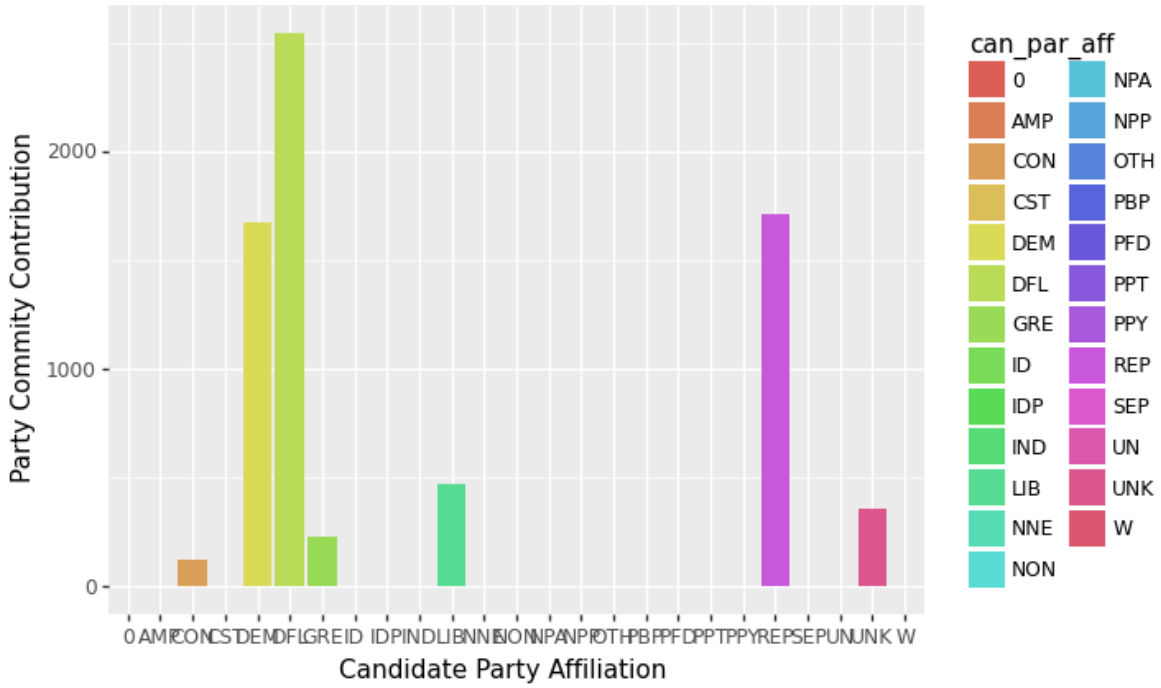
In [1]:
```
import warnings
warnings.filterwarnings('ignore')
from plotnine import *

from sklearn.decomposition import PCA
import pandas as pd

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_
error
from sklearn.model_selection import train_test_split # simple TT split c
v
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.linear_model import RidgeCV, LassoCV

from sklearn.metrics import silhouette_score

import numpy as np
```

In [ ]:
```
```

In [4]:
```
election = pd.read_csv("CandidateSummary.csv")

election  = election.replace(to_replace = "Y", value = 1).replace(to_rep
lace = np.nan, value = 0)
```

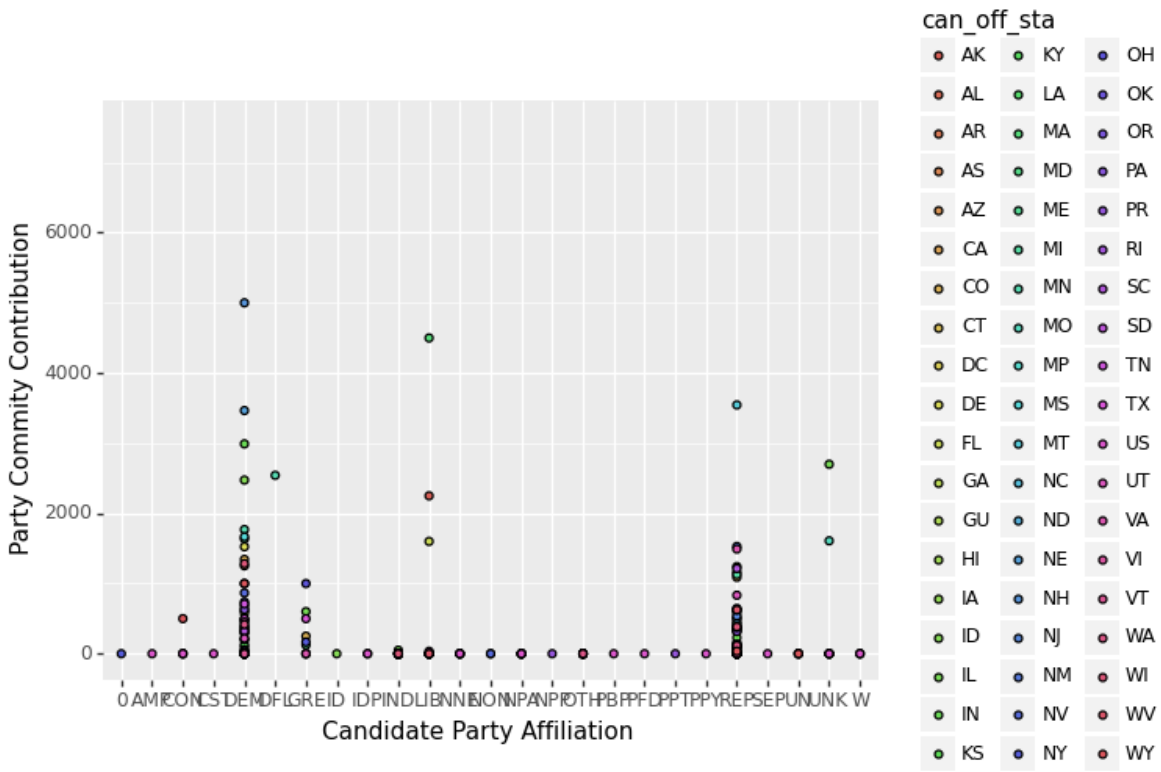
In [6]:
```
#1
(ggplot(election, aes(x = "can_par_aff", y = "par_com_con", fill = "can_
par_aff")) +
stat_summary(fun_data = "mean_sdl", geom = "bar")+labs(x = "Candidate Pa
rty Affiliation", y = "Party Commity Contribution"))
```



Out[6]: <ggplot: (8736683592424)>

We used a bar chart depicting each party and how their average contributions to candidates compare

In [10]:
```
(ggplot(election, aes(x = "can_par_aff", y = "par_com_con", fill = "can_
off_sta")) + ylim(0,7500) +
stat_summary(fun_data = "mean_sdl", geom = "point")+labs(x = "Candidate
 Party Affiliation", y = "Party Commity Contribution"))
```



Out[10]: <ggplot: (-9223633300176693020)>

This shows the dispursement of party committees to their candidate in each state.

