Final Project (100 points)

PLEASE READ ALL THE DIRECTIONS BEFORE STARTING THE PROJECT.

Part I

1.

Find a data set that's interesting to you. Make sure it has at least 7 variables and at least 100 rows, and at least 4 continus/interval columns. But the more the better. You may **NOT** use any of the datasets we've used in class (see here).

Some places to find data:

- data.gov
- kaggle.com/datasets
- your own data! (e.g. fitbit, data from a video game you play, etc...)
- https://github.com/BuzzFeedNews
- http://archive.ics.uci.edu/ml/index.php
- https://www.quandl.com/search
- http://academictorrents.com/browse.php
- your favorite sports teams!
- baseball data
- Fast Food and Food Data
- Data from your job/internship
- Scrape twitter data
- Tidy Tuesday Data
- fivethirtyeight
- League of Legends

2.

(**10 points**; Due Friday November 19th at 11:59pm; PDF) Pretend you're a company who is interested in the dataset you chose. Come up with at **least 7 questions** that you want to answer based on the variables (at the top of this document, **provide a short description of each of the variables in the model**).

These questions can be about the relationships between variables, or how well one thing can predict another, clustering...etc, but note that in your final project you must use at least **1 supervised learning model** (includes both regression and classification models), **1 clustering model, and 1 instance of dimensionality reduction** (PCA or LASSO), so keep that in mind when creating questions. You can use more than one of these for a single question (e.g. using PCA and then doing linear regression on the components).

You will be graded on the quality of the questions. Questions should be interesting and complex (e.g. questions like "is this model more than 90% accurate?" should be expanded to something like "is this model accurate as measured by accuracy, examination of patterns in the confusion matrix and/or consistent accuracy across gender/race/income/education groups?"). Questions related to the same model/analysis should be included as 1 question (for example, if you build a model predicting cat weight from cat height, cat age, and cat diet, the question should be somthing like "which variables have the strongest impact on cat weight?" instead of having three separate questions "what is the impact of cat height on cat weight?", "what is the impact of cat age on cat weight?", and ""what is the impact of cat diet on cat weight?")

3.

(**27 points**; Due Monday November 29th RIGHT BEFORE CLASS (if you do not have class on Monday it is due at 11:59pm); PDF) Now put on your data scientist hat. Write an **ORGANIZED analysis plan** to answer **3** of the questions you came up with. Think about which of the questions need a predictive model, which need a clustering model, which need dimensionality reduction, and which maybe need just visualizations/summaries. (at the top of this document, **provide a short description of each of the variables in the model**)

YOU MUST USE at least 1 supervised learning model, 1 clustering model and 1 instance of dimensionality reduction (two or more of these could be used to answer the same question). Dimensionality reduction includes LASSO and PCA.

Write up this plan as if you're submitting it to a company to tell them what you're planning to do. CLEARLY mark where each part (a-c) is and answer each part separately. For **each** question you need to:

- a) describe the analysis you're planning (include details like whether you're using standardization, regularization, model validation, distance/similarity metrics, how you'll choose clusters or hyperparameters, which variables you're using...etc)
- b) explain **why** this analysis and the choices you described above are good and explicitly **how** these methods will answer the question.
- c) describe **two** ggplot data visualizations you'll use to support your answers (graphs must be in ggplot, the ONLY acception is a dendrogram for HAC).

4.

(**5 points**; Due Friday December 3rd at 11:59pm) Peer review + write a critique of another person/group's plan (~ 1 page). You should answer:

- what does this plan do well?
- what could be improved (give specifics) and why?
- what are some (perhaps unavoidable) limitations of the data/analysis plan?

Part II

5.

(**38 points**; Due Monday December 13th at 11:59pm; PDF of Jupyter Notebook) **Perform the planned analyses** (be sure to note in markdown if there were any changes to your analysis plan since part #3 and **why**), and **make the graphs in a python notebook**. You should also include written (in markdown) answers to each of the 3 questions that you asked:

- a) **the analysis code.**
- b) **explicit answer to the question with detailed responses of how you came to this answer and the answer's importance.** This should be targeted at an audience that are NOT familiar with Data Science (e.g. pretend you're presenting these results to shareholders/your boss).
- c) **two ggplot data visualizations + captions** (graphs will be graded on how efficient and clear they are, so make sure you make good aesthetic choices that help emphasize your message).

Answers should be clear, concise, and complete. You will be graded on your code, the clarity of your responses, and the correctness of your methods. Save that notebook as a PDF. You must clearly label each question and the analyses that apply to it using Markdown.Don't forget to turn in a README with this part.

6.

(**8 points**; Due Sunday December 12th, 11:59pm; Video link) **Prepare a short presentation of the results** (powerpoint, prezi, keynote...etc, DO NOT just scroll through your notebook.). Make a short (5-9 minute, not under or over) **video presentation** explaining what you found. Upload it to youtube or similar site (you can put your video as Unlisted if you don't want anyone else to see)

I recommend OBS Streamlabs if you want to record your screen (with the presentation/data) and yourself presenting at one time. Or get someone to film you presenting it on a screen (or if needed print our your slides and hold them up!). Or if you're on a Mac you can use QuickTime to record your screen while you present.

7.

(**12 points**; During your scheduled Final) Watch other students' videos, discuss these questions:

- What are 2 things you enjoyed about their presentation?
- What is 1 thing they could be more clear about when presenting?
- What was the key idea you took away from their presentation?

and fill out the feedback form given to you by Chelsea.

Checklist

To review, throughout the project you'll need to submit:

1. the **name/a link to the data set** you're planning to use (you don't have to do this, but it saves you time/effort, in case your dataset won't work out)
2. A **PDF or text submission** with your questions.
3. A **PDF** with your analysis plan.
4. A **text submission or PDF** on Canvas with your critique of *another* students plan.
5. A **PDF** of your python notebooks. Please get rid of extra analyses/code that you did not end up using. You must clearly indicate where each question is being answered. Also include a **README**.
6. A **link** to a short video presentation (do not send the video directly).
7. A **text submission or PDF** on Canvas with your peer video feedback.

Group Option

Groups are optional (you may work alone) but if you choose to work in a group, READ THE MODIFICATIONS below, and know that working in a group is agreeing to **1)** follow these modifications and **2)** be graded together with your group members (you will not receive separate grades except for completing the peer review).

If you'd like to work in a group (up to 3) then:

- You must tell me your group members by **November 15th**.
- For Part I, #2 You need to come up with 5 questions **per member** and EACH of you must submit a copy of the same PDF to canvas.
- For Part I, #3 You should come up with the plan together and EACH submit a copy of the same plan. You need to answer 3 questions **per member** (so a group of 2 would need to answer 6 questions, a group of 3 would need to answer 9).
- For Part I, #4 You should write your critiques separately.
- For Part II, #5 You should work on it together (You can use Google Colab, it's like Google Docs for Jupyter notebooks, but without live updating[1]), and EACH submit a copy of the notebook PDF. Again, you need to answer 3 questions **per member** (so a group of 2 would need to answer 6 questions, a group of 3 would need to answer 9). **You must ALSO submit a joined statement (as a PDF) describing what each member did. Each of you should submit a copy of this.**
- For Part II, #6 You each need to make your OWN video presenting on 3 of the questions each, however you can collaborate on the presentation materials (the slides..etc). Each of you should submit a separate video.
- For Part II #7 You need to each do separate peer reviews.

[1] You won't be able to see the changes made live, but if you make changes, close the notebook, and your groupmate opens it on their computer, they will see your changes.

In [2]:
```
from plotnine import *
from sklearn.decomposition import PCA
import pandas as pd
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
import numpy as np
from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score #model evaluat
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
```

In [4]:

```
data = pd.read_csv("CPSC392_Final_Dataset.csv")
data = data.dropna()
data = data.drop(['Name'], axis = 1)
data.columns
```

Out[4]:

```
Index(['Salary', 'PA', 'R', 'H', '2B', '3B', 'HR', 'RBI', 'SO', 'BA', 'OBP',
       'SLG', 'OPS', 'OPS.1', 'TB'],
      dtype='object')
```

KAI's QUESTIONS

QUESTION 1: Choose and explain which clustering model would be best for clustering Homeruns vs. Doubles?

