GAM CAGNY 2021

Sam Kloese

11/29/2021

Load Packages

Preliminary Adjustments

```
set.seed(23) # for reproducibility

# Take a look at our data
glimpse(pg17trainclaim)
```

```
## Rows: 14,243
## Columns: 6
## $ id_client    <fct> A00000009, A00000016, A00000026, A00000040, A00000056, A0~
## $ id_vehicle   <fct> V01, V01, V01, V01, V01, V01, V01, V01, V01, V01, V01, V0~
## $ id_year      <fct> Year 0, Year 0, Year 0, Year 0, Year 0, Year 0, Year 0, Y~
## $ id_claim     <fct> CL01, CL01, CL01, CL01, CL01, CL01, CL01, CL01, CL01, CL0~
## $ claim_nb     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ claim_amount <dbl> 927.16, 555.48, 478.01, 512.83, 1236.00, 158.28, -477.91,~
```

```
glimpse(pg17trainpol)
```

```
## Rows: 100,000
## Columns: 31
## $ id_client        <fct> A00000001, A00000002, A00000003, A00000004, A00000005~
## $ id_vehicle       <fct> V01, V01, V01, V01, V01, V01, V01, V01, V01, V01, V01~
## $ id_policy        <fct> A00000001-V01, A00000002-V01, A00000003-V01, A0000000~
## $ id_year          <fct> Year 0, Year 0, Year 0, Year 0, Year 0, Year 0, Year ~
## $ pol_bonus        <dbl> 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.50, 0.64,~
## $ pol_coverage     <fct> Maxi, Maxi, Maxi, Median2, Maxi, Median1, Maxi, Maxi,~
## $ pol_duration     <int> 29, 3, 2, 22, 16, 5, 5, 2, 5, 26, 8, 4, 21, 25, 9, 6,~
## $ pol_sit_duration <int> 9, 1, 2, 1, 4, 1, 3, 2, 1, 6, 1, 4, 1, 8, 1, 2, 3, 3,~
## $ pol_pay_freq     <fct> Biannual, Biannual, Yearly, Yearly, Biannual, Monthly~
## $ pol_payd         <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, Yes, ~
## $ pol_usage        <fct> Retired, Retired, WorkPrivate, WorkPrivate, Retired, ~
## $ pol_insee_code   <fct> 36233, 92073, 92026, 78537, 38544, 76259, 38547, 3712~
## $ drv_drv2         <fct> No, No, No, Yes, Yes, No, No, No, No, Yes, Yes, No, N~
## $ drv_age1         <int> 85, 69, 37, 81, 62, 68, 77, 64, 38, 59, 66, 61, 65, 7~
## $ drv_age2         <int> 0, 0, 0, 21, 68, 0, 0, 0, 0, 33, 32, 0, 0, 0, 34, 56,~
## $ drv_sex1         <fct> M, M, M, M, F, M, M, M, M, M, M, M, F, M, M, M, F, M,~
## $ drv_sex2         <fct> , , , F, M, , , , , F, M, , , , F, F, , , , , M, F, ,~
```

```
## $ drv_age_lic1     <int> 62, 39, 18, 54, 37, 40, 55, 37, 19, 41, 45, 43, 43, 4~
## $ drv_age_lic2     <int> 0, 0, 0, 3, 48, 0, 0, 0, 0, 15, 14, 0, 0, 0, 14, 37, ~
## $ vh_age           <int> 10, 4, 11, 16, 11, 14, 7, 11, 9, 6, 4, 5, 5, 13, 1, 2~
## $ vh_cyl           <int> 1587, 2149, 1991, 1781, 1598, 1769, 1870, 1595, 1997,~
## $ vh_din           <int> 98, 170, 150, 90, 108, 60, 108, 101, 109, 90, 90, 127~
## $ vh_fuel          <fct> Gasoline, Diesel, Gasoline, Gasoline, Gasoline, Diese~
## $ vh_make          <fct> PEUGEOT, MERCEDES BENZ, BMW, VOLKSWAGEN, RENAULT, PEU~
## $ vh_model         <fct> 306, C220, Z3, GOLF, LAGUNA, 205, LAGUNA, A4, 307, PA~
## $ vh_sale_begin    <int> 10, 4, 12, 18, 13, 28, 10, 16, 9, 9, 4, 6, 7, 14, 3, ~
## $ vh_sale_end      <int> 9, 2, 11, 15, 11, 18, 6, 13, 7, 7, 3, 3, 4, 13, 1, 4,~
## $ vh_speed         <int> 182, 229, 210, 180, 195, 155, 193, 191, 183, 163, 180~
## $ vh_type          <fct> Tourism, Tourism, Tourism, Tourism, Tourism, Tourism,~
## $ vh_value         <int> 20700, 34250, 28661, 14407, 16770, 11564, 22450, 2053~
## $ vh_weight        <int> 1210, 1510, 1270, 1020, 1230, 850, 1350, 1195, 1260, ~
```

```
# Assemble data to model

# Some clients had more than 1 claim in a year
pg17trainclaim2 <- pg17trainclaim %>% # Aggregate claims to client and year
  group_by(id_client, id_year) %>%
  summarize(claim_count = n(),
            claim_amount = sum(claim_amount))
```

```
## `summarise()` has grouped output by 'id_client'. You can override using the `.groups` argument.
```

```
# Join the policy information and claims data
# If the client can't be found in the claims data, they had 0 claims for $0
pg17train <- pg17trainpol %>%
  left_join(pg17trainclaim2, by = c("id_client", "id_year")) %>%
  mutate(claim_count = replace_na(claim_count, replace = 0)) %>%
  mutate(claim_amount = replace_na(claim_amount, replace = 0)) %>%
  mutate(exposures = 1) %>% # Big assumption: All years are full years %>%
  mutate(drv_age1 = as.double(drv_age1)) %>% # Cap driver age between 25 and 75
  mutate(drv_age1 = case_when(drv_age1 > 75 ~ 75,
                              drv_age1 < 25 ~ 25,
                              TRUE ~ drv_age1)) %>%
  mutate(vh_age = as.double(vh_age)) %>% # Cap vehicle age at 20
  mutate(vh_age = case_when(vh_age > 20 ~ 20,
                            TRUE ~ vh_age)) %>%
  mutate(vh_din = as.double(vh_din)) %>% # Cap Metric HP at 200
  mutate(vh_din = case_when(vh_din > 200 ~ 200,
                            TRUE ~ vh_din))


dim(pg17train)
```

```
## [1] 100000     34
```