1. On average, and surprisingly, the Democratic-Farmer-Labor Party (DFL) donates the most to its candidates, followed by the Republican Party (REP) barely edging out the Democrats (DEM). A few more notable parties are the Libertarian (LIB), Unknown (UNK), and Green Parties (GRE). We decided instead of using a clustering method, we would just show data visualizations to better interpret the averages of each parties contributions.

In [ ]:
```
#2
```

In [11]:
```
l_predictor = ["tot_con"]

X_train, X_test, y_train, y_test = train_test_split(election[l_predictor
], election["votes"], test_size=0.2, random_state = 2)
```

In [12]:
```
LR_Model = LinearRegression()
LR_Model.fit(X_train, y_train)
```

Out[12]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)
```

In [13]:
```
y_pred = LR_Model.predict(X_test)

LR_Model.score(X_train, y_train)
```

Out[13]:
```
0.0007744525312664008
```

In [14]:
```
LR_Model.score(X_test, y_test)
```

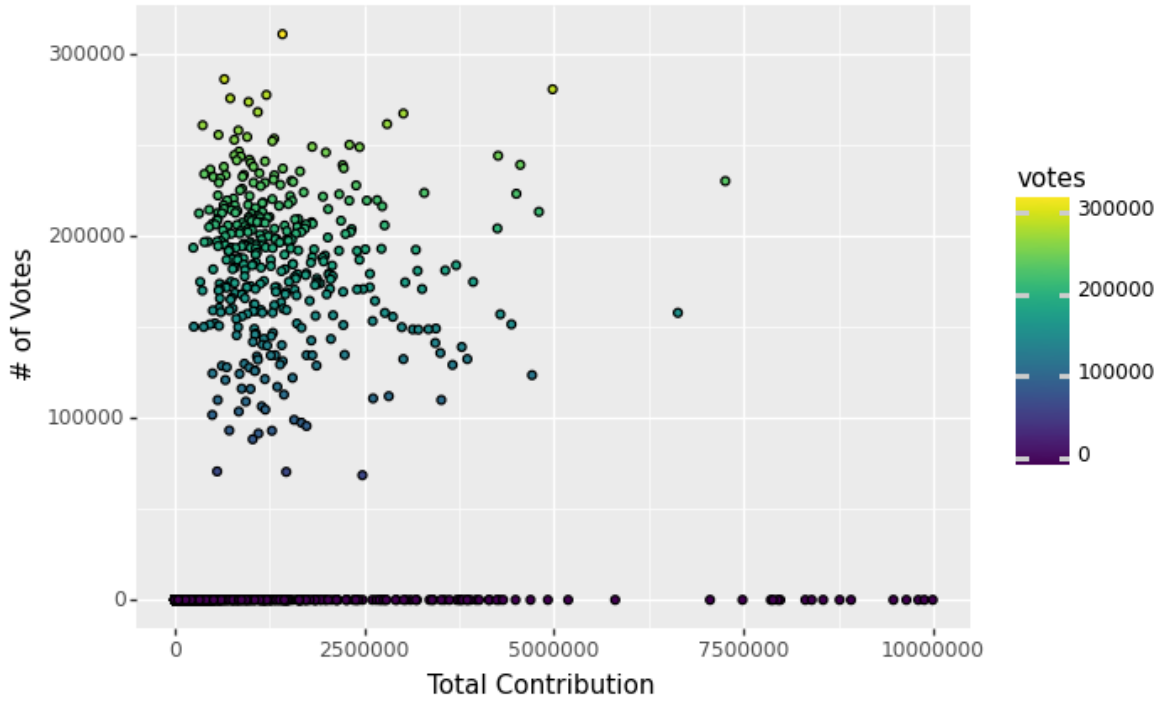
Out[14]:
```
-0.012339144256857892
```

In [15]:
```
true_vs_pred = pd.DataFrame({"predict": y_pred,"trueV": y_test})
true_vs_pred.head()
```

Out[15]:

	predict	trueV
1003	39009.140947	0.0
819	38995.849603	0.0
1489	38995.349329	0.0
1581	39025.493415	0.0
1557	38994.853652	0.0

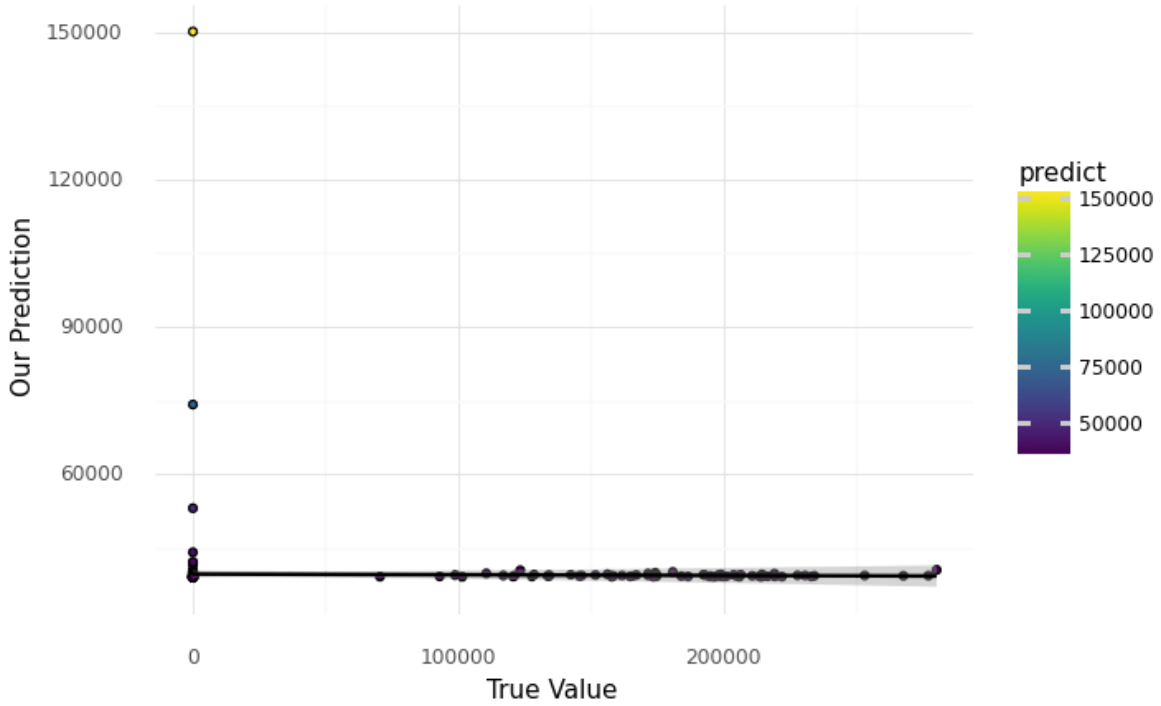
In [16]:
```
ggplot(election,aes(x = "tot_con", y = "votes", fill = "votes")) + geom_
point()+labs(x = "Total Contribution", y = "# of Votes") + xlim(0,100000
00)
```



Out[16]: <ggplot: (8736679832205)>

This graph shows the correlation between votes and total contribution is shown with a scatter plot

In [17]:
```
(ggplot(true_vs_pred, aes(x = "trueV", y = "predict", fill = "predict"))
+labs(x = "True Value", y = "Our Prediction") + geom_point()+ geom_smoot
h(method='lm')+ theme_minimal())
```



Out[17]: <ggplot: (-9223633300171216005)>

The graph above shows how we tested our regression by plotting our predicted votes against our true number

1. No, there is no linear relationship between votes and total contributions, as seen by our negative accuracy score. Additionally we graphed a scatter plot of total contribution and votes and there is not a linear relationship between the two. Our negative accuracy score infers that randomly guessing total votes based on total contribution would be better than this model. Although total contribution can be effective in predicting the winner, it is not for total votes. We believe this is due to the fact that every election is in a different location and it could be that some places just receive a lesser amount of votes even though it may have been enough to win.

In [ ]:
```
#3
```

In [18]:
```
predictors = ["ind_con", "can_con", "par_com_con"]