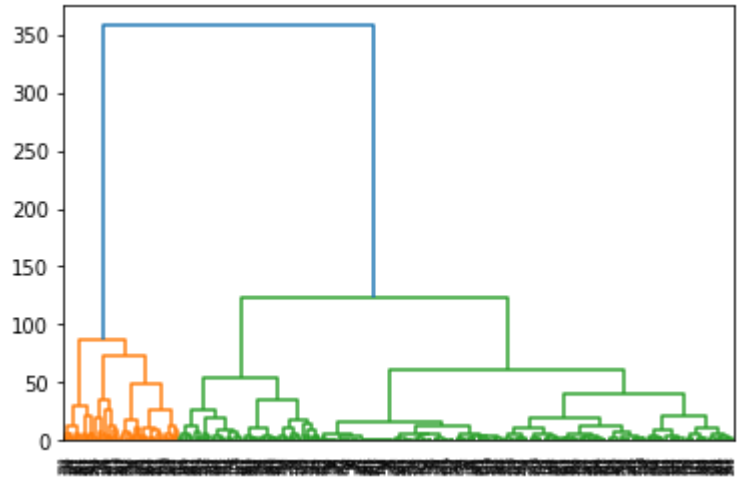
In [119…

```
z = StandardScaler()
data[["HR", "2B"]] = z.fit_transform(data[["HR", "2B"]])

(ggplot(data, aes(x = "HR", y = "2B")) + geom_point() + theme_minimal() + ggtitle("HR v
```

Out[119… <ggplot: (159992892025)>

In [115…

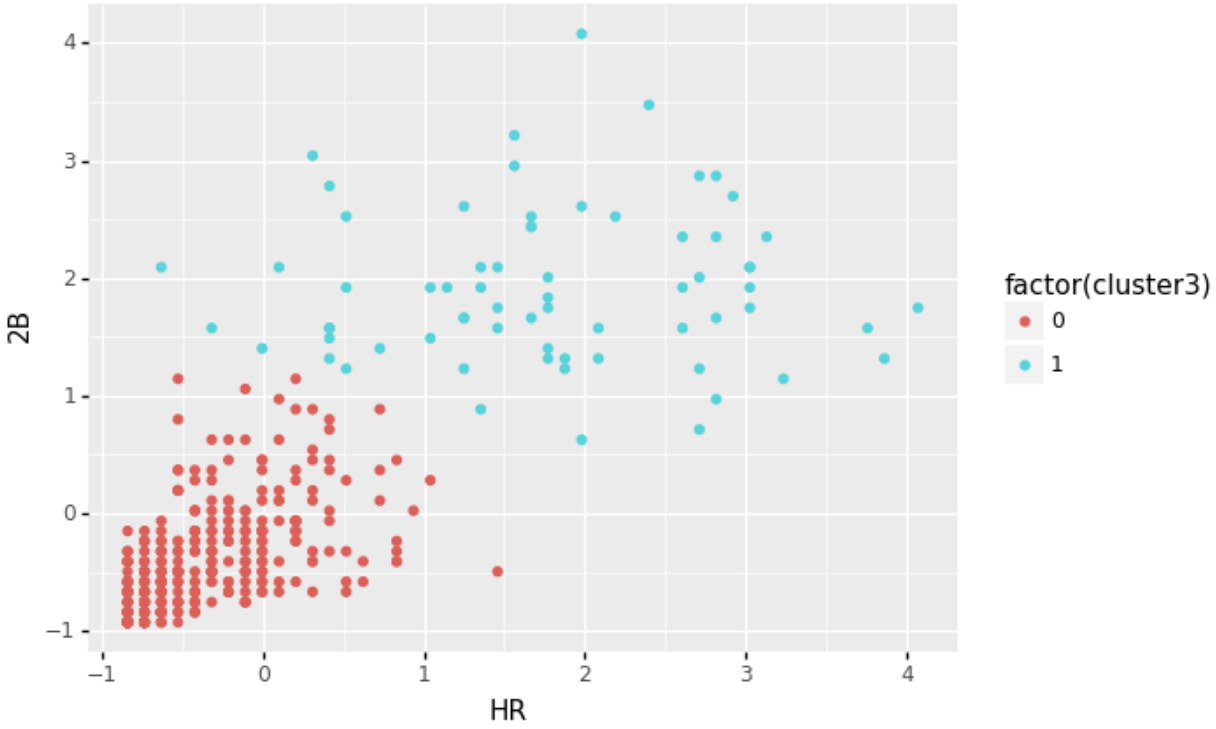
```
features = ["HR", "2B"]

z = StandardScaler()
X = data[features]

z.fit_transform(X)

hac = AgglomerativeClustering(affinity = "euclidean",
                              linkage = "ward")

hac.fit(X)
dendro = sch.dendrogram(sch.linkage(X, method='ward'))
```



In [116…
```
membership = hac.labels_

silhouette_score(X, membership)
```

Out[116…
```
0.7163967538586604
```

In [121…
```
data["cluster3"] = membership


(ggplot(data, aes(x = "HR", y = "2B")) + geom_point(aes(color = "factor(cluster3)")))
```



Out[121…
```
<ggplot: (159992893380)>
```

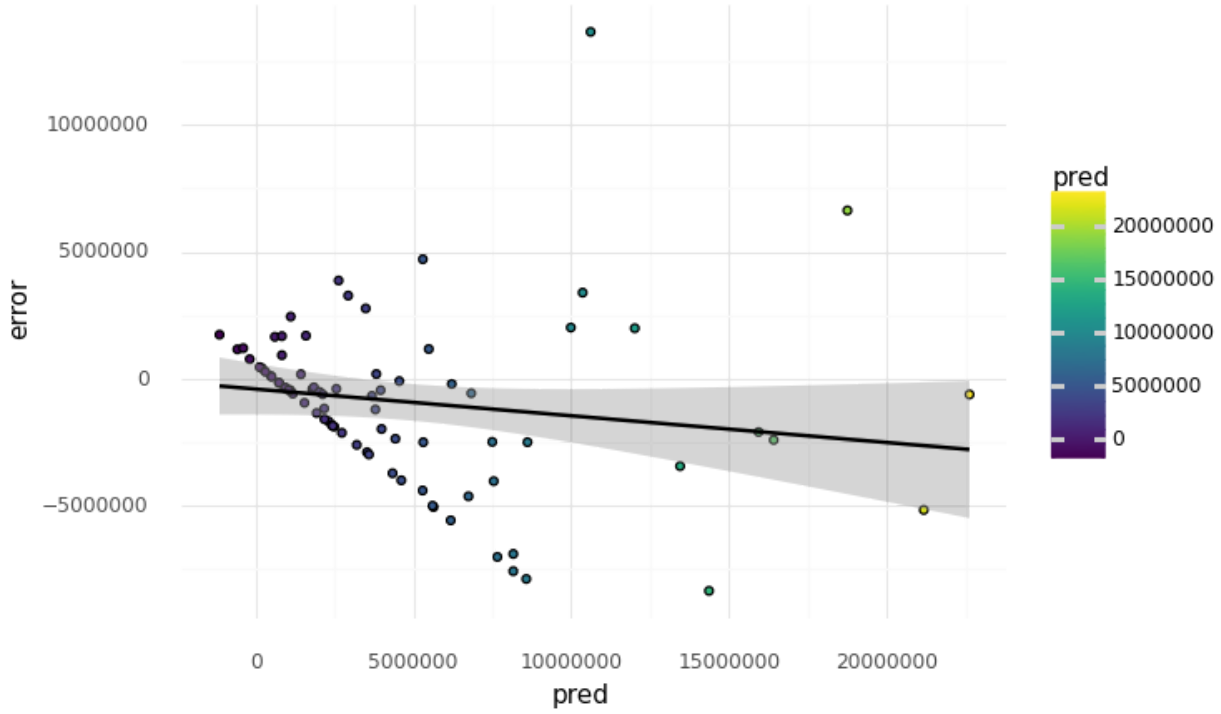
QUESTION 2: Create a regression model and determine if it's better to have all continuous variables in the model or to run regularization to take our insignificant variables?

