
Out[299]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [300]: predictedVals = lr.predict(Xz_test)
```

```
In [303]: r2_score(y_test, predictedVals)
```

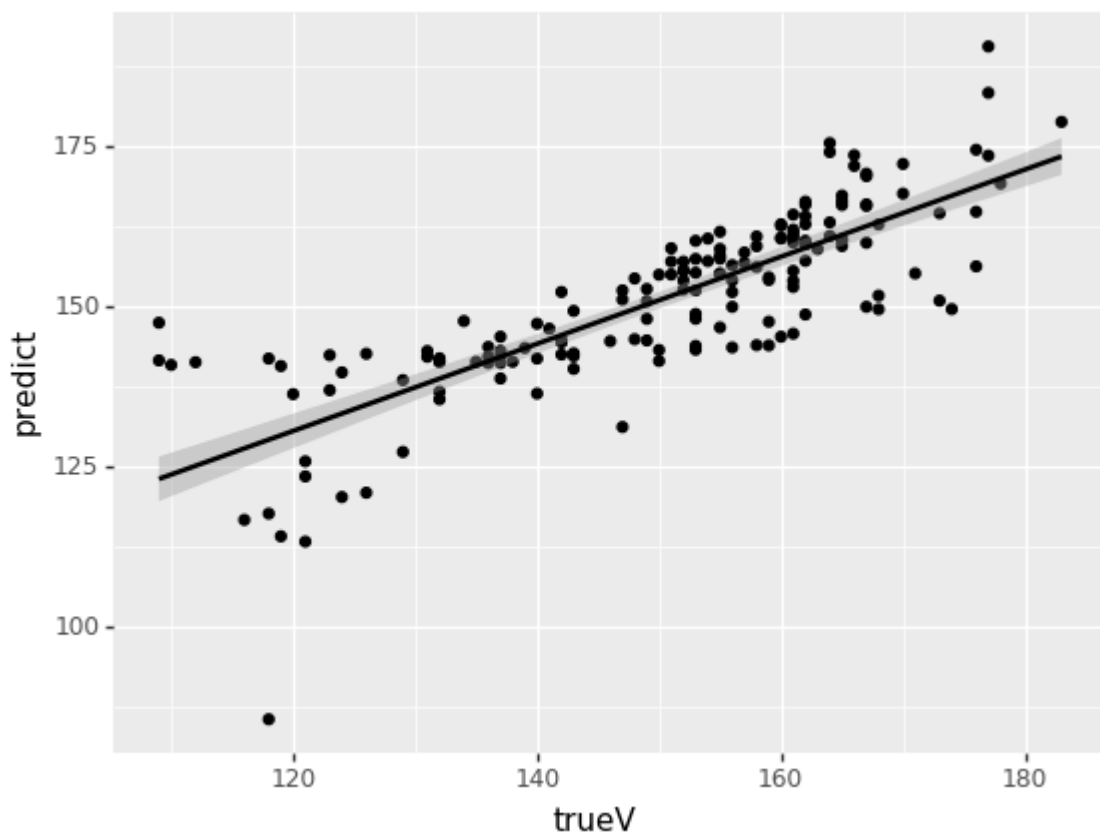
```
Out[303]: 0.64056237427221
```

```
In [305]: true_vs_pred = pd.DataFrame({"predict": predictedVals, "trueV": y_test})  
true_vs_pred.head()
```

Out[305]:

	predict	trueV
259	160.809959	160
629	140.753862	119
608	149.628065	168
446	170.436284	167
567	144.540154	142

```
In [307]: (ggplot(true_vs_pred, aes(x = "trueV", y = "predict")) + geom_point() + geom_smooth)
```



Out[307]: <ggplot: (-9223371912712479164)>

Question 3 Conclusions

```
In [ ]: # I fit a linear regression model to the chosen predictor variables, and used  
# Train Test Split cross-validation. I calculated an r^2 score of 0.64 using the  
# model on the test set, and looked at how the predicted Avg HR values compared  
# to the actual values. Based on this, I would say that the model can predict  
# Avg HR using other data from the model with decent accuracy.
```