X_train, X_test, y_train, y_test = train_test_split(election[predictors
], election["winner"], test_size=0.2, random_state = 2)
X_train.head()

lsr = Lasso()
lsr.fit(X_train, y_train)

predictedVals_train = lsr.predict(X_train)
predictedVals = lsr.predict(X_test)

TRAIN_ACCURACY = accuracy_score(y_train,predictedVals_train.round())
TEST_ACCURACY = accuracy_score(y_test,predictedVals.round())


TRAIN_MAE = mean_absolute_error(y_train, lsr.predict(X_train))
TEST_MAE = mean_absolute_error(y_test, lsr.predict(X_test))

print("TRAIN_ACCURACY: ", TRAIN_ACCURACY)
print("TEST_ACCURACY: ", TEST_ACCURACY)
print("TRAIN_MAE: ", TRAIN_MAE)
print("TEST_MAE: ", TEST_MAE)
```

```
TRAIN_ACCURACY:  0.7401791867677464
TEST_ACCURACY:  0.7382920110192838
TRAIN_MAE:  0.3824219003598092
TEST_MAE:  0.3829326136377968
```

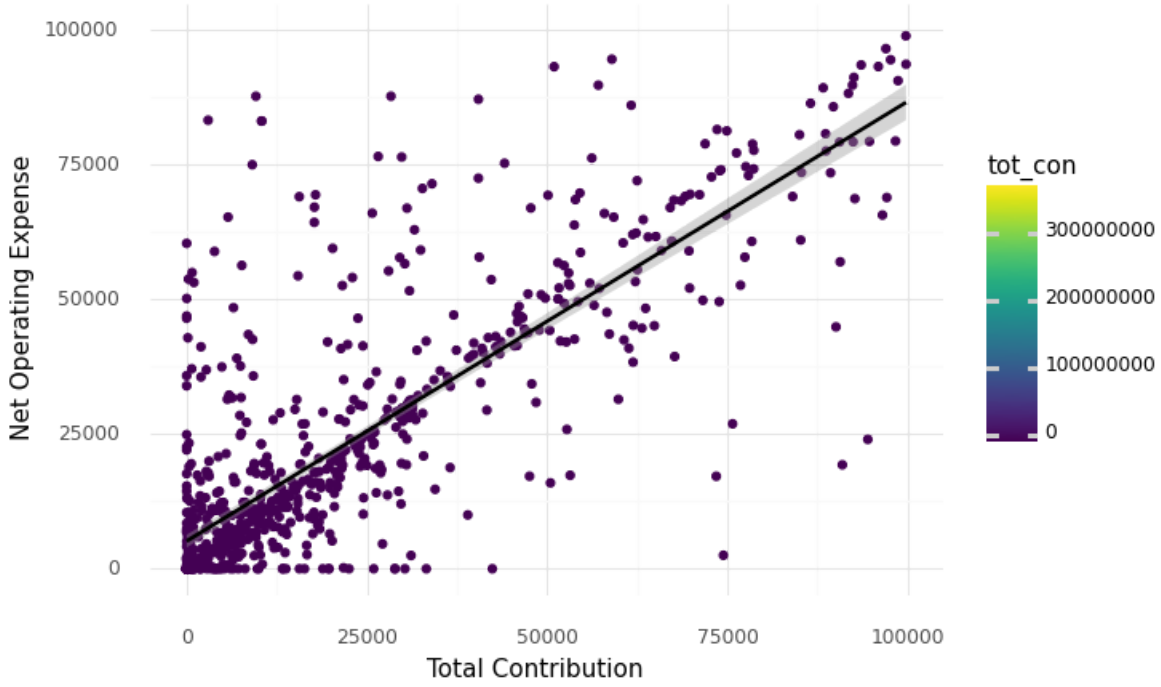
In [19]:
```
coef = pd.DataFrame({"Coefs": lsr.coef_, "Names": predictors})
coef
```

Out[19]:

	Coefs	Names
0	1.552046e-09	ind_con
1	-1.106291e-08	can_con
2	2.596352e-06	par_com_con

This is a dataframe of the coefficients of individual contribution, candidate contribution, and party committee contribution for our lasso regression of predicting the winner of an election

In [20]:
```
(ggplot(election, aes(x = "tot_con", y = "net_ope_exp", fill = "tot_con"
, color = "tot_con")) + geom_point()+ xlim(0,100000) + ylim(0, 100000) +
 geom_smooth(method='lm')+ theme_minimal() + labs(x = "Total Contributio
n", y = "Net Operating Expense"))
```



Out[20]: <ggplot: (-9223633300175067525)>

The graph above shows the relationship between net operating expenses for a campaign and the total contributions they receive

1. Aside from the total contribution, the party committee contributions affect the outcome of a candidate the most. The variables included were individual contributions, candidate contributions, party committee contributions. This was found by comparing the coefficients of all the contributions made to a candidates campaign. We decided to use Lasso as our dimensionality reduction because we wanted to include all of the variables given.

In [ ]:
```
#4 - How big of a role do net operating expenses play in a canditates su
cces or failure?
```

In [21]:
```
predictors = ["net_ope_exp"]