In [126…

```
features = ['PA', 'R', 'H', '2B', '3B', 'HR', 'RBI', 'SO', 'BA', 'OBP',
       'SLG', 'OPS', 'OPS.1', 'TB']
X = data[features]
y = data["Salary"]

lr = LinearRegression()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

zscore = StandardScaler()
zscore.fit(X_train)
Xz_train = zscore.transform(X_train[features])
Xz_test = zscore.transform(X_test[features])

lr.fit(Xz_train, y_train)

y_pred = lr.predict(Xz_test)
```

In [45]:

```
Assump = pd.DataFrame({"error": y_test - y_pred, "pred": y_pred})
(ggplot(Assump, aes(x = "pred", y = "error", fill = "pred"))
 + geom_point() + theme_minimal() + stat_smooth(method = "lm"))
```



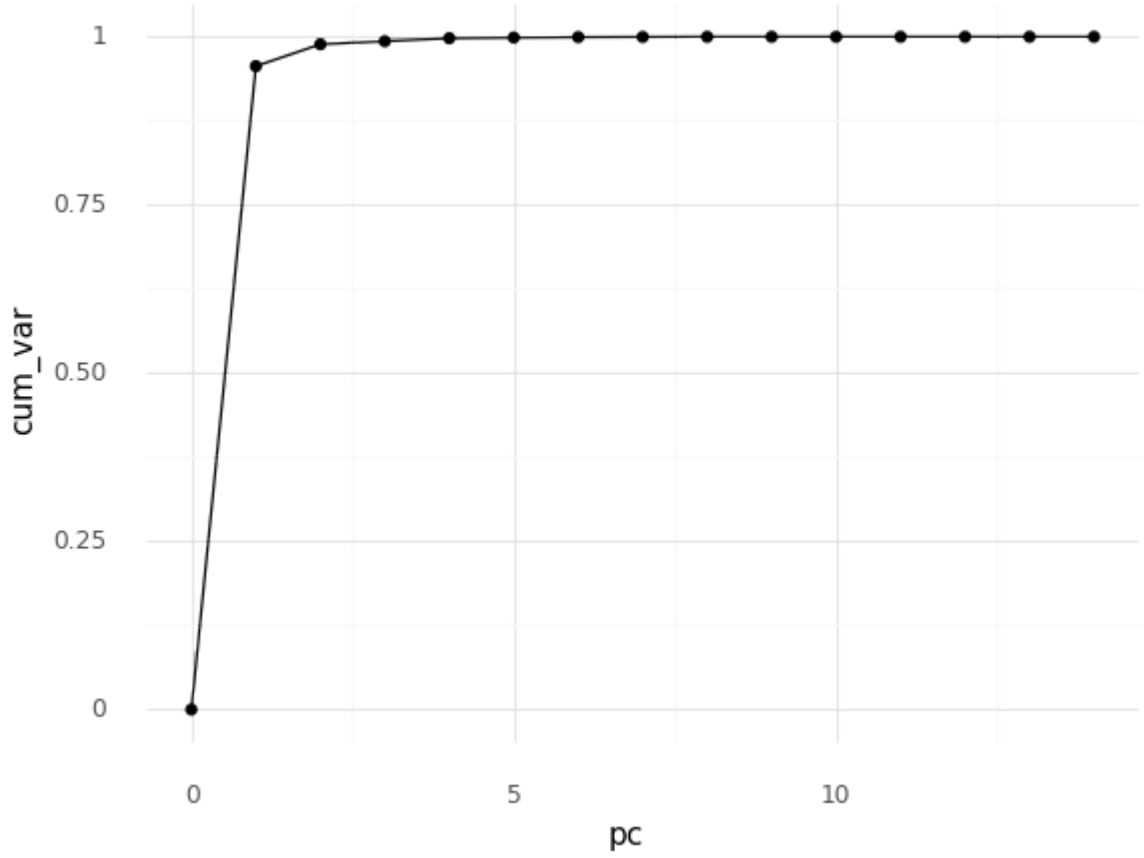
Out[45]: <ggplot: (159993488529)>

In [127…

```
print("R2 for test: ", r2_score(y_test, y_pred))
print("R2 for train: ", r2_score(y_train, lr.predict(Xz_train)))
print("MSE train: ", mean_squared_error(y_train, y_lr3))
print("MSE test: ", mean_squared_error(y_test, lr3.predict(test_3pc)))
```

```
R2 for test:  0.36904578435410196
R2 for train:  0.6896278933505967
MSE train:  85967490043559.67
MSE test:  52633960283478.336
```

In [8]:
```
pca = PCA()
pca.fit(X_train)

pcaDF = pd.DataFrame({"expl_var":pca.explained_variance_ratio_, "pc":range(1,15),
                      "cum_var":pca.explained_variance_ratio_.cumsum()})
pcaDF = pcaDF.append(pd.DataFrame({"expl_var" : [0], "pc": [0], "cum_var":[0]}))
(ggplot(pcaDF, aes(x = "pc", y = "cum_var")) + geom_line() + geom_point() + theme_minim
```



Out[8]: <ggplot: (159991256052)>

In [9]:
```
pcaDF
```

Out[9]:

	expl_var	pc	cum_var
0	9.563326e-01	1	0.956333
1	3.258543e-02	2	0.988918
2	4.502955e-03	3	0.993421
3	4.433030e-03	4	0.997854
4	9.695619e-04	5	0.998824
5	6.826370e-04	6	0.999506
6	2.603449e-04	7	0.999767
7	2.045683e-04	8	0.999971
8	2.884240e-05	9	1.000000
9	2.826093e-08	10	1.000000

	expl_var	pc	cum_var
10	1.207261e-08	11	1.000000
11	5.842535e-09	12	1.000000
12	1.124406e-12	13	1.000000
13	5.216733e-34	14	1.000000
0	0.000000e+00	0	0.000000

In [124…

```
zscore = StandardScaler()

pcs = pca.transform(X_train)
zscore.fit(X_train)
train_3pc = pd.DataFrame(pcs[:,0:3])

pcss = pca.transform(X_test)
zscore.fit(X_test)
test_3pc = pd.DataFrame(pcss[:,0:3])

lr3 = LinearRegression()

lr3.fit(train_3pc, y_train)
y_lr3 = lr3.predict(train_3pc)

print("R-Squared Scores:")

print("Train (3)", r2_score(y_train, y_lr3))
print("Test (3)", r2_score(y_test, lr3.predict(test_3pc)))
print("MSE:")
print("Train (3)", mean_squared_error(y_train, y_lr3))
print("Test (3)", mean_squared_error(y_test, lr3.predict(test_3pc)))
```

```
R-Squared Scores:
Train (3) 0.5808927737344354
Test (3) 0.5759544844276258
MSE:
Train (3) 21024355874156.316
Test (3) 22431575461728.355
```

QUESTION 3: What is more significant to a players salary, OPS+ or BA?

In [54]:

```
predictors = ['PA', 'R', 'H', '2B', '3B', 'HR', 'RBI', 'SO', 'BA', 'OBP',
       'SLG', 'OPS', 'OPS.1', 'TB']

X_train, X_test, y_train, y_test = train_test_split(data[predictors], data["Salary"], t

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

myLogit = LinearRegression()

myLogit.fit(Xz_train, y_train)
```

