Crime and Communities

Justin Han

1 Dataset exploration

To start, we want to examine and familiarize ourselves with the dataset. More importantly, we want to learn more about our dataset's nuances. Some of the attributes we should pay attention to include the dimension, the number of categorical versus numerical variables, and how many missing values there are. Below is a high level overview which includes the attributes of interest as well as some other useful to know information about our data, such as memory.

```
## Rows: 1,994
## Columns: 125
## $ population            <dbl> 11980, 23123, 29344, 16656, 140494, 28700, 59...
## $ householdsize         <dbl> 3.10, 2.82, 2.43, 2.40, 2.45, 2.60, 2.45, 2.4...
## $ racepctblack          <dbl> 1.37, 0.80, 0.74, 1.70, 2.51, 1.60, 14.20, 0....
## $ racePctWhite          <dbl> 91.78, 95.57, 94.33, 97.35, 95.65, 96.57, 84....
## $ racePctAsian          <dbl> 6.50, 3.44, 3.43, 0.50, 0.90, 1.47, 0.40, 1.2...
## $ racePctHisp           <dbl> 1.88, 0.85, 2.35, 0.70, 0.95, 1.10, 0.63, 0.7...
## $ agePct12t21           <dbl> 12.47, 11.01, 11.36, 12.55, 18.09, 11.17, 15....
## $ agePct12t29           <dbl> 21.44, 21.30, 25.88, 25.20, 32.89, 27.41, 27....
## $ agePct16t24           <dbl> 10.93, 10.48, 11.01, 12.19, 20.04, 12.76, 14....
## $ agePct65up            <dbl> 11.33, 17.18, 10.28, 17.57, 13.26, 14.42, 14....
## $ numbUrban             <dbl> 11980, 23123, 29344, 0, 140494, 28700, 59449,...
## $ pctUrban              <dbl> 100.00, 100.00, 100.00, 0.00, 100.00, 100.00,...
## $ medIncome             <dbl> 75122, 47917, 35669, 20580, 21577, 42805, 232...
## $ pctWWage              <dbl> 89.24, 78.99, 82.00, 68.15, 75.78, 79.47, 71....
## $ pctWFarmSelf          <dbl> 1.55, 1.11, 1.15, 0.24, 1.00, 0.39, 0.67, 2.9...
## $ pctWInvInc            <dbl> 70.20, 64.11, 55.73, 38.95, 41.15, 47.70, 35....
## $ pctWSocSec            <dbl> 23.62, 35.50, 22.25, 39.48, 29.31, 30.23, 32....
## $ pctWPubAsst           <dbl> 1.03, 2.75, 2.94, 11.71, 7.12, 5.41, 8.81, 4....
## $ pctWRetire            <dbl> 18.39, 22.85, 14.56, 18.33, 14.09, 17.23, 22....
## $ medFamInc             <dbl> 79584, 55323, 42112, 26501, 27705, 50394, 289...
## $ perCapInc             <dbl> 29711, 20148, 16946, 10810, 11878, 18193, 121...
## $ whitePerCap           <dbl> 30233, 20191, 17103, 10909, 12029, 18276, 125...
## $ blackPerCap           <dbl> 13600, 18137, 16644, 9984, 7382, 17342, 9820,...
## $ indianPerCap          <dbl> 5725, 0, 21606, 4941, 10264, 21482, 6634, 534...
## $ AsianPerCap           <dbl> 27101, 20074, 15528, 3541, 10753, 12639, 8802...
## $ OtherPerCap           <dbl> 5115, 5250, 5954, 2451, 7192, 21852, 7428, 53...
## $ HispPerCap            <dbl> 22838, 12222, 8405, 4391, 8104, 22594, 6187, ...
## $ NumUnderPov           <dbl> 227, 885, 1389, 2831, 23223, 1126, 10320, 960...
## $ PctPopUnderPov        <dbl> 1.96, 3.98, 4.75, 17.23, 17.78, 4.01, 17.98, ...
## $ PctLess9thGrade       <dbl> 5.81, 5.61, 2.80, 11.05, 8.76, 4.49, 10.09, 5...
## $ PctNotHSGrad          <dbl> 9.90, 13.72, 9.09, 33.68, 23.03, 13.89, 28.67...
## $ PctBSorMore           <dbl> 48.18, 29.89, 30.13, 10.81, 20.66, 27.01, 12....
## $ PctUnemployed         <dbl> 2.70, 2.43, 4.01, 9.86, 5.72, 4.85, 8.19, 4.1...
## $ PctEmploy             <dbl> 64.55, 61.96, 69.80, 54.74, 59.02, 65.42, 56....
## $ PctEmplManu           <dbl> 14.65, 12.26, 15.95, 31.22, 14.31, 14.02, 27....
```