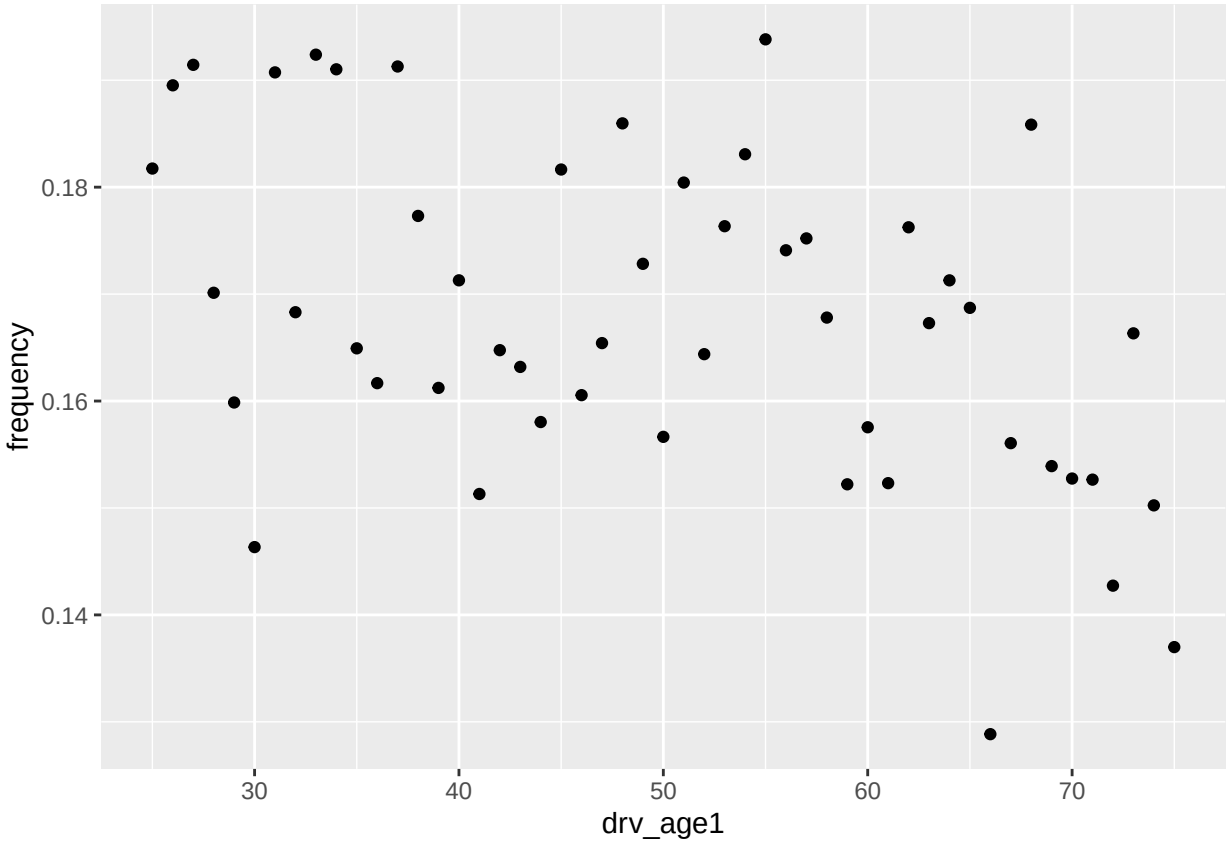
```
sum(pg17train$claim_count)
```

```
## [1] 16445
```