X_train, X_test, y_train, y_test = train_test_split(election[predictors
], election["winner"], test_size=0.2, random_state = 2)

myLogit3 = LogisticRegression(penalty = "none")
myLogit3.fit(X_train,y_train)

predictedVals_train = myLogit3.predict(X_train)
predictedVals = myLogit3.predict(X_test)

accuracy_score(y_test,predictedVals)
```

Out[21]: 0.7410468319559229

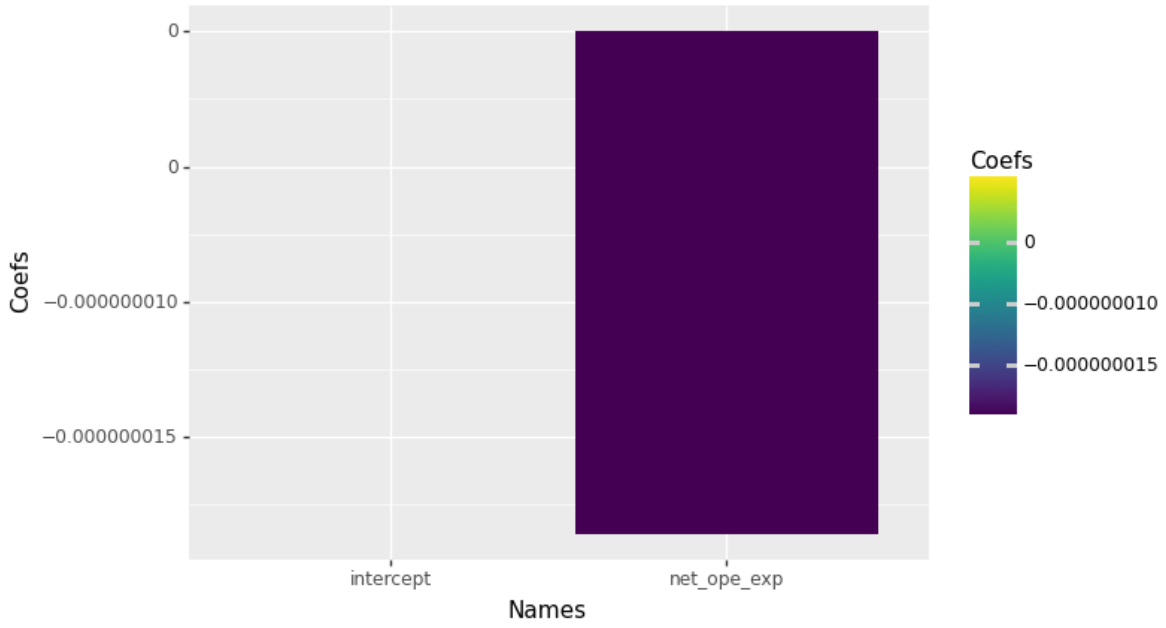
In [22]:
```
coef = pd.DataFrame({"Coefs": myLogit3.coef_[0], "Names": predictors})
coef = coef.append({"Coefs": myLogit3.intercept_[0], "Names": "intercep
t"}, ignore_index = True)
coef["Odds Coefs"] = np.exp(coef["Coefs"])
coef
```

Out[22]:

	Coefs	Names	Odds Coefs
0	-1.860084e-08	net_ope_exp	1.0
1	-2.717130e-13	intercept	1.0

This Data frame represents the effects of net operating expense on the winner of an election

In [23]:
```
(ggplot(coef,aes(x = "Names", y = "Coefs", fill = "Coefs"))+geom_bar(sta
t = "identity"))
```



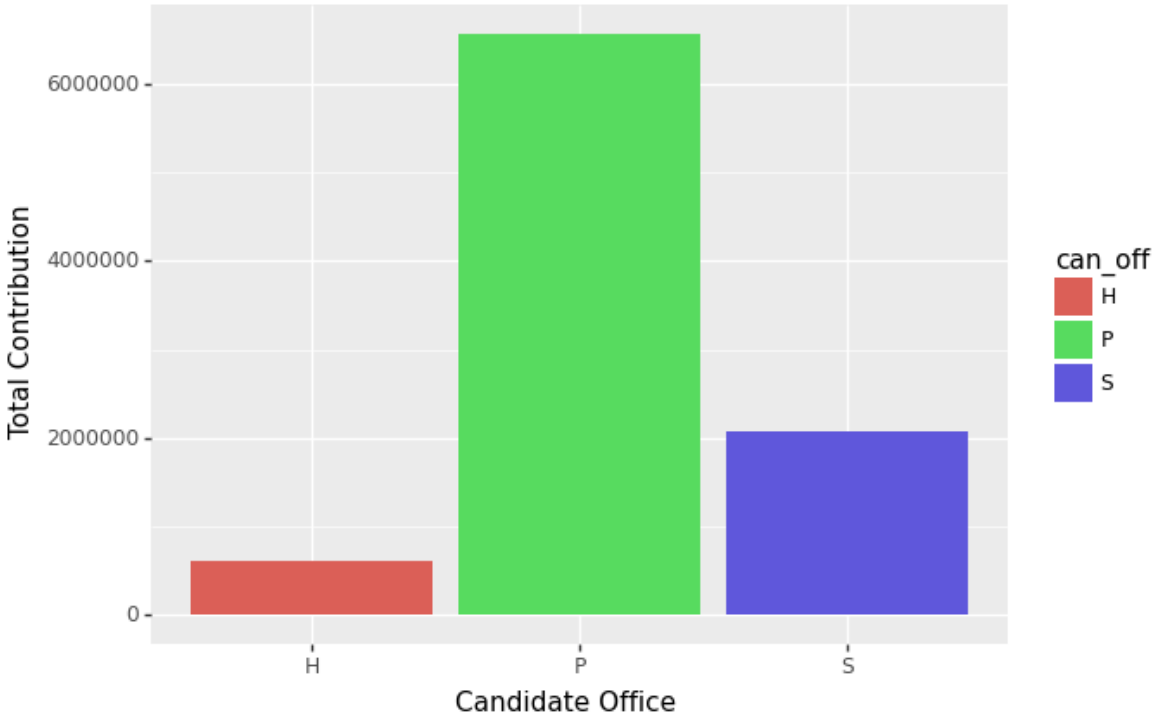
Out[23]: <ggplot: (8736680190507)>

This is a bar graph depicting our coefficients

1. As seen from the logistic regression, using net operating expenses to predict whether a candidate won an election is pretty accurate. When making predictions on the testing data set, the model was 74% accurate. Using the coefficients, as suggested, we found that every dollar spent on net operating expenses resulted in 0.000000186% decrease in becoming elected.This coefficient is significant considering many of these donations are in millions of dollars.

In [ ]:
```
#5 - Data Visualiztion to show the average total contributions to the ca
ndidates campaigns based on their respected office?
```

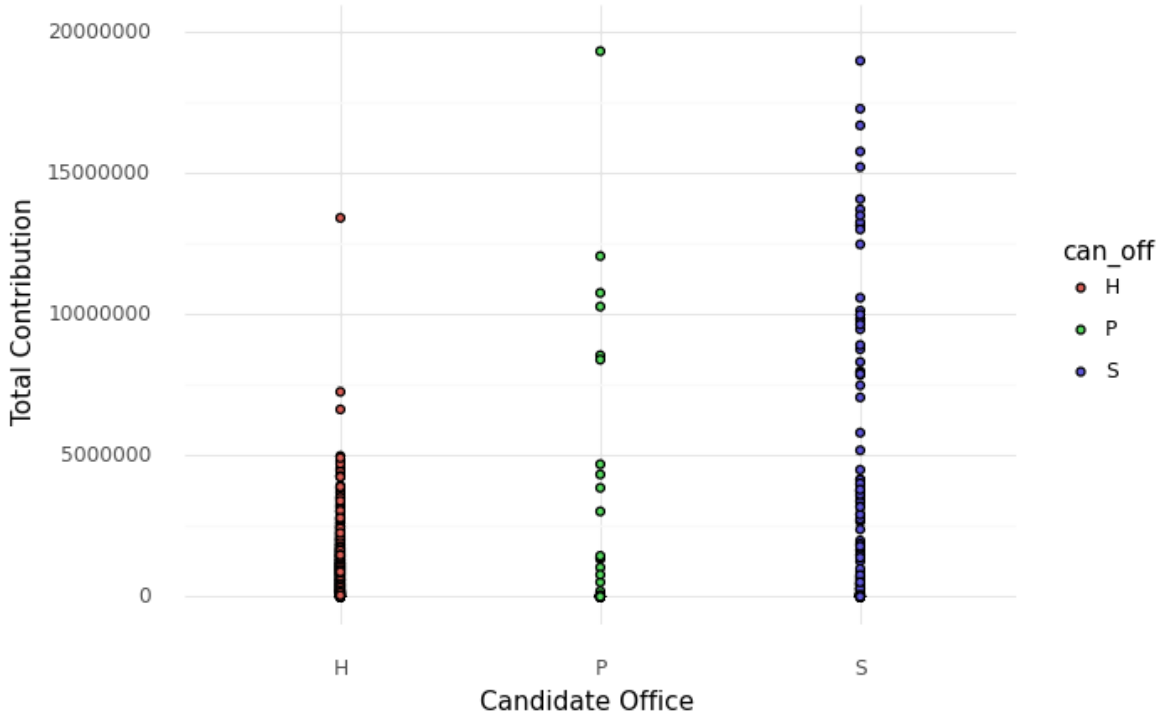
In [24]:
```
(ggplot(election, aes(x = "can_off", y = "tot_con", fill = "can_off")) +
stat_summary(fun_data = "mean_sdl", geom = "bar")+ labs(x = "Candidate O
ffice", y = "Total Contribution"))
```



Out[24]: <ggplot: (-9223633300177071500)>

The graph above groups the office a candidate is running with his peers and the height is determined by average total contributions to that candidates campaign

In [25]:
```
(ggplot(election, aes(x = "can_off", y = "tot_con", fill = "can_off"))+y
lim(0,20000000)+geom_point()+ theme_minimal() + labs(x = "Candidate Offi
ce", y = "Total Contribution"))
```



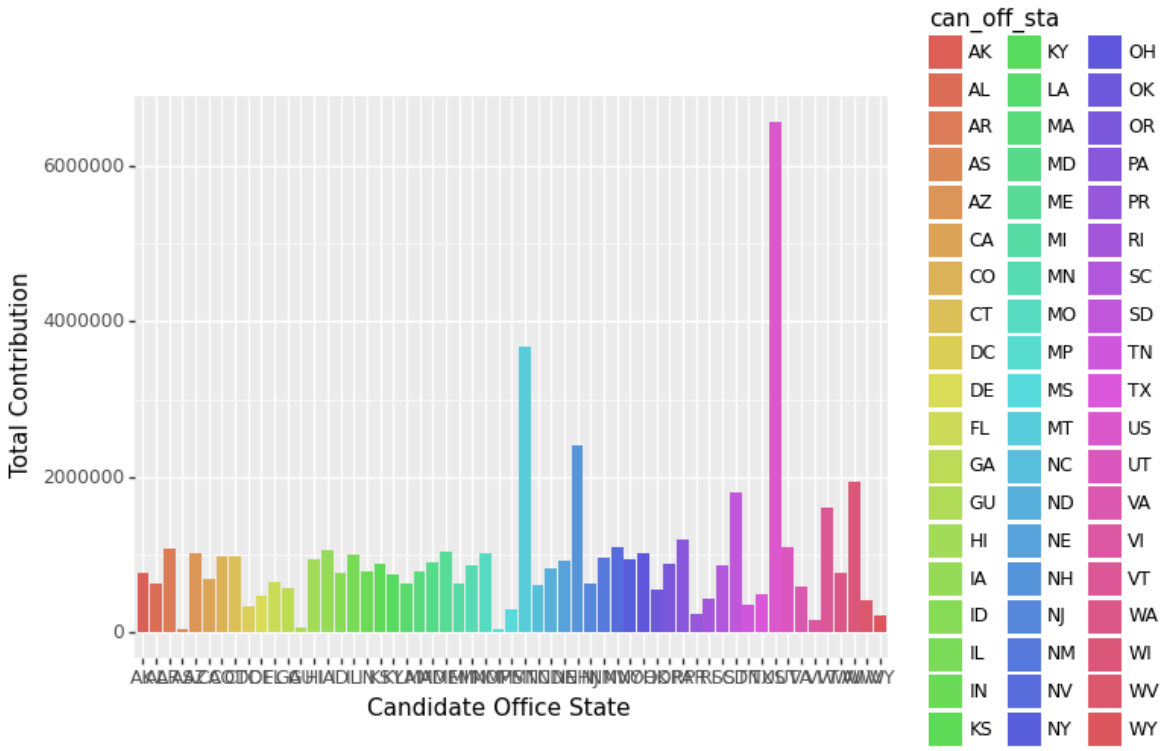