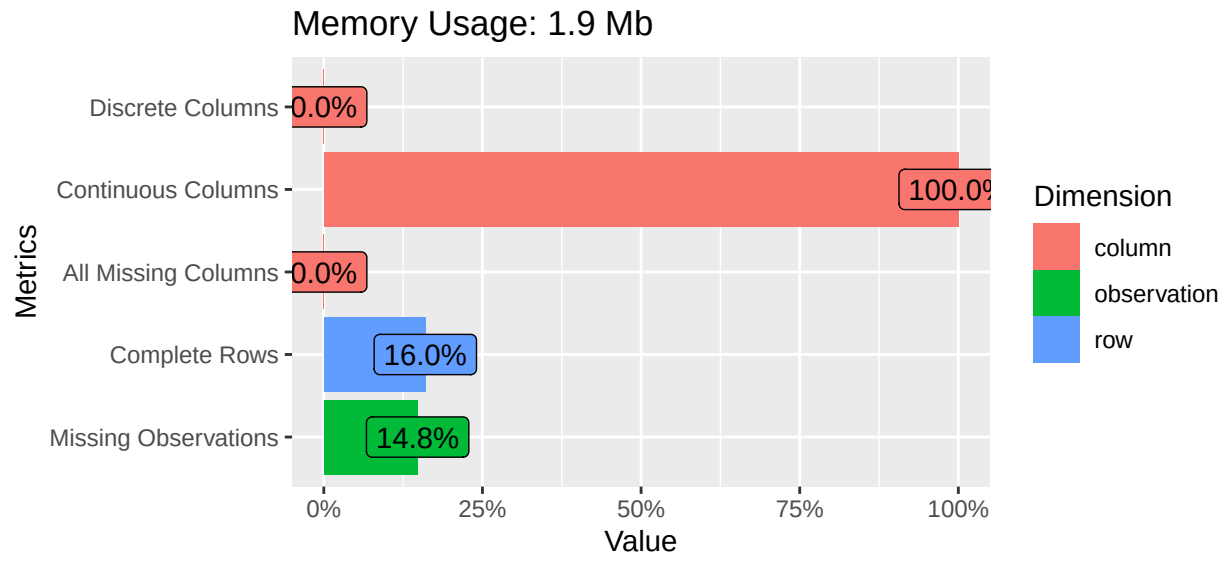
```
## $ PctEmplProfServ       <dbl> 28.82, 29.28, 21.52, 27.43, 26.83, 27.17, 21....
## $ PctOccupManu          <dbl> 5.49, 6.39, 8.79, 26.76, 14.72, 8.50, 21.92, ...
## $ PctOccupMgmtProf      <dbl> 50.73, 37.64, 32.48, 22.71, 23.42, 32.78, 18....
## $ MalePctDivorce        <dbl> 3.67, 4.23, 10.10, 10.98, 11.40, 5.97, 13.28,...
## $ MalePctNevMarr        <dbl> 26.38, 27.99, 25.78, 28.15, 33.32, 36.05, 28....
## $ FemalePctDiv          <dbl> 5.22, 6.45, 14.76, 14.47, 14.46, 9.06, 16.33,...
## $ TotalPctDiv           <dbl> 4.47, 5.42, 12.55, 12.91, 13.04, 7.64, 14.94,...
## $ PersPerFam            <dbl> 3.22, 3.11, 2.95, 2.98, 2.89, 3.14, 2.95, 3.0...
## $ PctFam2Par            <dbl> 91.43, 86.91, 78.54, 64.02, 71.94, 79.53, 62....
## $ PctKids2Par           <dbl> 90.17, 85.33, 78.85, 62.36, 69.79, 79.76, 58....
## $ PctYoungKids2Par      <dbl> 95.78, 96.82, 92.37, 65.38, 79.76, 92.05, 69....
## $ PctTeen2Par           <dbl> 95.81, 86.46, 75.72, 67.43, 75.33, 77.12, 62....
## $ PctWorkMomYoungKids   <dbl> 44.56, 51.14, 66.08, 59.59, 62.96, 65.16, 63....
## $ PctWorkMom            <dbl> 58.88, 62.43, 74.19, 70.27, 70.52, 72.81, 72....
## $ NumKidsBornNeverMar   <dbl> 31, 43, 164, 561, 1511, 263, 2368, 751, 3537,...
## $ PctKidsBornNeverMar   <dbl> 0.36, 0.24, 0.88, 3.84, 1.58, 1.18, 4.66, 1.6...
## $ NumImmig              <dbl> 1277, 1920, 1468, 339, 2091, 2637, 517, 1474,...
## $ PctImmigRecent        <dbl> 8.69, 5.21, 16.42, 13.86, 21.33, 11.38, 13.15...
## $ PctImmigRec5          <dbl> 13.00, 8.65, 23.98, 13.86, 30.56, 16.27, 22.8...
## $ PctImmigRec8          <dbl> 20.99, 13.33, 32.08, 15.34, 38.02, 23.93, 28....
## $ PctImmigRec10         <dbl> 30.93, 22.50, 35.63, 15.34, 45.48, 27.76, 33....
## $ PctRecentImmig        <dbl> 0.93, 0.43, 0.82, 0.28, 0.32, 1.05, 0.11, 0.4...
## $ PctRecImmig5          <dbl> 1.39, 0.72, 1.20, 0.28, 0.45, 1.49, 0.20, 0.6...
## $ PctRecImmig8          <dbl> 2.24, 1.11, 1.61, 0.31, 0.57, 2.20, 0.25, 0.9...
## $ PctRecImmig10         <dbl> 3.30, 1.87, 1.78, 0.31, 0.68, 2.55, 0.29, 1.0...
## $ PctSpeakEnglOnly      <dbl> 85.68, 87.79, 93.11, 94.98, 96.87, 89.98, 97....
## $ PctNotSpeakEnglWell   <dbl> 1.37, 1.81, 1.14, 0.56, 0.60, 0.60, 0.28, 0.4...
## $ PctLargHouseFam       <dbl> 4.81, 4.25, 2.97, 3.93, 3.08, 5.08, 3.85, 2.5...
## $ PctLargHouseOccup     <dbl> 4.17, 3.34, 2.05, 2.56, 1.92, 3.46, 2.55, 1.5...
## $ PersPerOccupHous      <dbl> 2.99, 2.70, 2.42, 2.37, 2.28, 2.55, 2.36, 2.3...
## $ PersPerOwnOccHous     <dbl> 3.00, 2.83, 2.69, 2.51, 2.37, 2.89, 2.42, 2.7...
## $ PersPerRentOccHous    <dbl> 2.84, 1.96, 2.06, 2.20, 2.16, 2.09, 2.27, 1.9...
## $ PctPersOwnOccup       <dbl> 91.46, 89.03, 64.18, 58.18, 57.81, 64.62, 65....
## $ PctPersDenseHous      <dbl> 0.39, 1.01, 2.03, 1.21, 2.11, 1.47, 1.90, 1.6...
## $ PctHousLess3BR        <dbl> 11.06, 23.60, 47.46, 45.66, 53.19, 47.35, 56....
## $ MedNumBR              <dbl> 3, 3, 3, 3, 2, 3, 2, 2, 2, 2, 2, 2, 3, 3, 2, ...
## $ HousVacant            <dbl> 64, 240, 544, 669, 5119, 566, 2051, 1562, 560...
## $ PctHousOccup          <dbl> 98.37, 97.15, 95.68, 91.19, 91.81, 95.11, 92....
## $ PctHousOwnOcc         <dbl> 91.01, 84.88, 57.79, 54.89, 55.50, 56.96, 63....
## $ PctVacantBoarded      <dbl> 3.12, 0.00, 0.92, 2.54, 2.09, 1.41, 6.39, 0.4...
## $ PctVacMore6Mos        <dbl> 37.50, 18.33, 7.54, 57.85, 26.22, 34.45, 56.3...
## $ MedYrHousBuilt        <dbl> 1959, 1958, 1976, 1939, 1966, 1956, 1954, 197...
## $ PctHousNoPhone        <dbl> 0.00, 0.31, 1.55, 7.00, 6.13, 0.69, 8.42, 2.6...
## $ PctWOFullPlumb        <dbl> 0.28, 0.14, 0.12, 0.87, 0.31, 0.28, 0.49, 0.1...
## $ OwnOccLowQuart        <dbl> 215900, 136300, 74700, 36400, 37700, 155100, ...
## $ OwnOccMedVal          <dbl> 262600, 164200, 90400, 49600, 53900, 179000, ...
## $ OwnOccHiQuart         <dbl> 326900, 199900, 112000, 66500, 73100, 215500,...
## $ OwnOccQrange          <dbl> 111000, 63600, 37300, 30100, 35400, 60400, 26...
## $ RentLowQ              <dbl> 685, 467, 370, 195, 215, 463, 186, 241, 192, ...
## $ RentMedian            <dbl> 1001, 560, 428, 250, 280, 669, 253, 321, 281,...
## $ RentHighQ             <dbl> 1001, 672, 520, 309, 349, 824, 325, 387, 369,...
## $ RentQrange            <dbl> 316, 205, 150, 114, 134, 361, 139, 146, 177, ...
## $ MedRent               <dbl> 1001, 627, 484, 333, 340, 736, 338, 355, 353,...
## $ MedRentPctHousInc     <dbl> 23.8, 27.6, 24.1, 28.7, 26.4, 24.4, 26.3, 25....
```

```
## $ MedOwnCostPctInc      <dbl> 21.1, 20.7, 21.7, 20.6, 17.3, 20.8, 15.1, 20....
## $ MedOwnCostPctIncNoMtg <dbl> 14.0, 12.5, 11.6, 14.5, 11.7, 12.5, 12.2, 12....
## $ NumInShelters         <dbl> 11, 0, 16, 0, 327, 0, 21, 125, 43, 1, 20, 28,...
## $ NumStreet             <dbl> 0, 0, 0, 0, 4, 0, 0, 15, 4, 0, 49, 2, 0, 1, 1...
## $ PctForeignBorn        <dbl> 10.66, 8.30, 5.00, 2.04, 1.49, 9.19, 0.87, 1....
## $ PctBornSameState      <dbl> 53.72, 77.17, 44.77, 88.71, 64.35, 77.30, 73....
## $ PctSameHouse85        <dbl> 65.29, 71.27, 36.60, 56.70, 42.29, 63.45, 54....
## $ PctSameCity85         <dbl> 78.09, 90.22, 61.26, 90.17, 70.61, 82.23, 85....
## $ PctSameState85        <dbl> 89.14, 96.12, 82.85, 96.24, 85.66, 93.53, 91....
## $ LemasSwornFT          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 198, NA, NA, ...
## $ LemasSwFTPerPop       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 183.53, NA, N...
## $ LemasSwFTFieldOps     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 187, NA, NA, ...
## $ LemasSwFTFieldPerPop  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 173.33, NA, N...
## $ LemasTotalReq         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 73432, NA, NA...
## $ LemasTotReqPerPop     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 68065.1, NA, ...
## $ PolicReqPerOffic      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 370.9, NA, NA...
## $ PolicPerPop           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 183.5, NA, NA...
## $ RacialMatchCommPol    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 89.32, NA, NA...
## $ PctPolicWhite         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 78.28, NA, NA...
## $ PctPolicBlack         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 11.11, NA, NA...
## $ PctPolicHisp          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 10.61, NA, NA...
## $ PctPolicAsian         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 0.00, NA, NA,...
## $ PctPolicMinor         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 21.72, NA, NA...
## $ OfficAssgnDrugUnits   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 13, NA, NA, N...
## $ NumKindsDrugsSeiz     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 12, NA, NA, N...
## $ PolicAveOTWorked      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 60.2, NA, NA,...
## $ LandArea              <dbl> 6.5, 10.6, 10.6, 5.2, 70.4, 10.9, 39.2, 30.9,...
## $ PopDens               <dbl> 1845.9, 2186.7, 2780.9, 3217.7, 1995.7, 2643....
## $ PctUsePubTrans        <dbl> 9.63, 3.84, 4.37, 3.31, 0.97, 9.62, 0.70, 1.4...
## $ PolicCars             <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 100, NA, NA, ...
## $ PolicOperBudg         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 9315474, NA, ...
## $ LemasPctPolicOnPatr   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 94.44, NA, NA...
## $ LemasGangUnitDeploy   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 10, NA, NA, N...
## $ LemasPctOfficDrugUn   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.0...
## $ PolicBudgPerPop       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 86346.3, NA, ...
## $ ViolentCrimesPerPop   <dbl> 41.02, 127.56, 218.59, 306.64, 442.95, 226.63...
```