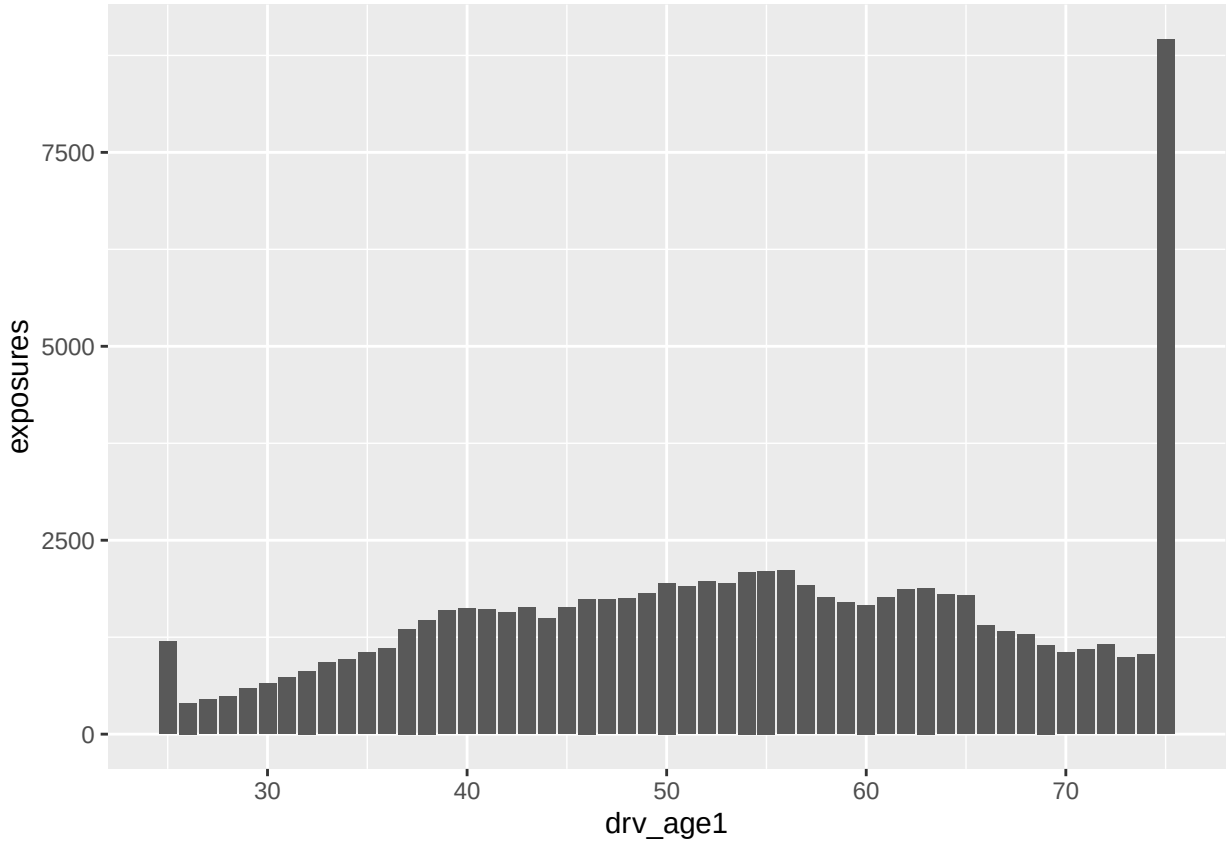
```r
# Remove record with NA's
pg17train2 <- pg17train[complete.cases(pg17train),]

# 80% of clients will be used in training
# 20% of clients will be used in testing
clients_unique <- unique(pg17train2$id_client)
clients_index <- sample(1:91488,
                        size = 73190,
                        replace = FALSE)
clients_train <- clients_unique[clients_index]

training_data <- pg17train2 %>%
  filter(id_client %in% clients_train)
testing_data <- pg17train2 %>%
  filter(!(id_client %in% clients_train))

rm(pg17train, pg17train2, pg17trainclaim, pg17trainclaim2,pg17trainpol,
   clients_index, clients_train, clients_unique)
```