Out[25]: <ggplot: (8736677704726)>

The graph shows the variation in levels of total contribution grouped by candidate office. You can see the dispursement of total contribution for specific candidates in that respective office

1. The president gets significantly more total contributions on average compared to the senate, beating the senate out by a mean of 4 million dollars. The senate also edges out the house by well over a million, showing the order of the main 3 offices in the United States. Instead of using a clustering method, we would just show data visualizations to better interpret the average total contributions to the candidates campaigns based on their respected office.

In [ ]:
```
#6 - Do candidates in different states receive more funding from their p
arties? Data Viz
```

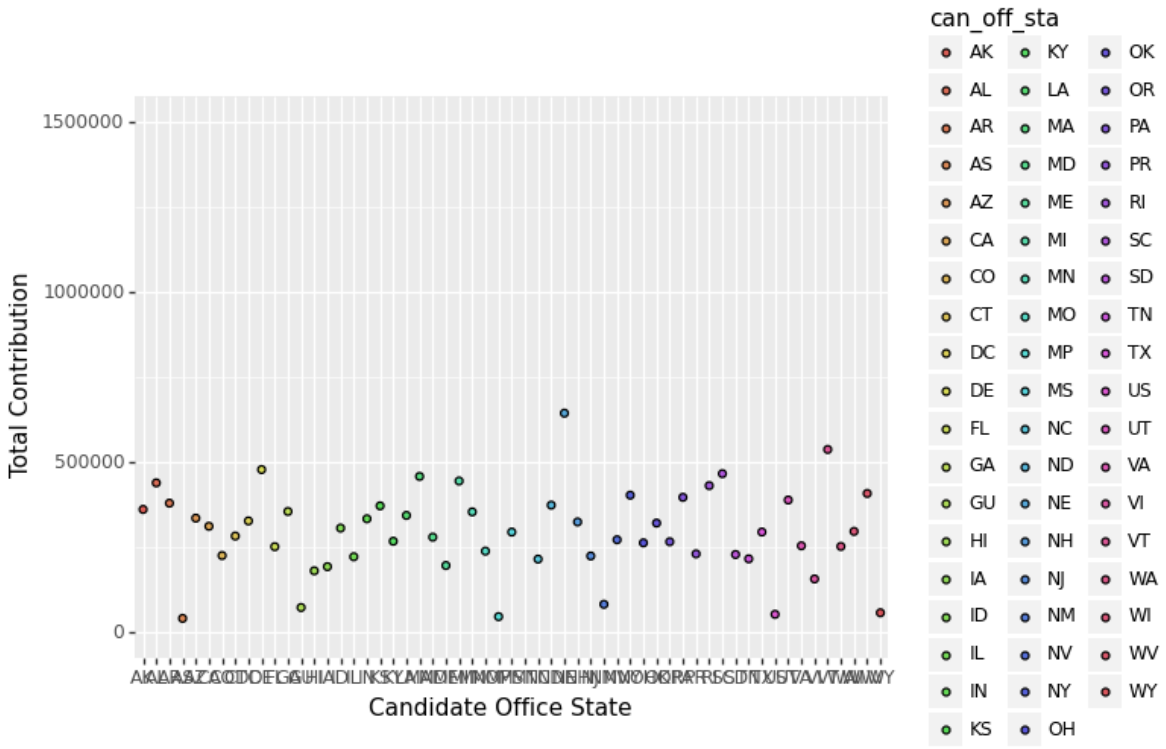
In [26]:
```
(ggplot(election, aes(x = "can_off_sta", y = "tot_con", fill = "can_off_
sta")) +
stat_summary(fun_data = "mean_sdl", geom = "bar") + labs(x = "Candidate
 Office State", y = "Total Contribution"))
```



Out[26]: <ggplot: (-9223633300171914138)>

This graph depicts the average total contributions a candidate get based on the state/territory they are from

In [27]:
```
(ggplot(election, aes(x = "can_off_sta", y = "tot_con", fill = "can_off_
sta")) + ylim(0,1500000) +
stat_summary(fun_data = "mean_sdl", geom = "point")+labs(x = "Candidate
 Office State", y = "Total Contribution"))
```



Out[27]: <ggplot: (-9223633300174579803)>

Here we used points to depict contributions based on state. Each point represents the average total contribution for that state

1. Candidates in different states do receive a significantly different amount of funding. Montana spends almost 4 million on average per candidate, and New Hampshire spends a bit over 2 million. Bigger states like California are where we begin to see some dropoff, as candidates from CA generate under a million, on average. This data set also included some of the United States territories, such as American Somoa which are typically the least donated to candidates. We decided instead of using a EM with mixtures of gaussian mixtures, we would just show data visualizations to better interpret the average each states candidate receives in funding.

In [28]:
```
#7
predictors = ["ind_con", "par_com_con"]