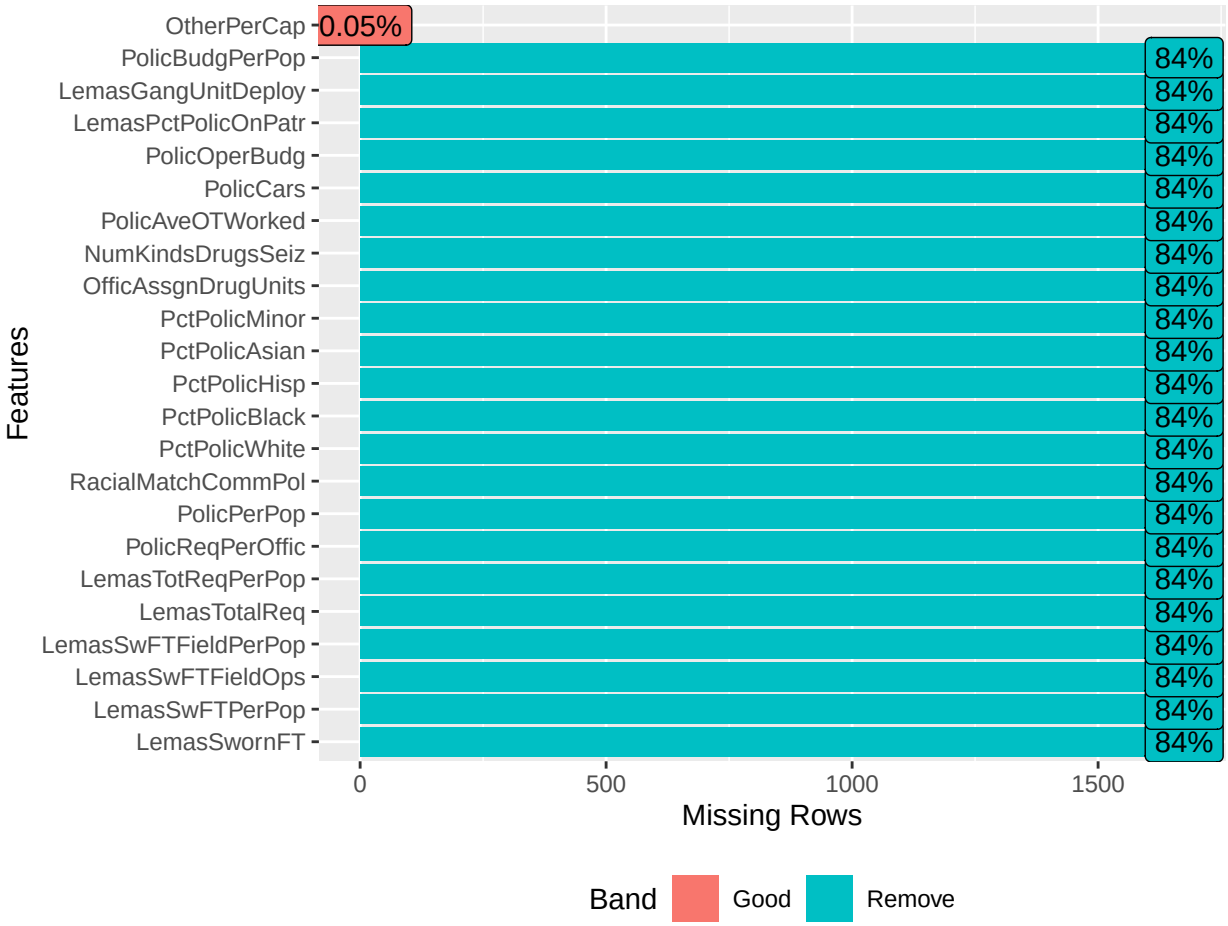
```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1 1994     125               0                125                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                36851           319             249250      2020864
```

Here, "complete_rows" refers to the number of rows without any missing values, "all_missing_columns" refers to the number of missing columns (the entire column is NA), "total_observations" refers to each value in the dataset (including missing values), and "discrete_columns" refers to the number of categorical variables in our data. From our findings, we see that there are no cateogorial variables. Additionally, out of the 249250 values, there are a total of 36851 missing values.

Our findings can be summarized visually with a barplot that gives us the proportions of each attribute.

