In [1]:

```python
import matplotlib.pyplot as plt
import numpy
import pandas
import sys
import Regression
from scipy.special import loggamma
from scipy.stats import norm, chi2
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
# Set some options for printing all the columns
numpy.set_printoptions(precision = 10, threshold = sys.maxsize)
numpy.set_printoptions(linewidth = numpy.inf)

pandas.set_option('display.max_columns', None)
pandas.set_option('display.expand_frame_repr', False)
pandas.set_option('max_colwidth', None)
pandas.set_option('precision', 10)

pandas.options.display.float_format = '{:,.7e}'.format
```

In [3]:

```python
claim_history = pandas.read_csv('claim_history.csv')
categorical = claim_history[['MSTATUS','CAR_TYPE','HOMEKIDS',
                             'KIDSDRIV','REVOKED','URBANICITY']].astype("category")

interval = claim_history[["CAR_AGE","MVR_PTS","TIF","TRAVTIME"]]
offset = claim_history[["EXPOSURE"]]
response = claim_history[["CLM_COUNT"]]

trainData = pandas.concat([interval,categorical,offset,response],axis=1).dropna()
trainData.reset_index(inplace = True)
trainData
```

Out[3]:

	index	CAR_AGE	MVR_PTS	TIF	TRAVTIME	MSTATUS	CAR_TYPE	HOMEKIDS	KIDSDRIV	REVOKED	URBANICITY	EXPOSURE
0	0	1.8000000e+01	3	11	14	No	Minivan	0	0	No	Highly Urban/ Urban	1.8900000e-01
1	1	1.0000000e+00	0	1	22	No	Minivan	0	0	No	Highly Urban/ Urban	1.0000000e+00
2	2	1.0000000e+01	2	1	26	No	Van	0	0	No	Highly Urban/ Urban	1.0000000e+00
3	3	1.0000000e+01	3	4	5	Yes	SUV	1	0	No	Highly Urban/ Urban	8.2800000e-01
4	4	6.0000000e+00	0	7	32	Yes	Minivan	0	0	No	Highly Urban/ Urban	7.2900000e-01
...	...	...	...	...	...	...	...	...	...	...	...	...
9657	10297	1.7000000e+01	2	15	21	Yes	Minivan	2	1	No	Highly Urban/ Urban	1.0000000e+00
9658	10298	1.0000000e+00	0	6	36	Yes	Panel Truck	0	0	No	Highly Urban/ Urban	1.0000000e+00
9659	10299	1.0000000e+00	0	7	12	Yes	SUV	0	0	No	Highly Urban/ Urban	1.0000000e+00
9660	10300	1.1000000e+01	0	6	36	Yes	Minivan	0	0	No	Highly Urban/ Urban	1.0000000e+00
9661	10301	9.0000000e+00	0	6	64	Yes	Minivan	0	0	No	Highly Rural/ Rural	1.0000000e+00

9662 rows × 13 columns

In [4]:

```
# reorder the categories of each categorical predictor in ascending order of number of observations
MSTATUS_freq = trainData["MSTATUS"].value_counts(ascending = True)
CAR_TYPE_freq = trainData["CAR_TYPE"].value_counts(ascending = True)
HOMEKIDS_freq = trainData["HOMEKIDS"].value_counts(ascending = True)
KIDSDRIV_freq = trainData["KIDSDRIV"].value_counts(ascending = True)
REVOKED_freq = trainData["REVOKED"].value_counts(ascending = True)
URBANICITY_freq = trainData["URBANICITY"].value_counts(ascending = True)
```

In [5]:

```
MSTATUS_pm = trainData["MSTATUS"].cat.reorder_categories(list(MSTATUS_freq.index))
CAR_TYPE_pm = trainData["CAR_TYPE"].cat.reorder_categories(list(CAR_TYPE_freq.index))
HOMEKIDS_pm = trainData["HOMEKIDS"].cat.reorder_categories(list(HOMEKIDS_freq.index))
KIDSDRIV_pm = trainData["KIDSDRIV"].cat.reorder_categories(list(KIDSDRIV_freq.index))
REVOKED_pm = trainData["REVOKED"].cat.reorder_categories(list(REVOKED_freq.index))
URBANICITY_pm = trainData["URBANICITY"].cat.reorder_categories(list(URBANICITY_freq.index))

## reorder categories
trainData_reordered_list = [trainData[["CAR_AGE"]],trainData[["MVR_PTS"]],trainData[["TIF"]],trainData[["TRAVTIME"]],
        MSTATUS_pm,CAR_TYPE_pm,HOMEKIDS_pm,KIDSDRIV_pm,REVOKED_pm,URBANICITY_pm,trainData[["CLM_COUNT"]],trainData[['
trainData_reordered = pandas.concat(trainData_reordered_list,axis=1)
trainData_reordered
```