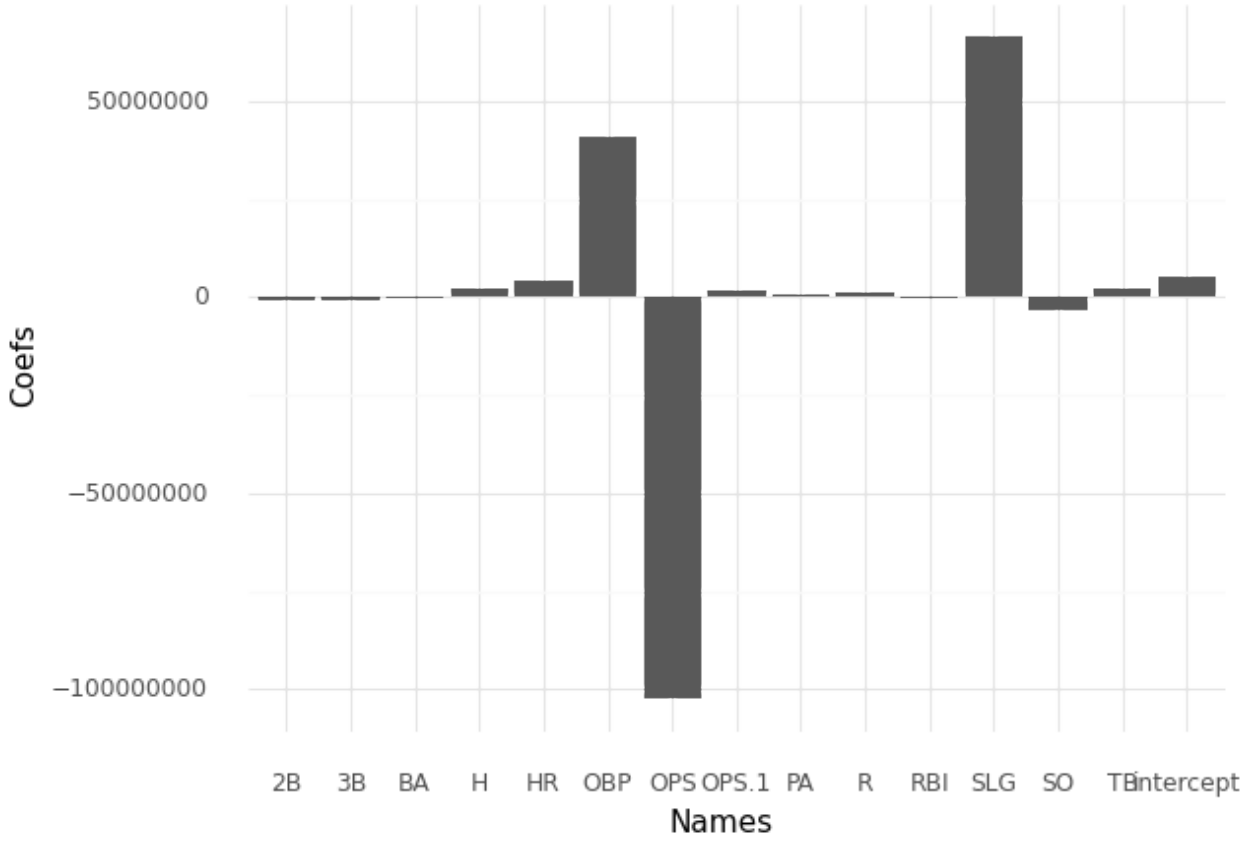
```
LinearRegression()
```

Out[54]:

In [57]:
```
y_pred = myLogit.predict(Xz_test)
print("R-Squared: ", r2_score(y_test, y_pred))
```

```
R-Squared:  0.6381767721701564
```

In [74]:
```
coef = pd.DataFrame({"Coefs": myLogit.coef_, "Names": predictors})
coef = coef.append({"Coefs": myLogit.intercept_, "Names": "intercept"}, ignore_index =

(ggplot(coef, aes(x = "Names", y = "Coefs")) + geom_bar(stat = "identity") + theme_mini
```



Out[74]: <ggplot: (159991811232)>

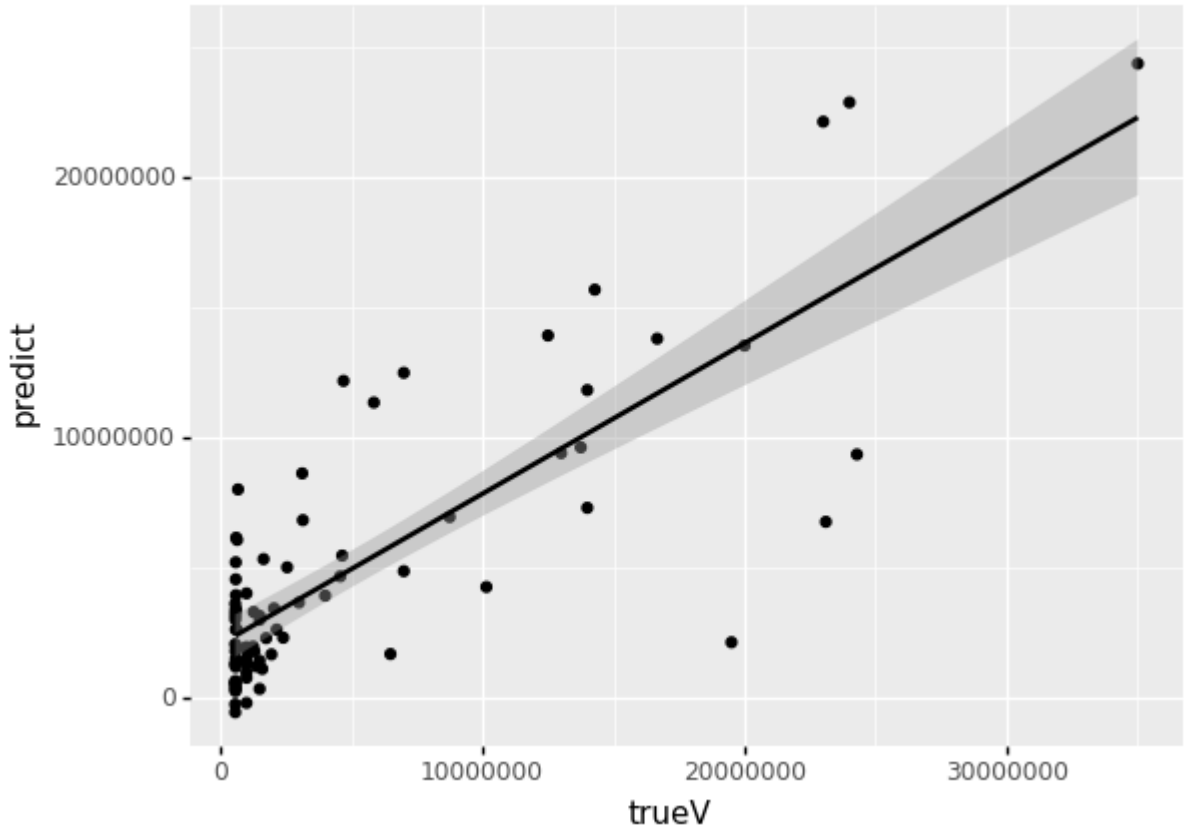
In [73]:
```
coef
```

Out[73]:

	Coefs	Names
0	6.947580e+05	PA
1	1.291301e+06	R
2	1.966931e+06	H
3	-7.232516e+05	2B
4	-9.596492e+05	3B
5	4.388041e+06	HR
6	-4.767047e+05	RBI

	Coefs	Names
7	-3.241229e+06	SO
8	-5.090476e+05	BA
9	4.084251e+07	OBP
10	6.629803e+07	SLG
11	-1.025088e+08	OPS
12	1.429337e+06	OPS.1
13	2.392617e+06	TB
14	5.175552e+06	intercept

In [76]:
```
true_vs_pred = pd.DataFrame({"predict": y_pred,"trueV": y_test})
(ggplot(true_vs_pred, aes(x = "trueV", y = "predict")) + geom_point() +stat_smooth(meth
```



Out[76]: <ggplot: (159992762263)>

In [77]:
```
true_vs_pred
```

Out[77]:

	predict	trueV
302	3.175019e+06	576700
445	6.930097e+06	8750000
355	7.282938e+06	14000000

	predict	trueV
250	2.449232e+05	578000
215	1.565910e+07	14285714
...	...	...
20	1.132582e+07	5857143
41	2.047460e+06	570500
293	1.915775e+06	650000
466	9.383361e+06	13000000
118	1.313485e+06	580500

80 rows × 2 columns

DARRON's QUESTIONS

In [ ]:
```
Question 1: Can we predict batting average with plate apperances and hits?
```

In [8]:
```
predictors = ["PA", "H", "RBI"]

X_train, X_test, y_train, y_test = train_test_split(data[predictors], data["BA"], test_

zscore = StandardScaler()
zscore.fit(X_train)
Xz_train = zscore.transform(X_train)
Xz_test = zscore.transform(X_test)
data[predictors] = zscore.fit_transform(data[predictors])

model = LinearRegression()
model.fit(Xz_train, y_train)
```

Out[8]:
```
LinearRegression()
```

In [ ]:
```
y_pred = model.predict(Xz_test)


print("Training R2 score is: ", model.score(Xz_train, y_train))
print("Testing R2 score is: ",model.score(Xz_test, y_test))

print("Training MSE is: ",mean_squared_error(y_train, model.predict(Xz_train)))
print("Testing MSE is: ",mean_squared_error(y_test, model.predict(Xz_test)))
```