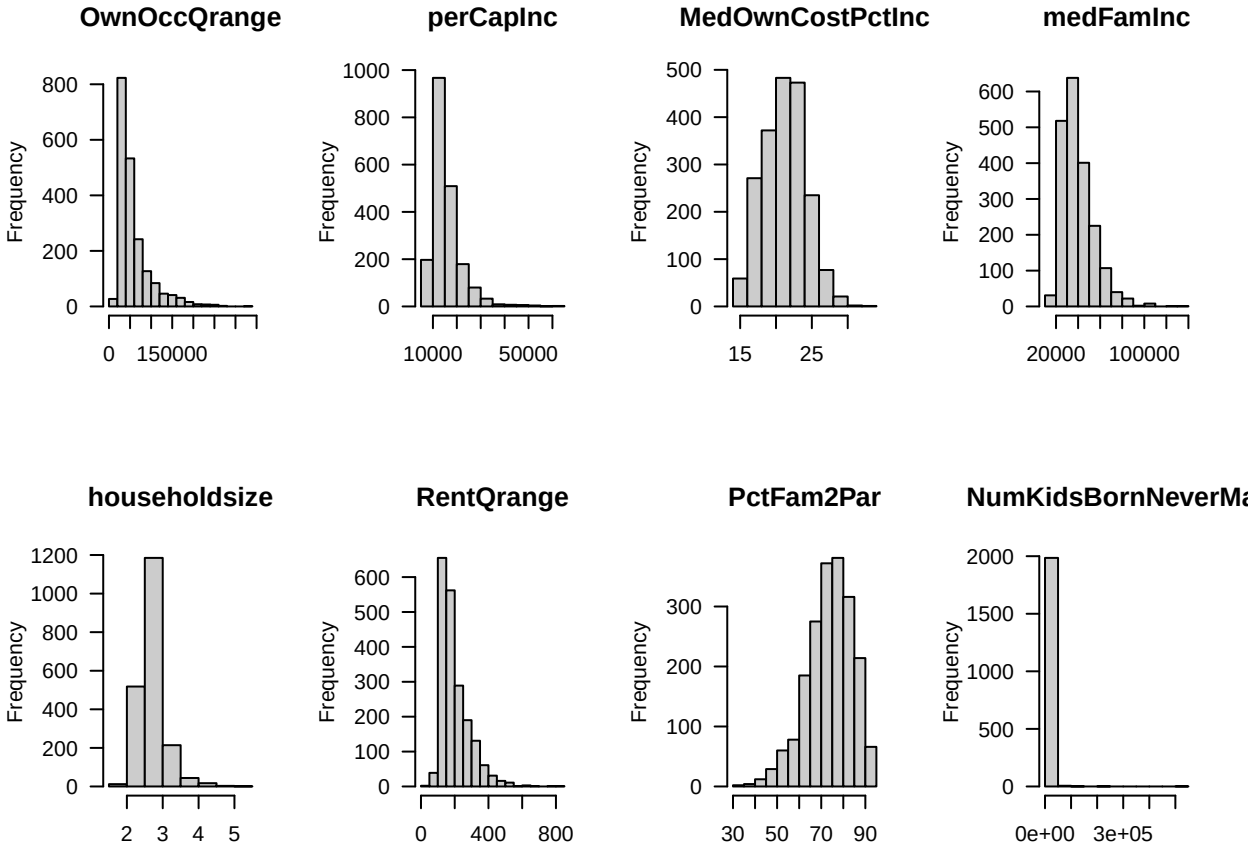


From our visualization, it becomes clear that only roughly 16% of all rows are not completely missing and about 15% of the values in the dataset are missing. Missing values definitely will cause problems when it comes to accurate analysis. Hence the next step will be to take a closer look at what is missing and what we can do to alleviate the problem. This can be visualized as well with a barplot which illustrates the proportion of missing values for each variable given that the variable contains missing values.

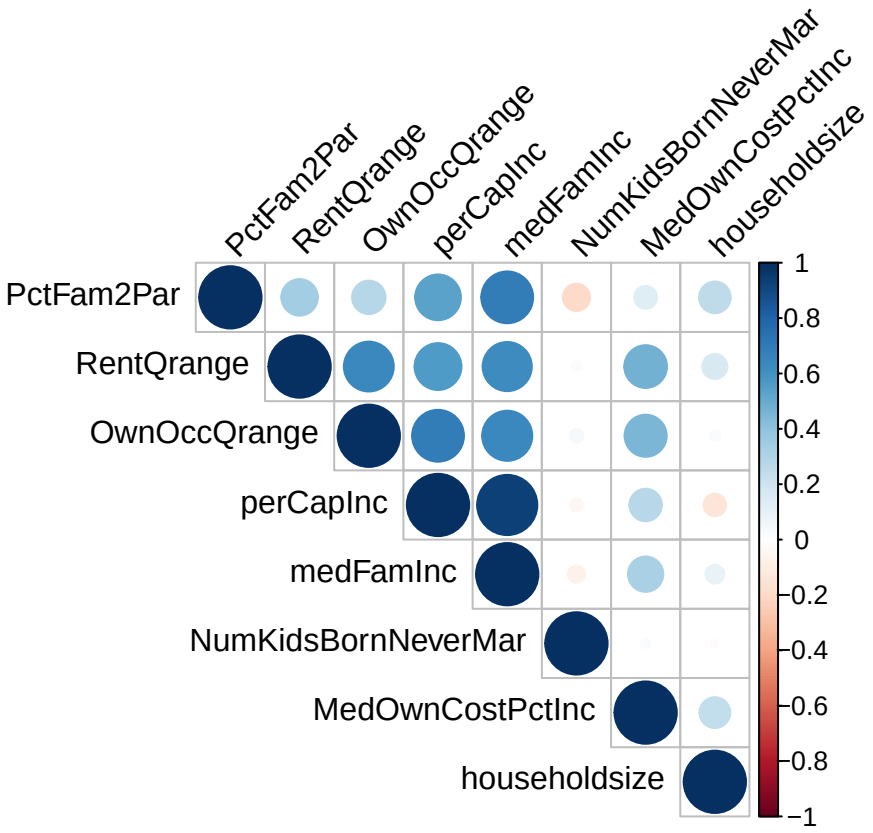


From our illustration, we observe that out of the 23 variables that contain missing values, all of them except for "OtherPerCap" has a significant portion of missing values. Because those variables are mostly missing values, we can drop them using the `drop_columns` function since they probably will not provide us with much information. Doing this drop reduces the number of variables in our dataset to 103 (previously 125).

Now that we have learned about and cleaned our dataset, we can visualize a random subset of features just to get a sense of their distributions. Doing this will hopefully give us a rough idea of the properties of the 103 variables we have available to work with. Below are the histograms of a randomly selected set of predictor variables.



We notice that many of the variables have a skewed distribution as depicted by the shapes of the histograms in the subset (this indicates that we might have to scale our data which will be addressed later). Having seen how each variable behaves on its own, we can even go one step further and examine how each variable behaves with respect to one another. Learning which variables are correlated can help us gain intuition behind why certain variables can be dropped. Below is a plot of the correlations (since the correlation plot consisting of all predictors is really large, we will just examine it for the above subset of predictors).



2 Analysis

Our goal is to develop a model to predict "ViolentCrimesPerPop". Since there are so many variables in our dataset, we hope to find an appropriate model using dimension reduction methods or shrinkage methods so that we can predict our desired response in a "friendly" way. In particular, we will use methods such as PC regression, PLS regression, Ridge regression, and Lasso regression. Addtionally, we will use cross-validation to select tuning parameters and utilize the three-way holdout method to perform model selection and model assessment. We begin by defining our response vector and feature matrix. Also, recall that there is one column ("OtherPerCap") which contains a single missing value of which we chose to keep. To deal with this missing value, we will impute the missing value with its respective column mean.