Plots of actual frequency

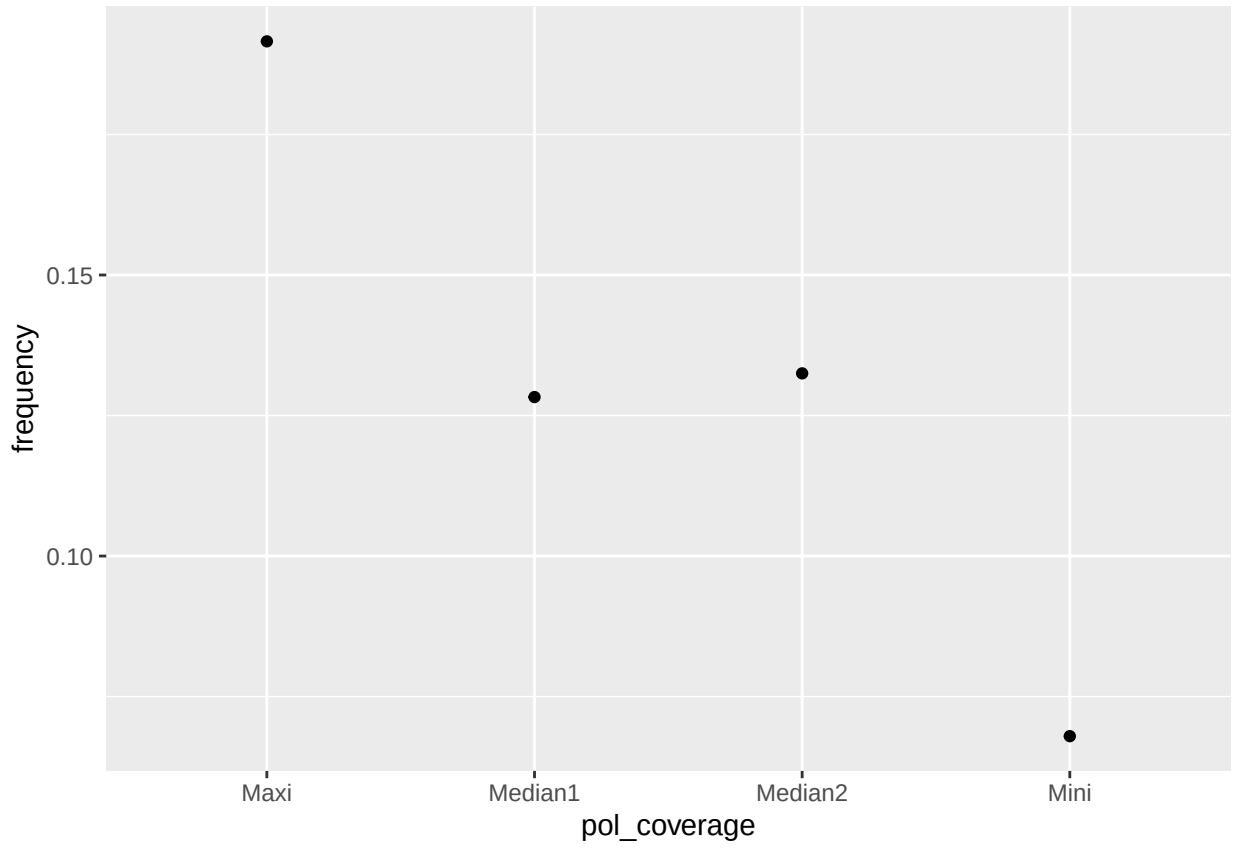
```r
# Plots by Driver Age
driver_age_summary <- training_data %>%
  group_by(drv_age1) %>%
  summarize(claims = sum(claim_count),
            exposures = sum(exposures)) %>%
  ungroup() %>%
  mutate(frequency = claims/exposures)

ggplot(aes(x = drv_age1), data = driver_age_summary) +
  geom_point(aes(y = frequency))
```