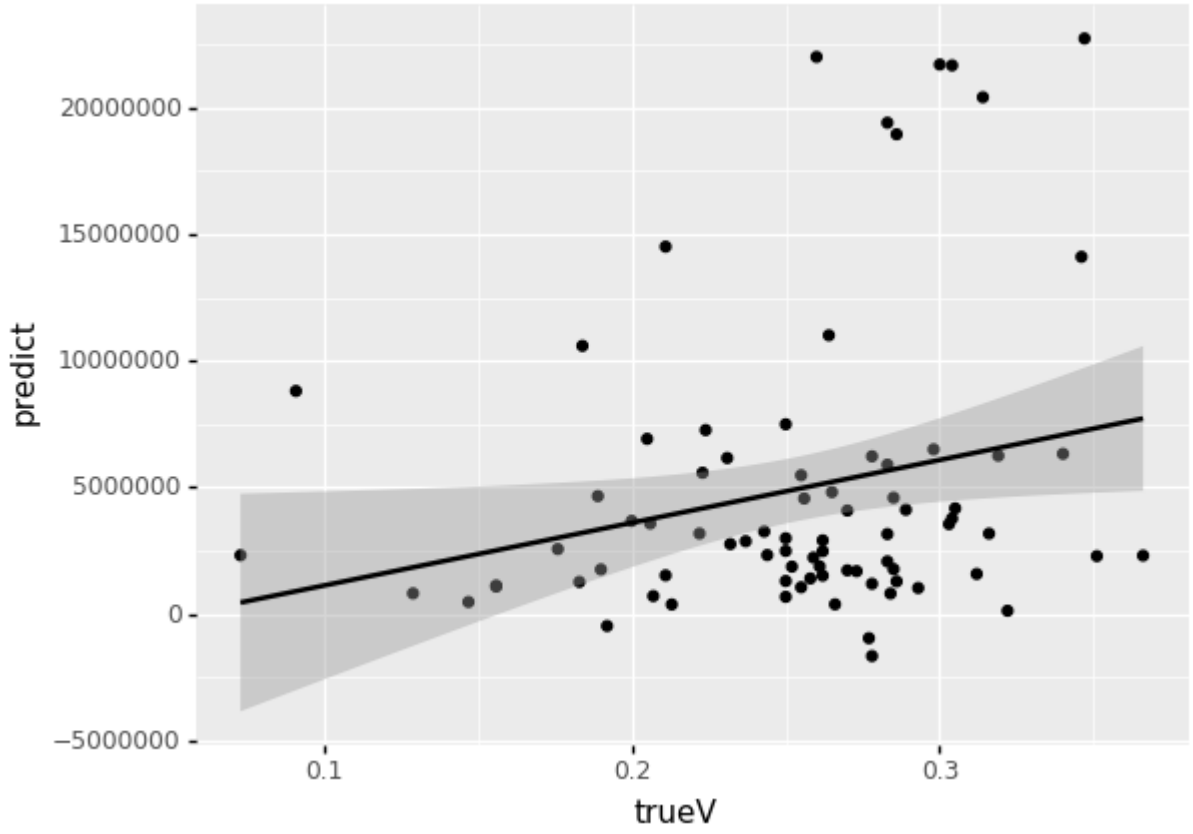
In [9]:
```
true_vs_pred = pd.DataFrame({"predict": y_pred,"trueV": y_test})
true_vs_pred.head()
```

Out[9]:

	predict	trueV
26	7.493376e+06	0.250

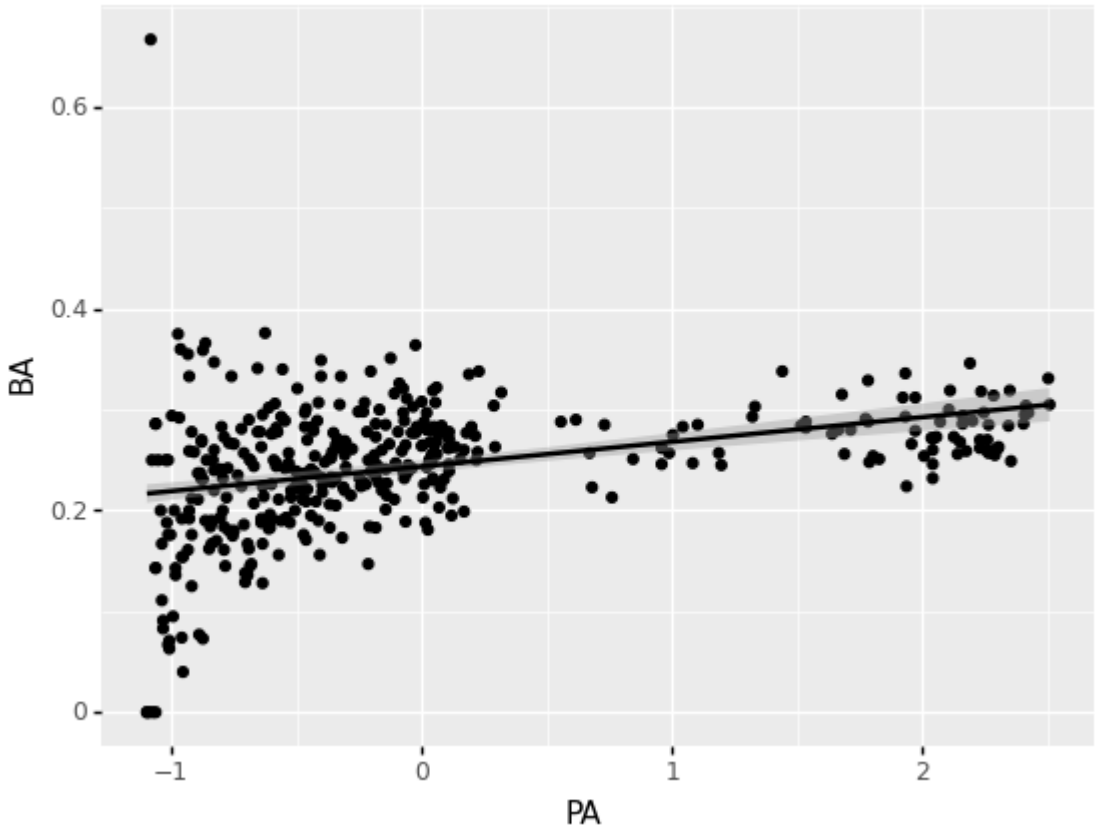
	predict	trueV
32	1.687036e+06	0.273
185	2.302522e+06	0.366
466	3.546363e+06	0.303
276	4.684856e+05	0.147

In [10]:
```
(ggplot(true_vs_pred, aes(x = "trueV", y = "predict")) + geom_point() +
stat_smooth(method = "lm"))
```



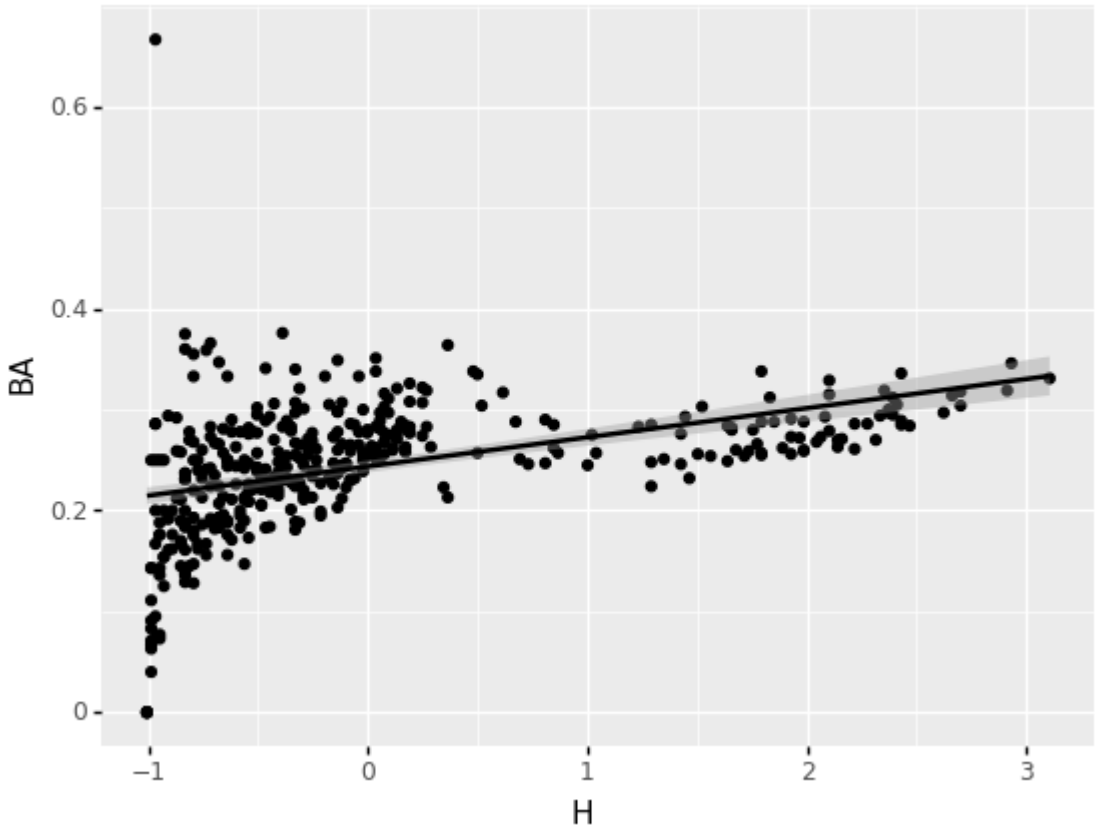
Out[10]: <ggplot: (97177442672)>

In [11]:
```
(ggplot(data, aes(x = "PA", y = "BA"))+ geom_point() + stat_smooth(method = "lm"))
```



Out[11]: <ggplot: (97177528507)>

In [12]:
```
(ggplot(data, aes(x = "H", y = "BA"))+ geom_point() + stat_smooth(method = "lm"))
```



Out[12]: <ggplot: (97177585553)>

QUESTION 2: Can we predict a players salary using RBI's and Batting Average?