2.1 Three-way hold-out method

Before we begin implementing any of the regularization methods, we first split the data into three different parts:

1. Training set: 60% of the data (chosen at random)
2. Validation set: 20% of the data (i.e. one half of the remaining 40% not in training, chosen at random)
3. Test set: 20% of the data (i.e. the other half of the remaining 40% not in training, chosen at random)

2.2 PCA

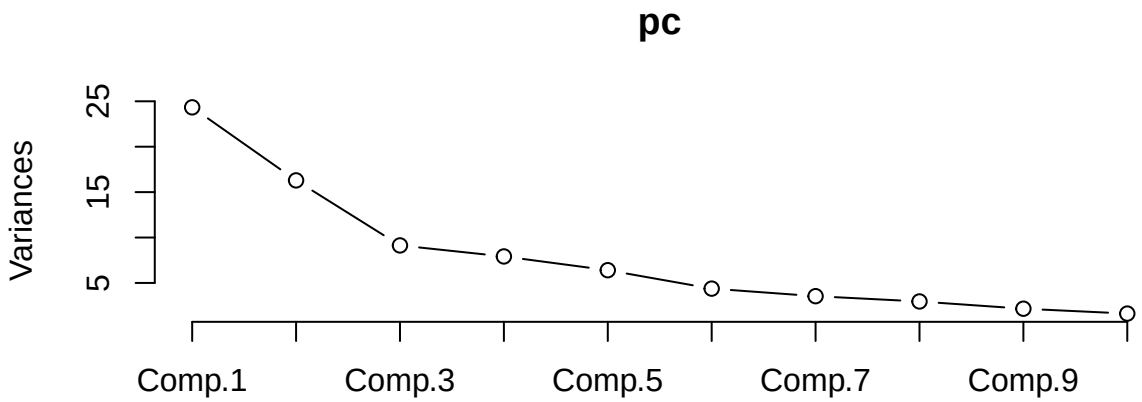
After successfully prepping our data, we standardize it in order to ensure that the significance of each variable is properly captured relative to one another. The first thing that we would like to do is attempt to reduce

the dimensionality by performing PCA.

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     4.9326789 4.0360965 3.02162558 2.81397046 2.53052082
## Proportion of Variance 0.2386621 0.1597868 0.08955688 0.07767062 0.06281126
## Cumulative Proportion  0.2386621 0.3984488 0.48800569 0.56567631 0.62848757
##                           Comp.6    Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     2.09168135 1.8823422 1.72071511 1.47110777 1.27572454
## Proportion of Variance 0.04291496 0.0347548 0.02904261 0.02122788 0.01596362
## Cumulative Proportion  0.67140253 0.7061573 0.73519995 0.75642783 0.77239145
##                           Comp.11    Comp.12    Comp.13    Comp.14    Comp.15
## Standard deviation     1.26639012 1.21586732 1.20654739 1.04850815 1.01969247
## Proportion of Variance 0.01573087 0.01450074 0.01427928 0.01078354 0.01019897
## Cumulative Proportion  0.78812232 0.80262306 0.81690234 0.82768588 0.83788485
##                           Comp.16     Comp.17     Comp.18     Comp.19
## Standard deviation     0.989307306 0.976271643 0.937987851 0.914970955
## Proportion of Variance 0.009600196 0.009348868 0.008630026 0.008211685
## Cumulative Proportion  0.847485042 0.856833910 0.865463936 0.873675622
##                           Comp.20     Comp.21     Comp.22     Comp.23
## Standard deviation     0.875911957 0.865512451 0.850786030 0.780493078
## Proportion of Variance 0.007525556 0.007347918 0.007100001 0.005975246
## Cumulative Proportion  0.881201178 0.888549096 0.895649097 0.901624343
##                           Comp.24     Comp.25     Comp.26    Comp.27
## Standard deviation     0.769692094 0.728701908 0.708472385 0.70300746
## Proportion of Variance 0.005811012 0.005208558 0.004923382 0.00484772
## Cumulative Proportion  0.907435354 0.912643912 0.917567294 0.92241501
##                           Comp.28     Comp.29    Comp.30    Comp.31
## Standard deviation     0.686829233 0.668760688 0.65251133 0.62607171
## Proportion of Variance 0.004627168 0.004386914 0.00417632 0.00384473
## Cumulative Proportion  0.927042182 0.931429096 0.93560542 0.93945015
##                           Comp.32     Comp.33     Comp.34     Comp.35
## Standard deviation     0.614053732 0.603214708 0.579554833 0.565048753
## Proportion of Variance 0.003698541 0.003569123 0.003294631 0.003131768
## Cumulative Proportion  0.943148688 0.946717811 0.950012441 0.953144209
##                           Comp.36     Comp.37     Comp.38    Comp.39   Comp.40
## Standard deviation     0.538861573 0.520473095 0.508690135 0.50557907 0.4844127
## Proportion of Variance 0.002848211 0.002657139 0.002538191 0.00250724 0.0023017
## Cumulative Proportion  0.955992420 0.958649559 0.961187750 0.96369499 0.9659967
##                           Comp.41     Comp.42    Comp.43     Comp.44
## Standard deviation     0.481134399 0.465598776 0.44850228 0.435680860
## Proportion of Variance 0.002270652 0.002126382 0.00197309 0.001861893
## Cumulative Proportion  0.968267341 0.970393723 0.97236681 0.974228707
##                           Comp.45     Comp.46     Comp.47    Comp.48
## Standard deviation     0.42850672 0.420775769 0.397115542 0.38737787
## Proportion of Variance 0.00180108 0.001736677 0.001546862 0.00147193
## Cumulative Proportion  0.97602979 0.977766464 0.979313325 0.98078526
##                           Comp.49     Comp.50    Comp.51     Comp.52
## Standard deviation     0.364008940 0.352446149 0.34947045 0.322576014
## Proportion of Variance 0.001299696 0.001218437 0.00119795 0.001020662
## Cumulative Proportion  0.982084952 0.983303389 0.98450134 0.985522001
##                           Comp.53      Comp.54      Comp.55      Comp.56
## Standard deviation     0.321055708 0.3142163085 0.3052621659 0.2853168174
## Proportion of Variance 0.001011064 0.0009684454 0.0009140367 0.0007984954
```