In [6]:

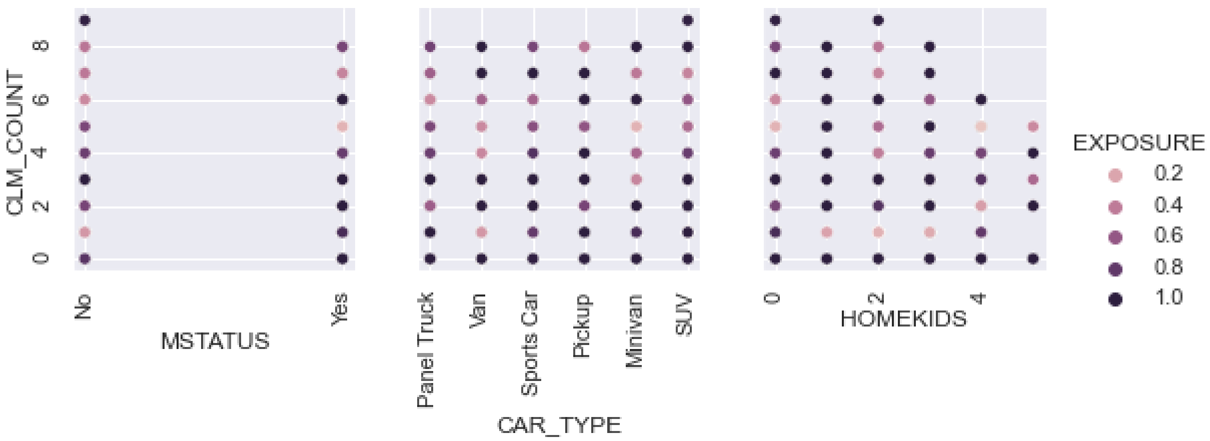
```
## get dummies
term_MSTATUS = pandas.get_dummies(MSTATUS_pm,prefix="MSTATUS")
term_CAR_TYPE = pandas.get_dummies(CAR_TYPE_pm,prefix="CAR_TYPE")
term_HOMEKIDS = pandas.get_dummies(HOMEKIDS_pm,prefix="HOMEKIDS")
term_KIDSDRIV = pandas.get_dummies(KIDSDRIV_pm,prefix="KIDSDRIV")
term_REVOKED = pandas.get_dummies(REVOKED_pm,prefix="REVOKED")
term_URBANICITY = pandas.get_dummies(URBANICITY_pm,prefix="URBANICITY")
```

In [24]:

```
sns.set(rc={'figure.figsize':(12,30)});
sns.pairplot(data=trainData_reordered,
             x_vars = ['CAR_AGE','MVR_PTS','TIF','TRAVTIME'],
             y_vars = ['CLM_COUNT'],
             hue = 'EXPOSURE');
```

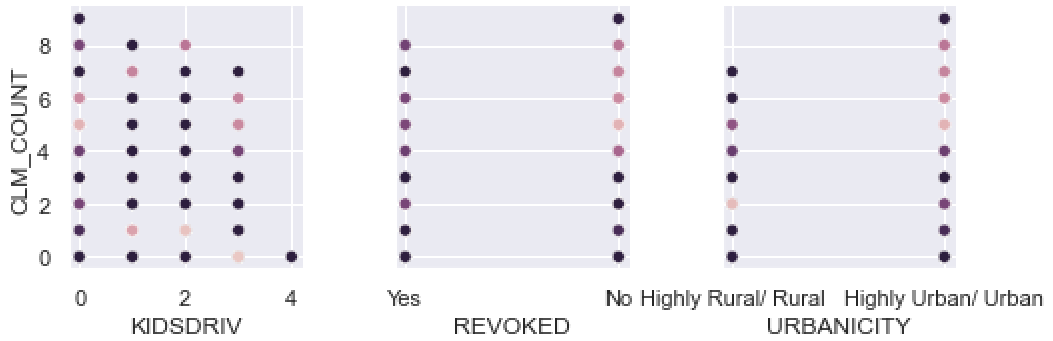
In [25]:

```python
sns.set(rc={'figure.figsize':(12,30)});
g = sns.pairplot(data=trainData_reordered,
             x_vars = ['MSTATUS','CAR_TYPE','HOMEKIDS'],
             y_vars = ['CLM_COUNT'],
             hue = 'EXPOSURE')
for ax in g.axes.flatten():
    ax.tick_params(rotation = 90)
```