In [14]:
```
predictors2 = ["RBI", "BA"]

predictors2 = list(filter(None,predictors2))
X_train, X_test, y_train, y_test = train_test_split(data[predictors2], data["Salary"],
zscore = StandardScaler()
zscore.fit(X_train)
Xzz_train = zscore.transform(X_train)
Xzz_test = zscore.transform(X_test)
data[predictors2] = zscore.fit_transform(data[predictors2])
```

In [15]:
```
model2 = LinearRegression()
model2.fit(Xzz_train, y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
y_pred2 = model.predict(Xz_test)


print("Training R2 score is: ", model.score(Xz_train, y_train))
print("Testing R2 score is: ",model.score(Xz_test, y_test))

print("Training MSE is: ",mean_squared_error(y_train, model.predict(Xz_train)))
print("Testing MSE is: ",mean_squared_error(y_test, model.predict(Xz_test)))
```

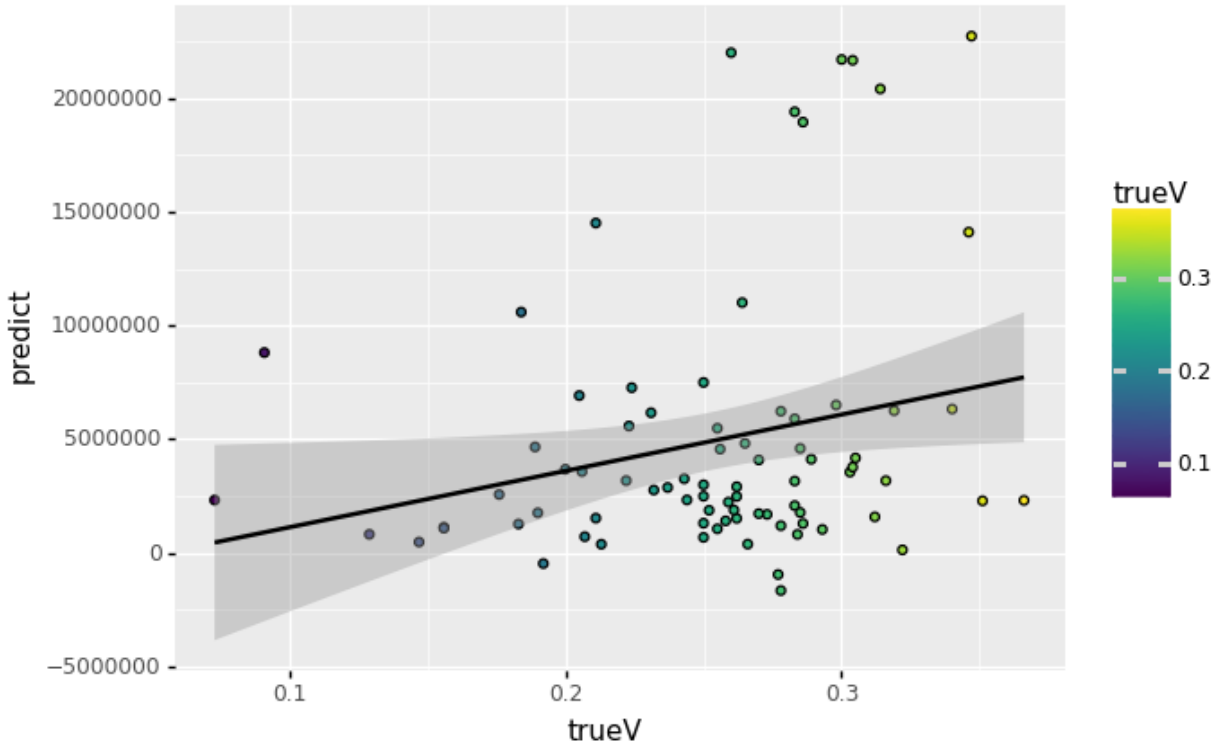
```
Training R2 score is:  -0.5501777875878626
Testing R2 score is:  -0.39754145652227457
Training MSE is:  80371700074221.23
Testing MSE is:  63391852760439.64
```

In [17]:
```
true_vs_pred2 = pd.DataFrame({"predict": y_pred2,"trueV": y_test})
true_vs_pred2.head()
```

Out[17]:

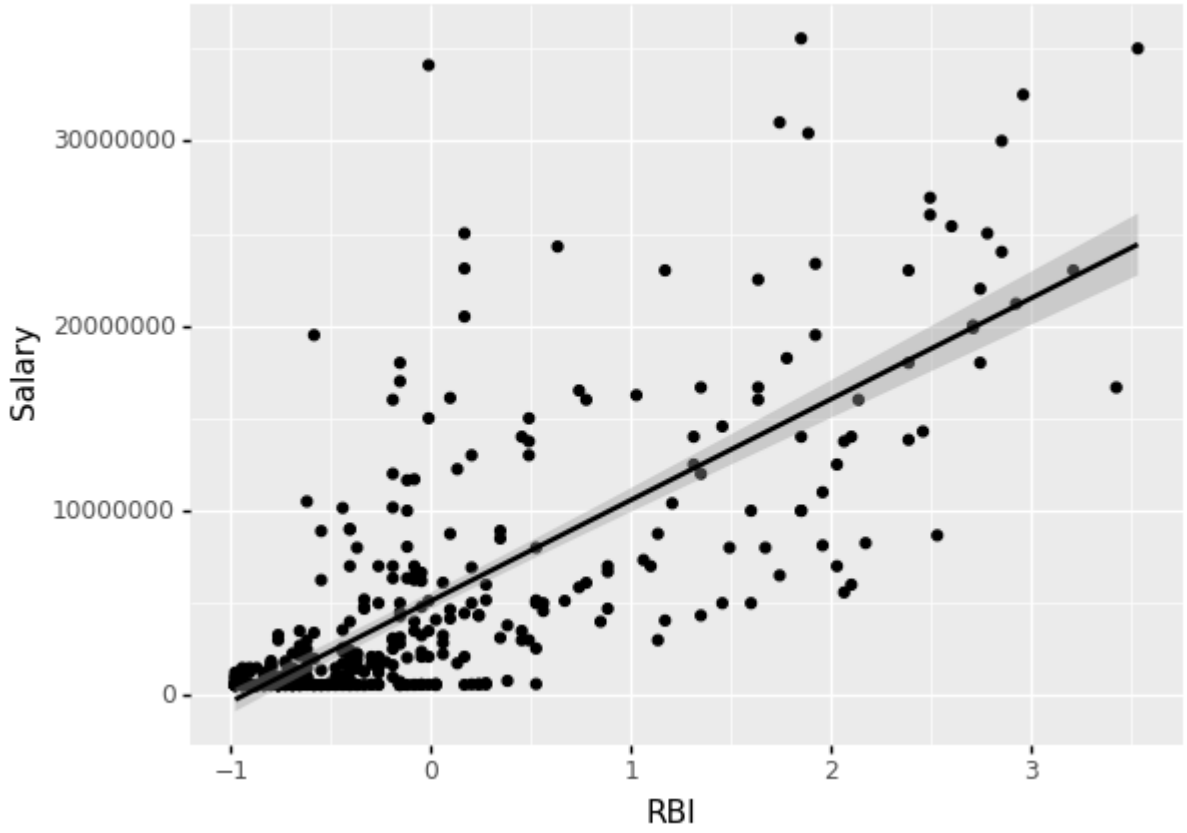
	predict	trueV
335	0.226650	1500000
449	0.230879	1000000
300	0.243207	584000
297	0.261117	591500
139	0.181568	25384615