```
## Cumulative Proportion  0.986533064 0.9875015095 0.9884155462 0.9892140416
##                            Comp.57      Comp.58      Comp.59      Comp.60
## Standard deviation     0.2737585853 0.2690977660 0.2599109128 0.2561377774
## Proportion of Variance 0.0007351114 0.0007102935 0.0006626233 0.0006435243
## Cumulative Proportion  0.9899491531 0.9906594466 0.9913220699 0.9919655942
##                            Comp.61      Comp.62      Comp.63      Comp.64
## Standard deviation     0.2529292131 0.2397154092 0.2362733634 0.2248870324
## Proportion of Variance 0.0006275028 0.0005636501 0.0005475795 0.0004960741
## Cumulative Proportion  0.9925930970 0.9931567471 0.9937043267 0.9942004007
##                            Comp.65      Comp.66      Comp.67      Comp.68
## Standard deviation     0.2173380176 0.2080171493 0.1986859627 0.1951771870
## Proportion of Variance 0.0004633286 0.0004244397 0.0003872149 0.0003736593
## Cumulative Proportion  0.9946637293 0.9950881690 0.9954753839 0.9958490432
##                            Comp.69      Comp.70      Comp.71      Comp.72
## Standard deviation     0.1856936330 0.1800098925 0.1784278362 0.1727328029
## Proportion of Variance 0.0003382297 0.0003178414 0.0003122791 0.0002926627
## Cumulative Proportion  0.9961872728 0.9965051142 0.9968173933 0.9971100560
##                            Comp.73      Comp.74      Comp.75      Comp.76
## Standard deviation     0.1659213685 0.1619551449 0.1518608298 0.1510578655
## Proportion of Variance 0.0002700364 0.0002572807 0.0002262087 0.0002238228
## Cumulative Proportion  0.9973800924 0.9976373731 0.9978635817 0.9980874045
##                            Comp.77      Comp.78      Comp.79      Comp.80
## Standard deviation     0.1443429543 0.1389526415 0.1326806473 0.1262231991
## Proportion of Variance 0.0002043661 0.0001893875 0.0001726763 0.0001562774
## Cumulative Proportion  0.9982917706 0.9984811581 0.9986538345 0.9988101118
##                            Comp.81      Comp.82      Comp.83      Comp.84
## Standard deviation     0.1235752103 0.1212765575 0.1170376777 0.1110082965
## Proportion of Variance 0.0001497892 0.0001442685 0.0001343597 0.0001208728
## Cumulative Proportion  0.9989599010 0.9991041695 0.9992385292 0.9993594020
##                            Comp.85      Comp.86      Comp.87      Comp.88
## Standard deviation     0.1057593846 1.009005e-01 8.309721e-02 7.888263e-02
## Proportion of Variance 0.0001097123 9.986286e-05 6.773148e-05 6.103522e-05
## Cumulative Proportion  0.9994691143 9.995690e-01 9.996367e-01 9.996977e-01
##                            Comp.89      Comp.90      Comp.91      Comp.92
## Standard deviation     0.0745019065 7.400372e-02 6.520519e-02 6.049967e-02
## Proportion of Variance 0.0000544443 5.371861e-05 4.170442e-05 3.590241e-05
## Cumulative Proportion  0.9997521882 9.998059e-01 9.998476e-01 9.998835e-01
##                            Comp.93      Comp.94      Comp.95      Comp.96
## Standard deviation     5.818417e-02 4.970809e-02 4.677072e-02 4.339695e-02
## Proportion of Variance 3.320683e-05 2.423661e-05 2.145684e-05 1.847294e-05
## Cumulative Proportion  9.999167e-01 9.999410e-01 9.999624e-01 9.999809e-01
##                            Comp.97      Comp.98      Comp.99     Comp.100
## Standard deviation     3.192576e-02 2.233391e-02 1.563978e-02 1.363459e-02
## Proportion of Variance 9.997699e-06 4.892685e-06 2.399269e-06 1.823483e-06
## Cumulative Proportion  9.999909e-01 9.999958e-01 9.999982e-01 1.000000e+00
##                           Comp.101     Comp.102
## Standard deviation     3.670186e-08 1.477783e-08
## Proportion of Variance 1.321277e-17 2.142097e-18
## Cumulative Proportion  1.000000e+00 1.000000e+00
```

We can nicely summarize our findings visually with a plot which will give us a sense of what PCs are important and the amount of variance each captures.

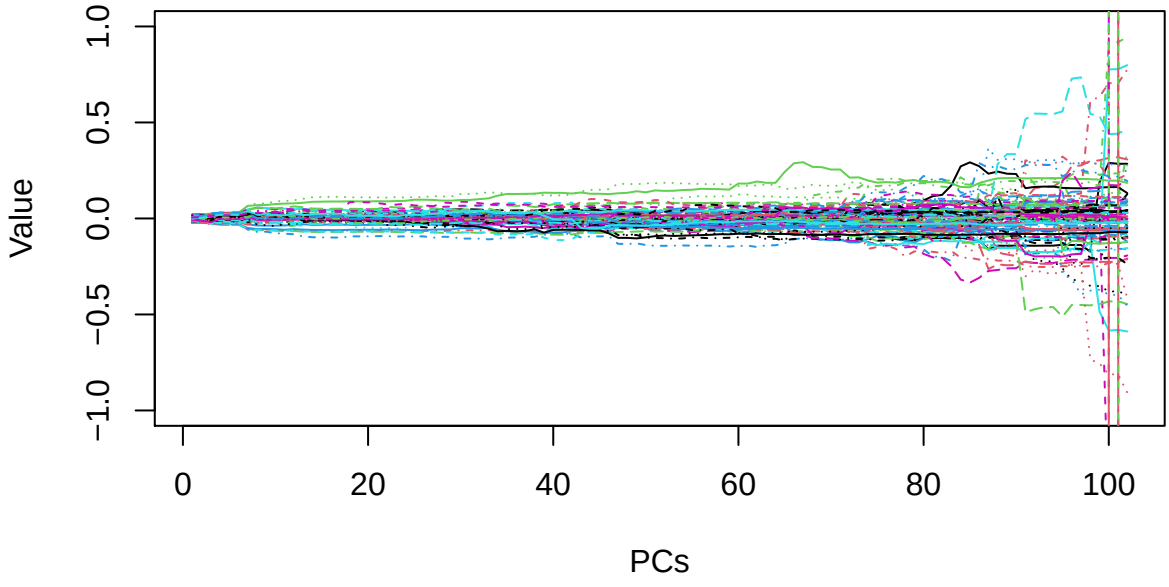


The plot only shows up to the first nine PCs. Despite that, we notice the quick drop off in variance captured which indicates that we do not need to use all of the PCs to get good results since the first couple of PCs already capture most of the variance of our original data. In fact, if we look at the cumulative proportions, we notice that the first 17 PCs already capture roughly 85% of the total variance and the first 34 captures about 95%! This is a big step forward since it shows a significant drop in the number of PCs we would need (total of 102 PCs), reducing dimensionality while still capturing as much variability of the original data as possible. We can build our model using just a few PCs and still get good results; this ultimately sets the stage for dimension reduction techniques like PCR as well as PLSR and gives us a reason as to why we should utilize them.

3 Methods

3.1 Principal Component Regression

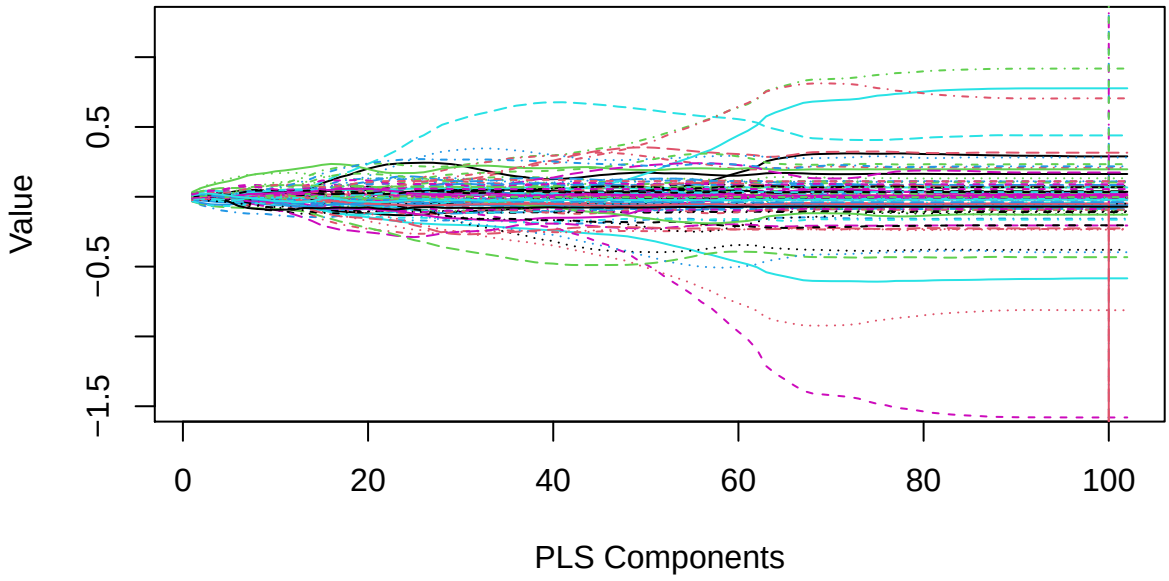