X_train, X_test, y_train, y_test = train_test_split(election[predictors
], election["winner"], test_size=0.2, random_state = 2)

myLogit4 = LogisticRegression(penalty = "none")
myLogit4.fit(X_train,y_train)

predictedVals_train = myLogit4.predict(X_train)
predictedVals = myLogit4.predict(X_test)

accuracy_score(y_test,predictedVals)
```

Out[28]: 0.6831955922865014

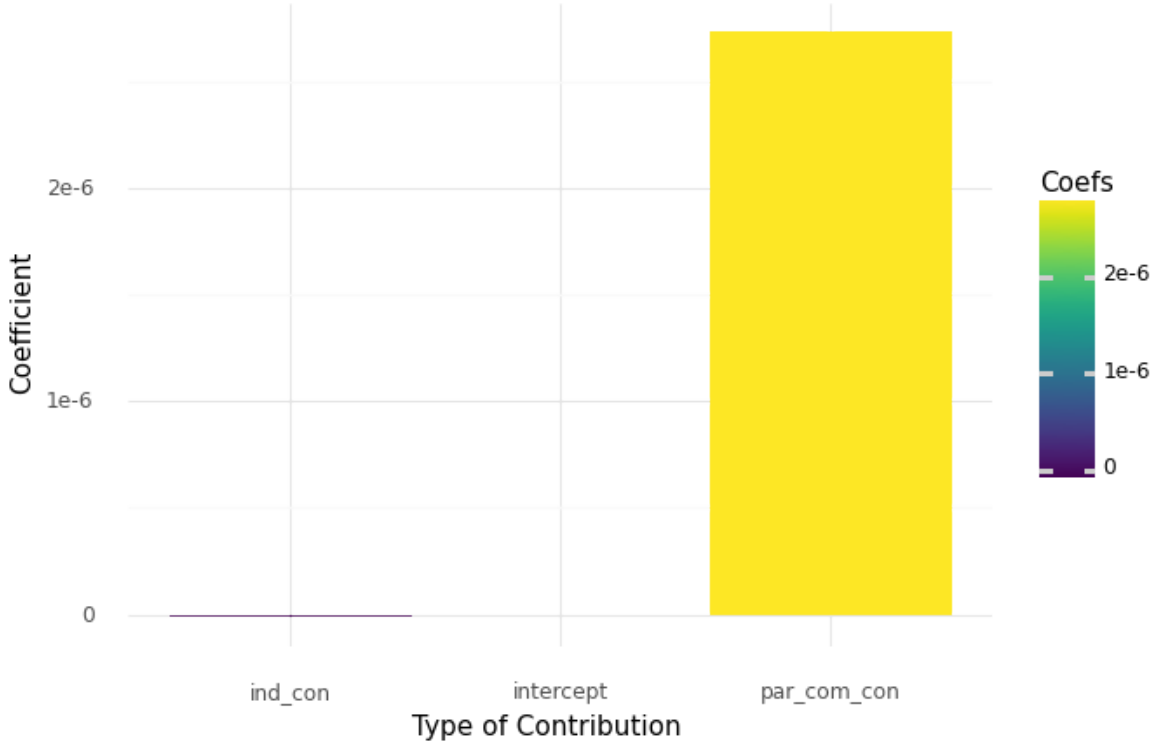
In [29]:
```
coef = pd.DataFrame({"Coefs": myLogit4.coef_[0], "Names": predictors})
coef = coef.append({"Coefs": myLogit4.intercept_[0], "Names": "intercep
t"}, ignore_index = True)
coef["Odds Coefs"] = np.exp(coef["Coefs"])
coef
```

Out[29]:

	Coefs	Names	Odds Coefs
0	-1.432802e-08	ind_con	1.000000
1	2.729950e-06	par_com_con	1.000003
2	-6.088053e-09	intercept	1.000000

This Dataframe shows the coefficients of individual contribution and party committy contribution and how greatly they effect the winner of an election

In [30]:
```
(ggplot(coef,aes(x = "Names", y = "Coefs", fill = "Coefs"))+geom_bar(sta
t = "identity")+theme_minimal()+labs(x = "Type of Contribution", y = "Co
efficient"))
```



Out[30]: <ggplot: (8736677403421)>

The graph above is to depict the coefficients and intercept of individual contributions and party committee contributions

1. As seen in the logistic regression, party contributions have much more of an effect on the outcome of an election compared to individual contributions. Instead of using the p-values we initially thought would be a good idea, we used the coefficients to give us more insight into the effect that these variables have on the election and lead us to our conclusion. We found that by using these variables, we can predict the outcome of the candidates in our test set correctly 68.3% of the time.

In [31]:
```
#8
```

In [32]:
```
predictors = ["tot_con", "ind_con", "can_con", "tot_loa", "par_com_con",
"net_ope_exp"]