In [18]:
```
(ggplot(true_vs_pred, aes(x = "trueV", y = "predict")) + geom_point(aes(fill = "trueV")
stat_smooth(method = "lm"))
```



Out[18]: <ggplot: (97177807674)>

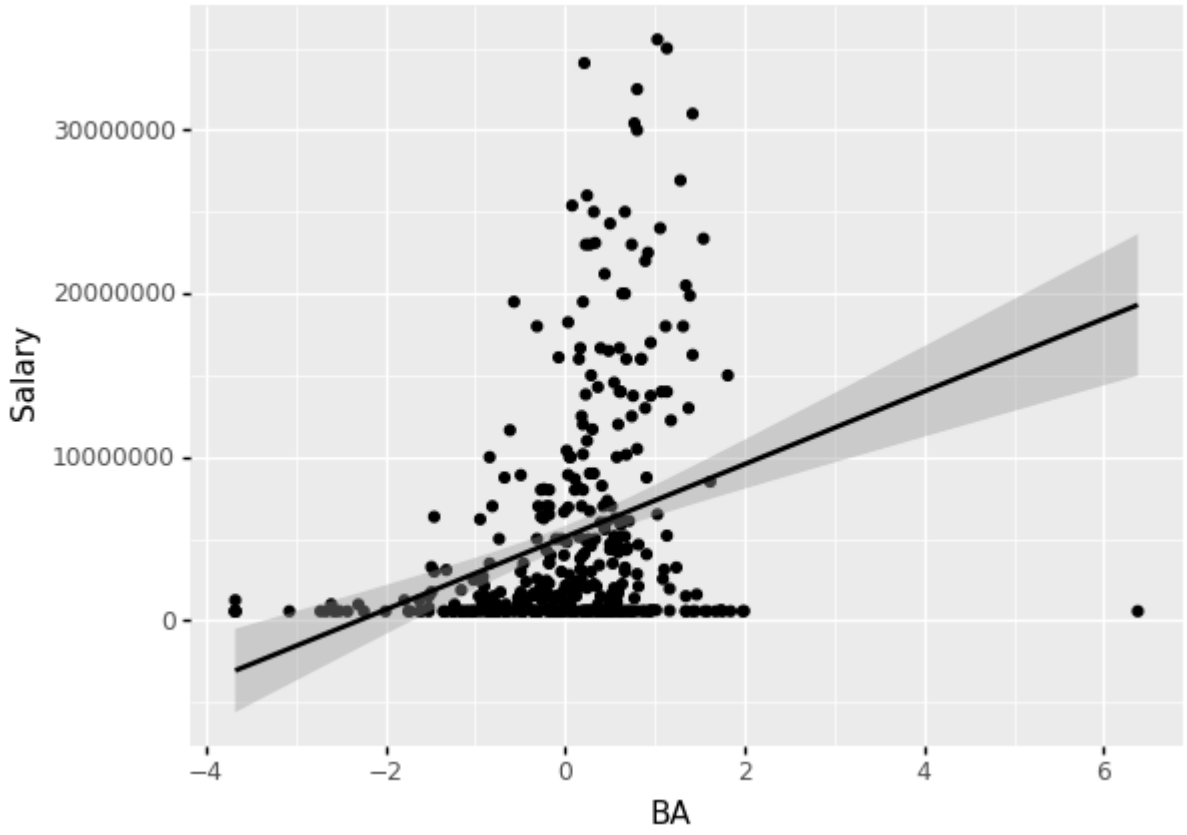
In [20]:
```
(ggplot(data, aes(x = "RBI", y = "Salary"))+ geom_point() + stat_smooth(method = "lm"))
```



Out[20]: <ggplot: (97179011940)>

In [21]:

```
(ggplot(data, aes(x = "BA", y = "Salary"))+ geom_point() + stat_smooth(method = "lm"))
```

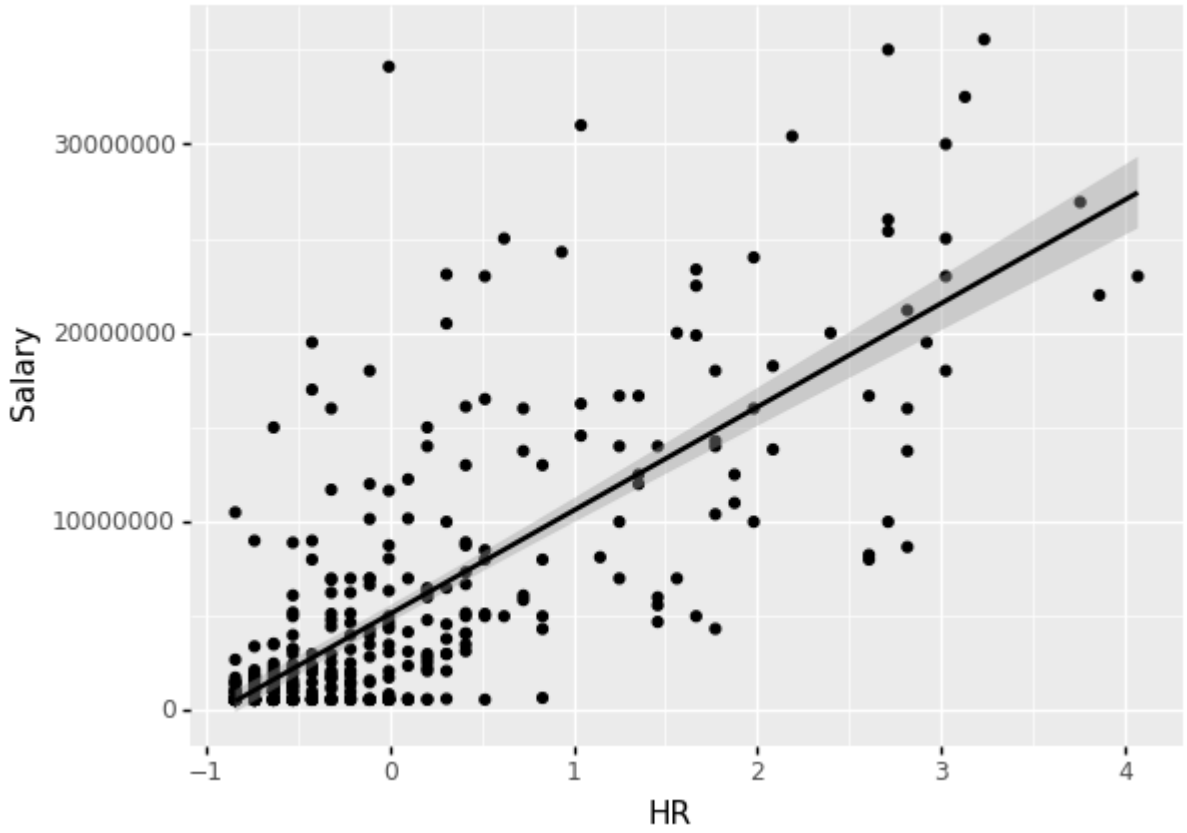


Out[21]: <ggplot: (97179083882)>

QUESTION 3: Can the high variance components predict salary?