In [26]:

```
sns.set(rc={'figure.figsize':(12,30)});
g = sns.pairplot(data=trainData_reordered,
             x_vars = ['KIDSDRIV','REVOKED','URBANICITY'],
             y_vars = ['CLM_COUNT'],
             hue = 'EXPOSURE');
```



In [33]:

```
cand = [trainData[["CAR_AGE"]],trainData[["MVR_PTS"]],trainData[["TIF"]],trainData[["TRAVTIME"]],
        term_MSTATUS,term_CAR_TYPE,term_HOMEKIDS,term_KIDSDRIV,term_REVOKED,term_URBANICITY]
X_train_cand = pandas.concat(cand,axis=1)
y_train = trainData['CLM_COUNT']
o_train = numpy.log(trainData['EXPOSURE'])
```

Question 2

(a): Please provide a summary report of the Forward Selection. The report should include (1) the step number, (2) the predictor entered, (3) the number of non-aliased parameters in the current model, (4) the log-likelihood value of the current model, (5) the Deviance Chi-squares statistic between the current and the previous models, (6) the corresponding Deviance Degree of Freedom, and (7) the corresponding Chi-square significance.

In [34]:

```python
cand = [trainData[["CAR_AGE"]],trainData[["MVR_PTS"]],trainData[["TIF"]],trainData[["TRAVTIME"]],
        term_MSTATUS,term_CAR_TYPE,term_HOMEKIDS,term_KIDSDRIV,term_REVOKED,term_URBANICITY]
CandidateFeature = list(trainData.columns)[1:11]

# set Model M0
X_train = trainData[['CLM_COUNT']].copy()
X_train.insert(0, 'Intercept', 1.0)
X_train.drop(columns = ['CLM_COUNT'], inplace = True)

step_summary = pandas.DataFrame()
step_detail = pandas.DataFrame()

outList = Regression.PoissonModel(X_train, y_train, o_train)
llk_0 = outList[3]
df_0 = len(outList[4])
step_summary = step_summary.append([['0','Intercept', df_0, llk_0, numpy.nan, numpy.nan, numpy.nan]], ignore_index =
step_summary.columns = ["Step","Entered","Num of non-aliased",
                       "Log-likelihood","Deviance chi-sqaures",
                       "Degree of freedom","Significance"]

for i in range(len(CandidateFeature)):
    for m,n in zip(cand,range(len(CandidateFeature))):
        X = X_train.join(m)
        outList = Regression.PoissonModel(X, y_train, o_train)
        llk_1 = outList[3]
        df_1 = len(outList[4])
        deviance_chisq = 2 * (llk_1 - llk_0)
        deviance_df = df_1 - df_0
        deviance_sig = chi2.sf(deviance_chisq, deviance_df)

        step_detail = step_detail.append([[i+1,CandidateFeature[n],
                                           df_1, llk_1, deviance_chisq, deviance_df, deviance_sig]],
                                         ignore_index = True)


    step_detail.columns = step_summary.columns
    step_detail = step_detail.sort_values("Significance",ascending=True).reset_index()
    select_pred = step_detail.loc[0,"Entered"]
    select_index = step_detail.loc[0,"index"]
    select_cand = cand[step_detail.iloc[0,0]]
    step_detail.drop("index",axis=1,inplace=True)
```

```
    if step_detail.iloc[0,6] < 0.05:

        CandidateFeature.remove(select_pred)
        cand.pop(select_index)
        row = step_detail.iloc[0]
        step_summary = step_summary.append(row, ignore_index = True)
        llk_0 = row.iloc[3]
        df_0 = row.iloc[2]
        X_train = X_train.join(select_cand)

    else:
        break

    step_detail = pandas.DataFrame()
step_summary["Entered"] = "+ " + step_summary["Entered"]
```

In [35]:

```
step_summary
```

Out[35]:

	Step	Entered	Num of non-aliased	Log-likelihood	Deviance chi-sqaures	Degree of freedom	Significance
0	0	+ Intercept	1	-1.7324418e+04	NaN	NaN	NaN
1	1	+ URBANICITY	2	-1.6480902e+04	1.6870313e+03	1.0000000e+00	0.0000000e+00
2	2	+ MVR_PTS	3	-1.6132234e+04	6.9733640e+02	1.0000000e+00	1.1348123e-153
3	3	+ CAR_AGE	4	-1.5898009e+04	4.6844946e+02	1.0000000e+00	6.9691259e-104
4	4	+ MSTATUS	5	-1.5720053e+04	3.5591160e+02	1.0000000e+00	2.1869339e-79
5	5	+ CAR_TYPE	10	-1.5539157e+04	3.6179176e+02	5.0000000e+00	5.0584841e-76
6	6	+ HOMEKIDS	15	-1.5392207e+04	2.9390050e+02	5.0000000e+00	2.0505120e-61
7	7	+ REVOKED	16	-1.5265933e+04	2.5254762e+02	1.0000000e+00	7.2284812e-57
8	8	+ TRAVTIME	17	-1.5160198e+04	2.1147065e+02	1.0000000e+00	6.5617071e-48
9	9	+ TIF	18	-1.5100155e+04	1.2008501e+02	1.0000000e+00	6.0606924e-28
10	10	+ KIDSDRIV	22	-1.5059492e+04	8.1327846e+01	4.0000000e+00	9.1126095e-17

b). What predictors does your final model contain?

All the predictors. ie, Intercept + URBANICITY + MVR_PTS + CAR_AGE + MSTATUS + CAR_TYPE + HOMEKIDS + REVOKED + TRAVTIME + TIF + KIDSDRIV

c). What are the aliased parameters in your final model? Please list the predictor's name and the aliased categories.

In [36]:

```
print("Num of aliased parameter:", len(outList[1][outList[1].sum()==0].index))
print("\n")
print("Aliased categories:", list(outList[1][outList[1].sum()==0].index))
print("\n")
print("Predictor's name:",["URBANICITY","MVR_PTS","CAR_TYPE","HOMEKIDS","REVOKED","KIDSDRIV"])
```

```
Num of aliased parameter: 6


Aliased categories: ['URBANICITY_Highly Urban/ Urban', 'MSTATUS_Yes', 'CAR_TYPE_SUV', 'HOMEKIDS_0', 'REVOKED_No', 'KIDSDRIV_0']


Predictor's name: ['URBANICITY', 'MVR_PTS', 'CAR_TYPE', 'HOMEKIDS', 'REVOKED', 'KIDSDRIV']
```

d) How many non-aliased parameters are in your final model?

In [37]:

```
# There are 22 non-aliased parameter in the final model.
print("Num of non-aliased parameter:",len(outList[4]))
```

```
Num of non-aliased parameter: 22
```

e) Please show a table of the complete set of parameters of your final model (including the aliased parameters). Besides the parameter estimates, please also include the standard errors, and the 95% asymptotic confidence intervals. Conventionally, aliased parameters have missing standard errors and confidence intervals.

In [38]:
```python
outList = Regression.PoissonModel(X_train, y_train, o_train)
outList[0]
```

Out[38]:

	Estimate	Standard Error	Lower 95% CI	Upper 95% CI	Exponentiated
Intercept	-3.0899055e-01	4.4542990e-02	-3.9629321e-01	-2.2168790e-01	7.3418770e-01
URBANICITY_Highly Rural/ Rural	-1.7747507e+00	5.4664122e-02	-1.8818904e+00	-1.6676109e+00	1.6952572e-01
URBANICITY_Highly Urban/ Urban	0.0000000e+00	0.0000000e+00	0.0000000e+00	0.0000000e+00	1.0000000e+00
MVR_PTS	8.6651950e-02	4.4411743e-03	7.7947409e-02	9.5356492e-02	1.0905171e+00
CAR_AGE	-3.9096222e-02	2.1075227e-03	-4.3226890e-02	-3.4965553e-02	9.6165817e-01
MSTATUS_No	4.5801472e-01	2.2882576e-02	4.1316569e-01	5.0286374e-01	1.5809323e+00
MSTATUS_Yes	0.0000000e+00	0.0000000e+00	0.0000000e+00	0.0000000e+00	1.0000000e+00
CAR_TYPE_Panel Truck	4.4773672e-02	4.4359555e-02	-4.2169458e-02	1.3171680e-01	1.0457911e+00
CAR_TYPE_Van	-1.2597749e-02	4.3077496e-02	-9.7028090e-02	7.1832592e-02	9.8748127e-01
CAR_TYPE_Sports Car	1.6964520e-01	3.6059326e-02	9.8970218e-02	2.4032018e-01	1.1848844e+00
CAR_TYPE_Pickup	6.1091987e-02	3.2939073e-02	-3.4674093e-03	1.2565138e-01	1.0629967e+00
CAR_TYPE_Minivan	-4.7056408e-01	3.4298058e-02	-5.3778704e-01	-4.0334112e-01	6.2464982e-01
CAR_TYPE_SUV	0.0000000e+00	0.0000000e+00	0.0000000e+00	0.0000000e+00	1.0000000e+00
HOMEKIDS_5	5.6564534e-01	2.1978536e-01	1.3487395e-01	9.9641673e-01	1.7605836e+00
HOMEKIDS_4	2.1806020e-01	7.5268223e-02	7.0537196e-02	3.6558321e-01	1.2436619e+00
HOMEKIDS_3	3.0677995e-01	4.2503582e-02	2.2347446e-01	3.9008544e-01	1.3590419e+00
HOMEKIDS_1	3.1655783e-01	3.5695190e-02	2.4659654e-01	3.8651911e-01	1.3723956e+00
HOMEKIDS_2	2.7872042e-01	3.5187547e-02	2.0975409e-01	3.4768674e-01	1.3214378e+00
HOMEKIDS_0	0.0000000e+00	0.0000000e+00	0.0000000e+00	0.0000000e+00	1.0000000e+00
REVOKED_Yes	4.4035139e-01	2.8286839e-02	3.8491021e-01	4.9579258e-01	1.5532529e+00
REVOKED_No	0.0000000e+00	0.0000000e+00	0.0000000e+00	0.0000000e+00	1.0000000e+00
TRAVTIME	1.0300156e-02	7.2736499e-04	8.8745463e-03	1.1725765e-02	1.0103534e+00
TIF	-3.1354940e-02	2.9564320e-03	-3.7149440e-02	-2.5560440e-02	9.6913153e-01

	Estimate	Standard Error	Lower 95% CI	Upper 95% CI	Exponentiated
KIDSDRIV_4	-2.0173417e+01	7.9888483e+03	-1.5678028e+04	1.5637682e+04	1.7329904e-09
KIDSDRIV_3	3.6497635e-01	1.0704432e-01	1.5517335e-01	5.7477936e-01	1.4404799e+00
KIDSDRIV_2	3.8083803e-01	5.5367926e-02	2.7231889e-01	4.8935717e-01	1.4635105e+00
KIDSDRIV_1	2.5083678e-01	4.0376690e-02	1.7169992e-01	3.2997364e-01	1.2851003e+00
KIDSDRIV_0	0.0000000e+00	0.0000000e+00	0.0000000e+00	0.0000000e+00	1.0000000e+00

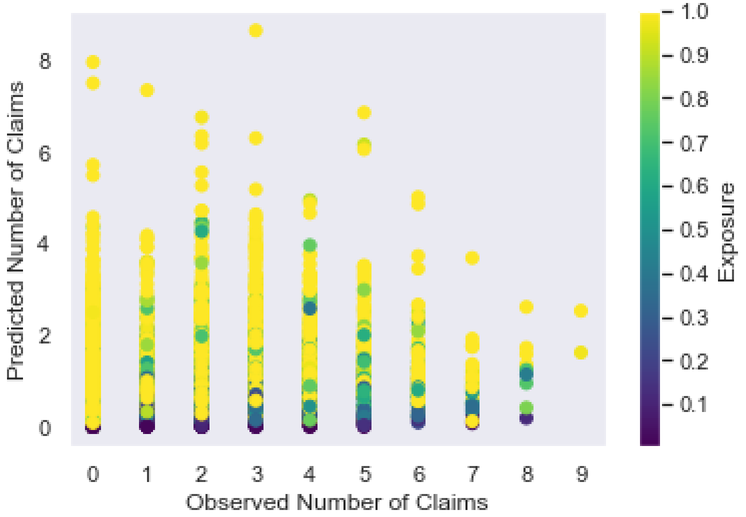
Question 3

In [39]:
```
y_pred = outList[6]
trainData.insert(12, column = "y_pred", value = y_pred)
```

a). Please plot the predicted number of claims versus the observed number of claims.