X_train, X_test, y_train, y_test = train_test_split(election[predictors
], election["winner"], test_size=0.2, random_state = 2)
X_train.head()
```

Out[32]:

	tot_con	ind_con	can_con	tot_loa	par_com_con	net_ope_exp
1627	487056.97	464548.83	0.0	0.00	1758.14	468435.42
230	1173573.41	545053.39	0.0	0.00	0.00	791665.56
1502	100.00	100.00	0.0	0.00	0.00	4.59
1358	49734.06	49734.06	0.0	4902.75	0.00	111400.41
334	1812004.49	1051821.30	0.0	0.00	5000.00	1460930.13

In [33]:
```
myLogit = LogisticRegression(penalty = "none")
myLogit.fit(X_train,y_train)
```

Out[33]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='none',
                   random_state=None, solver='lbfgs', tol=0.0001, verbo
se=0,
                   warm_start=False)
```

In [34]:
```
predictedVals_train = myLogit.predict(X_train)
predictedVals = myLogit.predict(X_test)
```

In [35]:
```
print("TRAIN: ",accuracy_score(y_train,predictedVals_train))
print("TEST: ",accuracy_score(y_test,predictedVals))
```

```
TRAIN:  0.8917987594762233
TEST:  0.8512396694214877
```

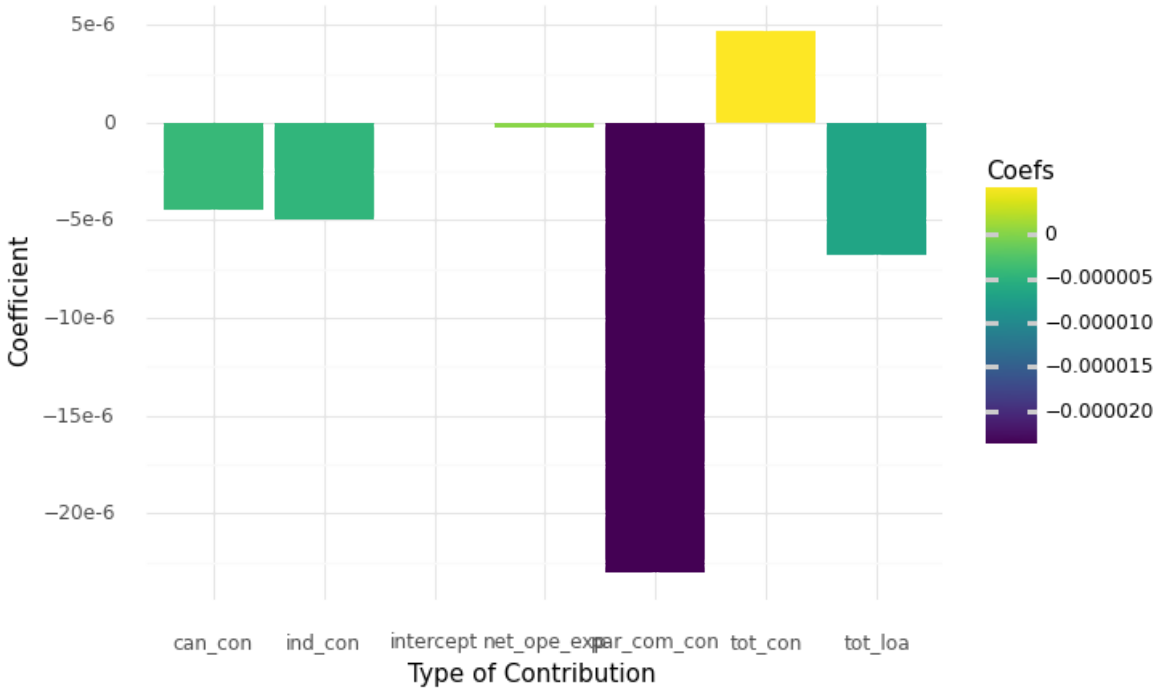
In [36]:
```
coef = pd.DataFrame({"Coefs": myLogit.coef_[0], "Names": predictors})
coef = coef.append({"Coefs": myLogit.intercept_[0], "Names": "intercept"
}, ignore_index = True)
coef["Odds Coefs"] = np.exp(coef["Coefs"])
coef
```

Out[36]:

	Coefs	Names	Odds Coefs
0	4.661790e-06	tot_con	1.000005
1	-4.940068e-06	ind_con	0.999995
2	-4.498483e-06	can_con	0.999996
3	-6.735840e-06	tot_loa	0.999993
4	-2.306216e-05	par_com_con	0.999977
5	-2.893570e-07	net_ope_exp	1.000000
6	-4.129954e-08	intercept	1.000000

This Data frame shows the coefficients of each of our predictors which represents their effect on predicting the winner of an election.

In [37]:
```
(ggplot(coef,aes(x = "Names", y = "Coefs", fill = "Coefs"))+geom_bar(sta
t = "identity")+theme_minimal()+labs(x = "Type of Contribution", y = "Co
efficient"))
```



Out[37]: <ggplot: (8736677636819)>

Above, we visualized the coefficients of the model in a bar graph.