In [22]:
```
features = ["PA","R","H","2B", "3B", "HR","RBI","SO","BA","OBP","SLG","OPS","TB"]
data[features] = zscore.fit_transform(data[features])
```

In [24]:
```
ggplot(data, aes(x = "HR", y = "Salary")) + geom_point() +stat_smooth(method = "lm")
```



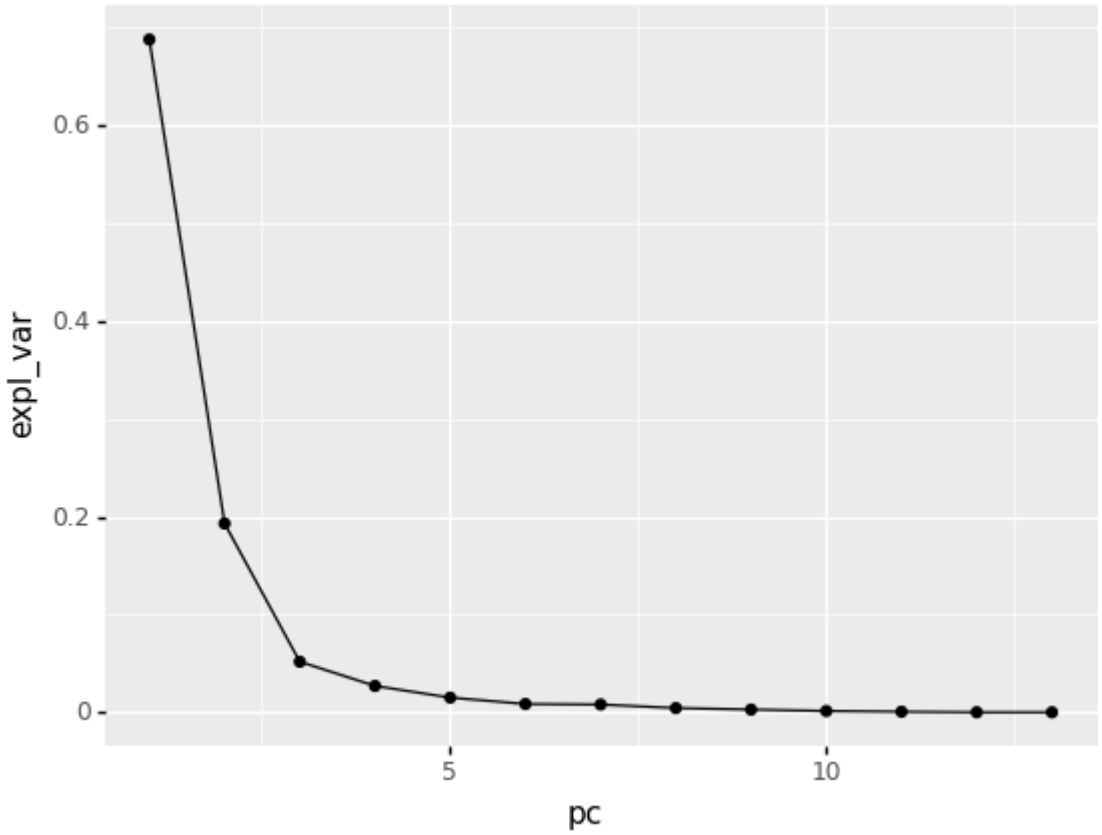
Out[24]: <ggplot: (97179193128)>

In [26]:
```
pca = PCA()
pca.fit(data[features])
pcaDF = pd.DataFrame({"expl_var" : pca.explained_variance_ratio_, "pc": range(1,14), "c
pcaDF.head()
```

Out[26]:

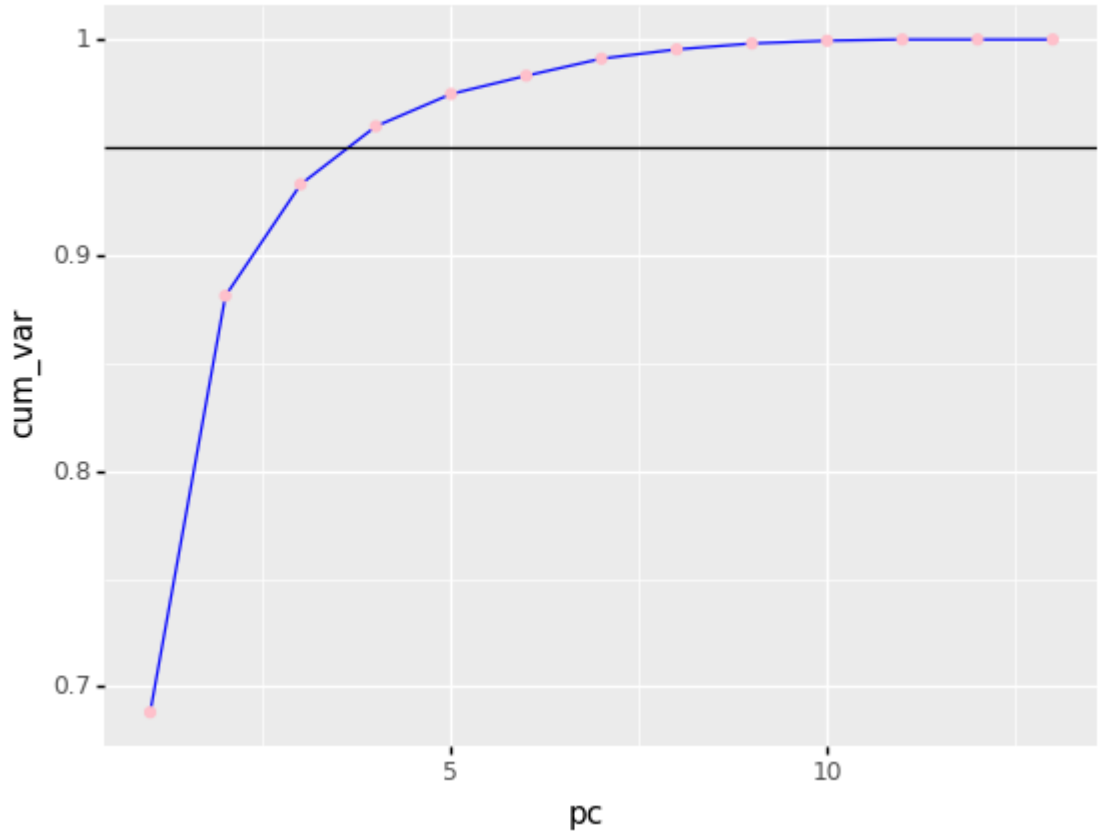
	expl_var	pc	cum_var
0	0.688077	1	0.688077
1	0.193048	2	0.881125
2	0.051709	3	0.932834
3	0.026920	4	0.959754
4	0.014974	5	0.974729

In [27]:
```
(ggplot(pcaDF, aes(x = "pc", y = "expl_var")) + geom_line() + geom_point())
```



Out[27]: <ggplot: (97179189483)>

In [28]:
```
pcaDF.append({"expl_var":0, "pc":0 , "cum_var": 0}, ignore_index = True)
(ggplot(pcaDF, aes(x = "pc", y = "cum_var")) + geom_line(color = "blue") +
 geom_point(color = "pink") + geom_hline(yintercept = 0.95))
```



Out[28]: <ggplot: (97179274915)>

In [29]:
```
loadings = pd.DataFrame({"loading": pca.components_.flatten(),
            "comp": np.repeat(range(1,14), 13 ,
        axis=0), "variable":np.tile(features,13) })

loadings.head(27)
```

Out[29]:

	loading	comp	variable
0	0.316891	1	PA
1	0.320321	1	R
2	0.319238	1	H
3	0.307304	1	2B
4	0.213698	1	3B
5	0.300076	1	HR
6	0.315393	1	RBI
7	0.298702	1	SO
8	0.193424	1	BA
9	0.195326	1	OBP
10	0.214836	1	SLG
11	0.220009	1	OPS

	loading	comp	variable
12	0.325722	1	TB
13	-0.170958	2	PA
14	-0.140204	2	R
15	-0.137761	2	H
16	-0.140081	2	2B
17	-0.115855	2	3B
18	-0.095800	2	HR
19	-0.116885	2	RBI
20	-0.182199	2	SO
21	0.462770	2	BA
22	0.458357	2	OBP
23	0.429516	2	SLG
24	0.465815	2	OPS
25	-0.131690	2	TB
26	0.011090	3	PA

In [30]:
```
pcomps3 = pca.transform(data[features])
pcomps3 = pd.DataFrame(pcomps3[:,0:3])
pcomps5 = pca.transform(data[features])
pcomps5 = pd.DataFrame(pcomps5[:, 0:5])
pcomps10 = pca.transform(data[features])
pcomps10 = pd.DataFrame(pcomps10[:, 0:10])
```

In [31]:
```
lr3 = LinearRegression()
lr3.fit(pcomps3, data["Salary"])
print("3 PCs:    ", lr3.score(pcomps3, data["Salary"]))
```

```
3 PCs:     0.594071184218403
```

In [32]:
```
lr3.fit(pcomps5, data["Salary"])
print("5 PCs:    ", lr3.score(pcomps5, data["Salary"]))
```

```
5 PCs:     0.5941221693605517
```

In [33]:
```
lr3.fit(pcomps10, data["Salary"])
print("10 PCs:    ", lr3.score(pcomps10, data["Salary"]))
```

```
10 PCs:     0.6604461027976123
```

In [ ]:
```

```