In [58]:

```
plt.figure(figsize=(6,4))
sg = plt.scatter(trainData['CLM_COUNT'], trainData['y_pred'], c = e_train, marker = 'o',cmap = 'viridis')
plt.xlabel('Observed Number of Claims')
plt.ylabel('Predicted Number of Claims')
plt.xticks(range(10))
plt.grid(axis = 'both')
cbar = plt.colorbar(sg, label = 'Exposure')
cbar.set_ticks(numpy.arange(0.0, 1.1, 0.1))
plt.show()
```

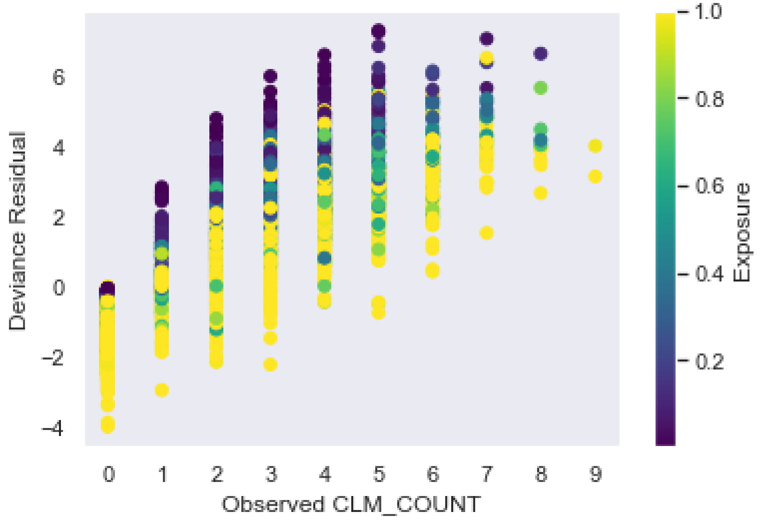


b). Please plot the Deviance residuals versus the observed number of claims.

In [57]:

```
dR2 = numpy.nan_to_num(y_train * numpy.log(y_train/y_pred)) - (y_train - y_pred)
devResid = numpy.where(y_train > y_pred, 1.0, -1.0) * numpy.where(dR2 > 0.0, numpy.sqrt(2.0 * dR2), 0.0)

plt.figure(figsize=(6,4))
sg = plt.scatter(trainData['CLM_COUNT'], devResid, c = e_train, marker = 'o',cmap = 'viridis')
plt.xlabel('Observed CLM_COUNT')
plt.ylabel('Deviance Residual')
plt.xticks(range(10))
plt.grid(axis = 'both')
plt.colorbar(sg, label = 'Exposure')
plt.show();
```



Question 4

a). Please calculate the Root Mean Squared Error, the Relative Error, and the R-squared metrics.

In [45]:

```
# Root Mean Squared Error
y_resid = y_train - y_pred
print('Sum of Residuals = ', numpy.sum(y_resid))

mse = numpy.sum(numpy.power(y_resid, 2)) / n_sample

rmse = numpy.sqrt(mse)
print('Root Mean Squared Error = ', rmse)

# Relative Error
relerr = mse / numpy.var(y_train, ddof = 0)
print('Relative Error = ', relerr)

sqcor = numpy.power(numpy.corrcoef(y_train, y_pred), 2)
print('Squared Correlation = ', sqcor[0,1])
```

```
Sum of Residuals =  -5.764832167187706e-09
Root Mean Squared Error =  1.5111521924451772
Relative Error =  0.9877772709997402
Squared Correlation =  0.06306347805003958
```

b). Please comment on the Final Model based on the above three metrics and the diagnostic charts in Question 3

RMSE is 1.511, which measures the size of a typical difference between a predicted target value and an observed target value.The smaller the metric, the better the model is trained, but it's actually difficult to determine what a small metric should be.

Relative Error is 0.9878, which the model performance between a saturated model and an uniformative model. Since it's very close to 1, I would say most of efforts are in vain since it's just improved a little bit from an uniformative model (relative error is 1)

Squared correlation is 0.0631, which measures the Pearson correlation between the observed target values and the predicted value. The larger the R2 metric, the better the model is trained, so we can tell the model is not trained pretty well, very close to uniformative model.

From the first scatterplot of q3 , we don't see any strong positive correlation between predicted number of claims versus the observed number of claims, meaning that the model predictions is not good at all.

From the second scatterplot of q3, we can see the deviance residual is positively correlated with observed number of claims. As the observed number of claims goes up, the deviance residual is increasing meaning the difference of likelihoods between the fitted model and the saturated model becomes bigger and bigger(worse and worse fit). Also, the exposure is always at the top of each bucket, relatively small proportion of 0 and big numbers 8 and 9, high proportion in 2-7 CLM_COUNT.