In the first method, we will use Principal Component Regression (PCR). The idea is that we can fit a least squares model on a small number of PCs and obtain coefficients that give us a good model to use in predicting our response. We use the training set to fit PCR. The `pcr` function returns the regression coefficients in terms of the X variables, which simplifies interpretation. We can then visualize how the coefficients grow in terms of the number of retained PCs.



The graph consists of a different line for each coefficient (total of 102 because there are 102 features). Each line captures how each of the coefficients change as the number of retained PCs increases. The coefficients start off stable but as the number of PCs increase, we can see that the values of the coefficients start to grow both in the postive and negative directions. We observe something interesting when the number of PCs reach 60. We can see that at that point, the values of the coefficients begin to fluctuate really rapidly, leading to significant differences between them.

3.2 Partial Least Squares Regression

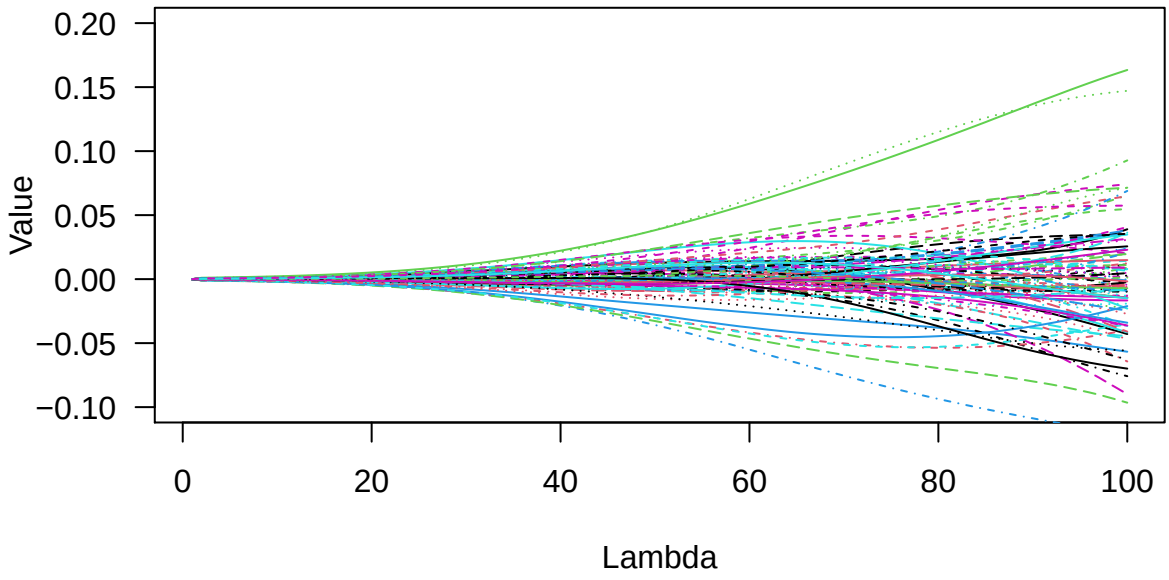
The next method is Partial Least Squares Regression (PLSR). It is very similar to PCR except in PLS, it makes use of the response in order to identify new features (PLS components) that not only approximate the old features well, but also that are related to the response. The function `plsr` returns the regression coefficients in terms of the X variables, which simplifies interpretation yet again. Similar to what we did with PCR, We can visualize how the coefficients grow in terms of the number of retained PLS components.



The graph consists of a different line for each coefficient (total of 102 because there are 102 features). Each line captures how each of the coefficients change as the number of retained PLS components increases. Unlike the coefficients in PCR, we notice that the cofficients in PLS experience much greater changes and are much more sensitive to the number of components retained.

3.3 Ridge Regression

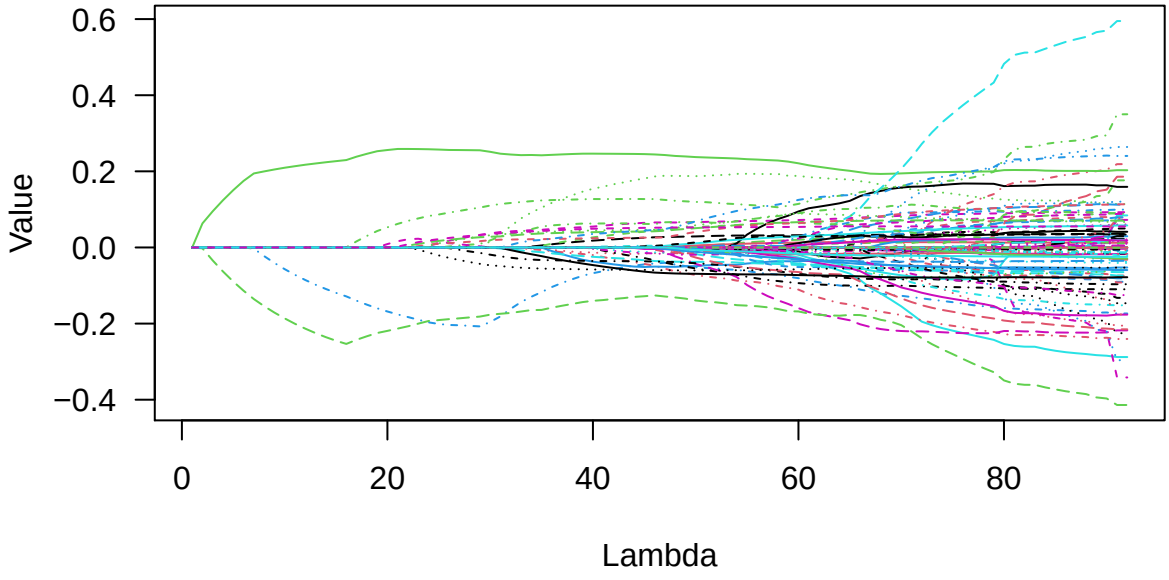
So far, we fitted two dimension reduction methods on our training data and observed how the regression coefficients change when we retain different numbers of components for PCR and PLSR. We can also use shrinkage methods like Ridge regression (RR) and Lasso regression to examine some interesting effects that the tuning parameter, lambda, has on the regression coefficients. This way, we will be able to have multiple sources of comparision, providing us more insight about the different regularization techniques used. Below is a graph of how the regression coefficients change in terms of lambda for Ridge regression.



The graph consists of a different line for each coefficient (total of 102 because there are 102 features). Each line captures how each of the coefficients change with varying values of lambda. We notice that the change in value of the cofficients for RR follow a similar shape as that of PLSR and PCR.

3.4 Lasso Regression

Lasso regression is similar to Ridge regression in the sense that both are shrinkage methods that involve the parameter, lambda. However, one disadvantage that Ridge regression has is the fact that RR will include all features (in this case 102) in the final model. Although this may not be a problem for prediction accuracy, it may make model interpretation somewhat challenging especially when the number of features we are working with is large. Lasso regression helps overcome this issue by slightly adjusting the way the penalty is set up. Below is a graph of how the regression coefficients change in terms of lambda for Lasso regression.



The graph consists of a different line for each coefficient (total of 102 because there are 102 features). Each line captures how each of the coefficients change with varying values of lambda. Again, we notice here that

the change in value of the cofficients for Lasso follow a similar shape as that of PLSR and PCR. The one thing that stands out about the regression coefficients for Lasso is how sensitive it is to lambda early on.

4 Cross-Validation

The question to ask now is: what is the best number of components to have? What is the best lambda to use? The answers to these questions can be found by using k-fold cross validation. The size of k is somewhat arbitrary, but typically 5 or 10 is good. We will go with $k = 5$ here. That means we will have to create folds by splitting the training data into 5 subsets of similar size and then train on the first $k - 1$ folds and reserve the kth fold for "testing". We repeat this process for each fold, training on all folds except for the one that has been reserved for "testing".

Once we have created the folds, we will train our methods on the training data and compute cross-validation MSEs for each value of a tuning parameter (the number of components and lambda are examples of tuning parameters). We note the tuning parameters that yield the lowest cross-validation MSE for each model (training MSE for each).