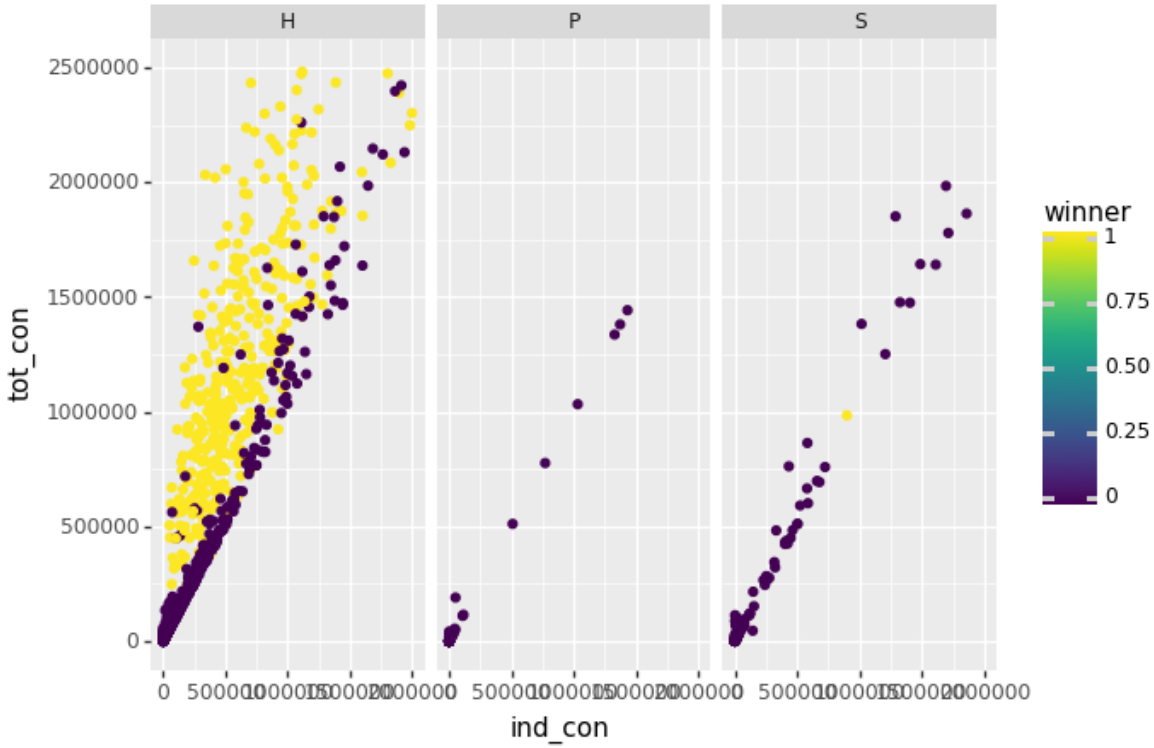
1. We made a logistic regression using all of the numeric predictors in our dataset and made predictions for the outcome of each candidate and found that it was very accurate. When making predictions for the training set we accurately predicted 89.2% of the time. When put up against the testing data set , which would show more accurately how effective the model will be when put up against other similar datasets, we found that we accurately predicted the candidates outcome 85.1% of the time.

In [ ]:
```
#9
```

In [38]:
```
predictors = ["tot_con", "ind_con", "can_con", "tot_loa", "par_com_con",
"net_ope_exp"]
X = election[predictors]
km = KMeans(n_clusters = 3)
km.fit(X)

office = km.predict(X)

X["cluster"] = office

silhouette_score(X, office)
```

Out[38]:
```
0.994116577841678
```

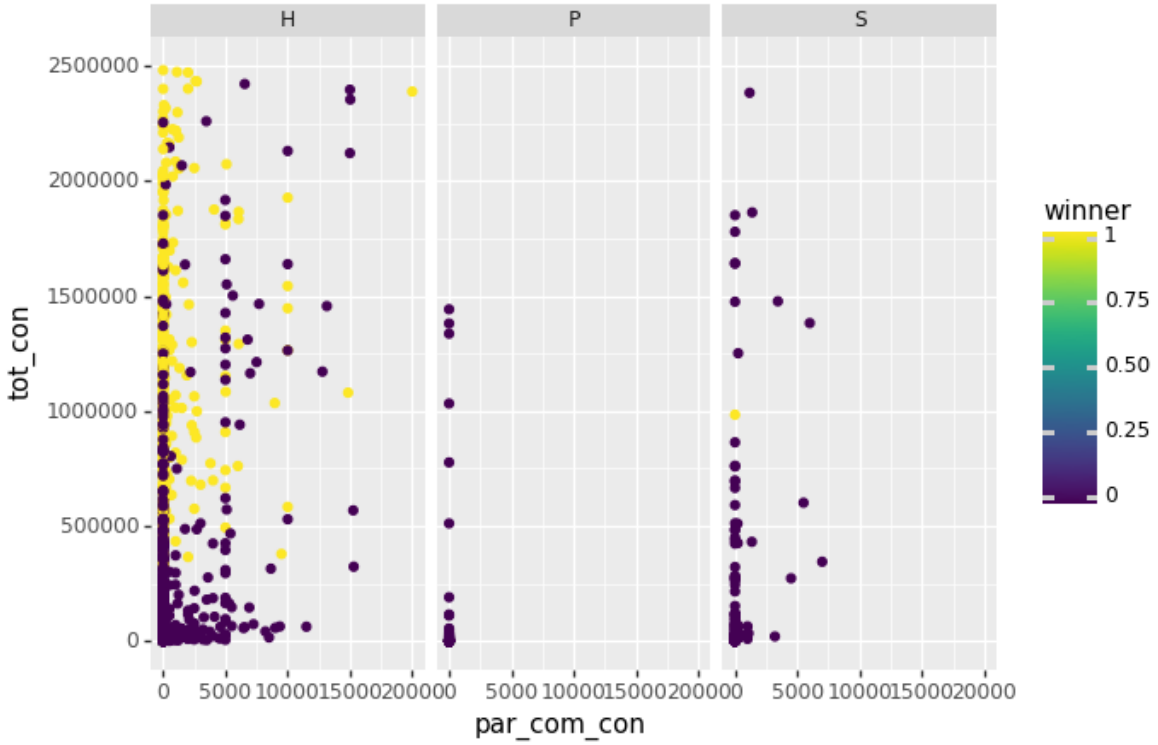
In [39]:
```
(ggplot(election, aes("ind_con", "tot_con", color = 'winner')) + geom_po
int()+
xlim(0,2000000)+ ylim(0, 2500000) +
 facet_wrap('can_off'))
```



Out[39]:
```
<ggplot: (8736677700893)>
```

Here we graph total contribution and individual contribution, going from house on the left, president in the middle, and senate on the right. Winners of said elections are in yellow while the losers are in purple

In [40]:
```
(ggplot(election, aes("par_com_con", "tot_con", color = 'winner')) + geo
m_point()+
xlim(0,20000)+ ylim(0, 2500000)+facet_wrap('can_off'))
```



Out[40]: <ggplot: (8736682849014)>

Above we graphed total contribution compared to party committee contribution, facet wrapped from house. President, and senate in order from left to right. Winners are in yellow and losers are in purple.

1. Instead of answering the initial question, we decided to use a clustering method with our numeric variables in order to find a deeper level of classification. Unfortunately, we couldn't properly cluster the party and individual contributions into a ggplot. However, we successfully made 2 graphs showing the difference between the number of individual donations compared to the total in the first graph and the number of party committee contributions in the second. We were shockingly surprised by the results of this set of graphs, as nearly no candidate wins without individual support, yet a significant majority of candidates win with minimal to none party support.