```
##         pcr        plsr       ridge       lasso
## 14.00000000  4.00000000  0.32492694  0.01223329
```

The above table summarizes, for every method, the best number of components to keep as well as the best lambda to choose (determined by the lowest cross-validation MSE for each). Now that we have found the best tuning parameters for each model, the next question to ask is: which model is the best one? We will use the validation set to determine the answer. In the next step, we will fit our ideal models (those with the best tuning parameters that we just found for each) on our validation set and compare how each model does.

```
##   pcr_mse  plsr_mse ridge_mse lasso_mse
## 0.3624519 0.3588764 0.3476880 0.5369280
```

The model that gives us the lowest validation MSE is the best model out of the bunch; in our case, Ridge Regression yields the lowest validation MSE. Once we have identified the best model, we need to assess its performance using the test set.

5 Final Model Performance

The test MSE for our best model (Ridge regression) is around 0.33. We note that the test MSE is greater than the validation MSE, but by a very small amount. We can conclude that our model did a fairly good job in predicting "ViolentCrimesPerPop". For our last step, we will use our entire dataset (training set, validation set, and test set) to fit our crowned model and observe its coefficients.

```
## 102 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## population            -0.0491638432
## householdsize         -0.0296575851
## racepctblack           0.1360736818
## racePctWhite          -0.0711683487
## racePctAsian          -0.0270917885
## racePctHisp            0.0008447537
## agePct12t21            0.0474468570
```

```
## agePct12t29           -0.1631222262
## agePct16t24            0.0149398041
## agePct65up            -0.0220666184
## numbUrban             -0.0662126258
## pctUrban               0.0782010845
## medIncome              0.0025570737
## pctWWage              -0.1230538106
## pctWFarmSelf           0.0204778255
## pctWInvInc            -0.0846178506
## pctWSocSec             0.0441072657
## pctWPubAsst            0.0578071799
## pctWRetire            -0.0769737452
## medFamInc             -0.0105337681
## perCapInc             -0.0488610155
## whitePerCap           -0.0144640937
## blackPerCap           -0.0117364695
## indianPerCap          -0.0015510813
## AsianPerCap            0.0332978613
## OtherPerCap            0.0333035847
## HispPerCap             0.0116231328
## NumUnderPov           -0.0294094010
## PctPopUnderPov        -0.0899341263
## PctLess9thGrade       -0.1630281039
## PctNotHSGrad           0.1008233839
## PctBSorMore            0.0348004401
## PctUnemployed         -0.0287455401
## PctEmploy              0.0927356630
## PctEmplManu           -0.0577608813
## PctEmplProfServ       -0.0143637236
## PctOccupManu           0.0273220063
## PctOccupMgmtProf       0.0263912145
## MalePctDivorce         0.1572377452
## MalePctNevMarr         0.1145609615
## FemalePctDiv          -0.0852828205
## TotalPctDiv           -0.0272195207
## PersPerFam            -0.0223951462
## PctFam2Par            -0.0268378529
## PctKids2Par           -0.1644580429
## PctYoungKids2Par       0.0111063241
## PctTeen2Par           -0.0018010256
## PctWorkMomYoungKids    0.0302839580
## PctWorkMom            -0.1053712451
## NumKidsBornNeverMar   -0.0722701022
## PctKidsBornNeverMar    0.2087152419
## NumImmig               0.0605466888
## PctImmigRecent         0.0224669666
## PctImmigRec5          -0.0130889453
## PctImmigRec8          -0.0122816049
## PctImmigRec10          0.0177751472
## PctRecentImmig        -0.0145640151
## PctRecImmig5          -0.0368117995
## PctRecImmig8          -0.0078361943
## PctRecImmig10          0.0283682677
## PctSpeakEnglOnly       0.0125285763
```

```
## PctNotSpeakEnglWell   -0.0456592215
## PctLargHouseFam        0.0081864184
## PctLargHouseOccup     -0.0760607938
## PersPerOccupHous       0.1568150205
## PersPerOwnOccHous     -0.0288321659
## PersPerRentOccHous    -0.0266881439
## PctPersOwnOccup       -0.1048253263
## PctPersDenseHous       0.1333692545
## PctHousLess3BR         0.0404593300
## MedNumBR               0.0117927734
## HousVacant             0.1516442887
## PctHousOccup          -0.0348632729
## PctHousOwnOcc          0.0523937702
## PctVacantBoarded       0.0767204224
## PctVacMore6Mos        -0.0480007455
## MedYrHousBuilt         0.0020052541
## PctHousNoPhone         0.0324747898
## PctWOFullPlumb        -0.0062906739
## OwnOccLowQuart        -0.0079906551
## OwnOccMedVal           0.0160187159
## OwnOccHiQuart         -0.0139603868
## OwnOccQrange          -0.0249140880
## RentLowQ              -0.0999785921
## RentMedian             0.0014985799
## RentHighQ             -0.0533925818
## RentQrange             0.0431898420
## MedRent                0.1232714127
## MedRentPctHousInc      0.0121121724
## MedOwnCostPctInc      -0.0103880424
## MedOwnCostPctIncNoMtg -0.0744301120
## NumInShelters          0.0789941307
## NumStreet             -0.0039622510
## PctForeignBorn         0.1349230872
## PctBornSameState      -0.0249750695
## PctSameHouse85         0.0069380749
## PctSameCity85          0.0259725524
## PctSameState85         0.0148322542
## LandArea               0.0020638694
## PopDens               -0.0531134884
## PctUsePubTrans        -0.0041650996
## LemasPctOfficDrugUn    0.0451312598
```