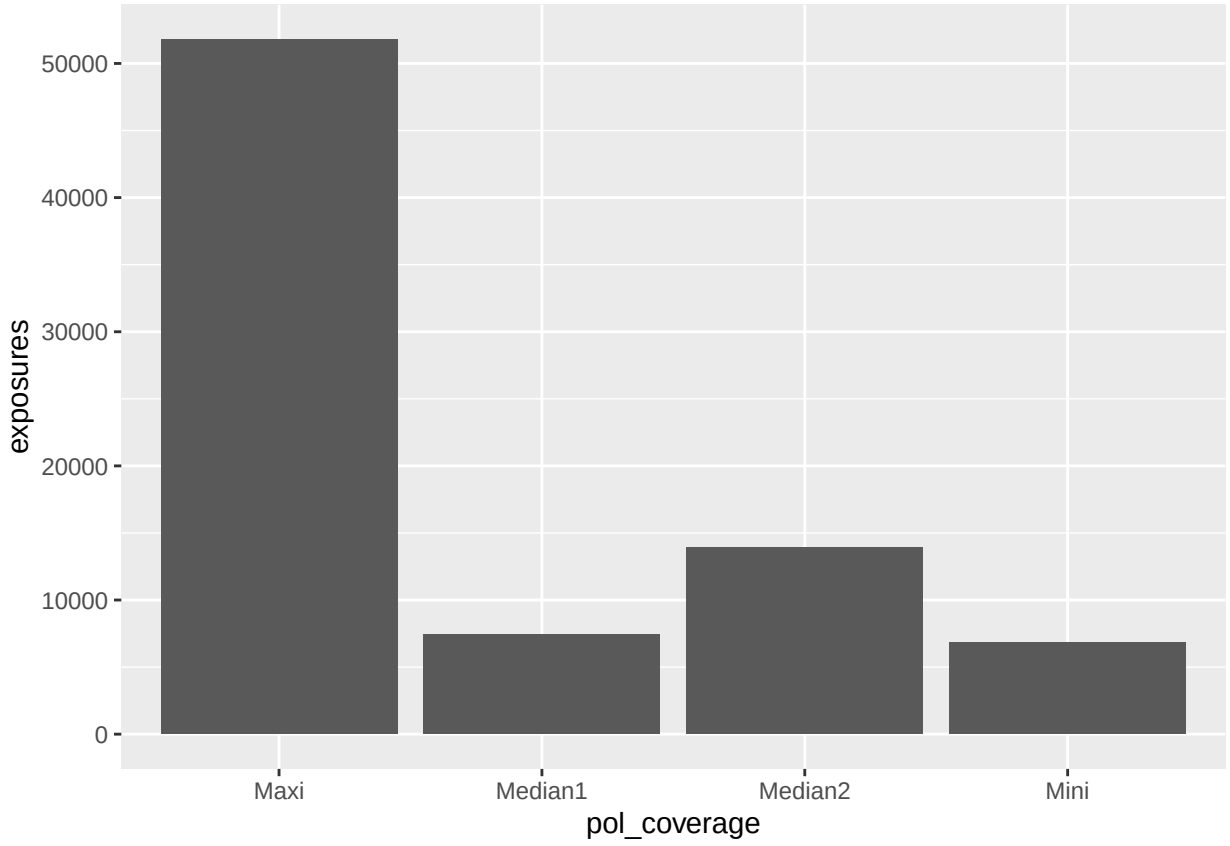

```
ggplot(aes(x = drv_age1), data = driver_age_summary) +
  geom_col(aes(y = exposures))
```



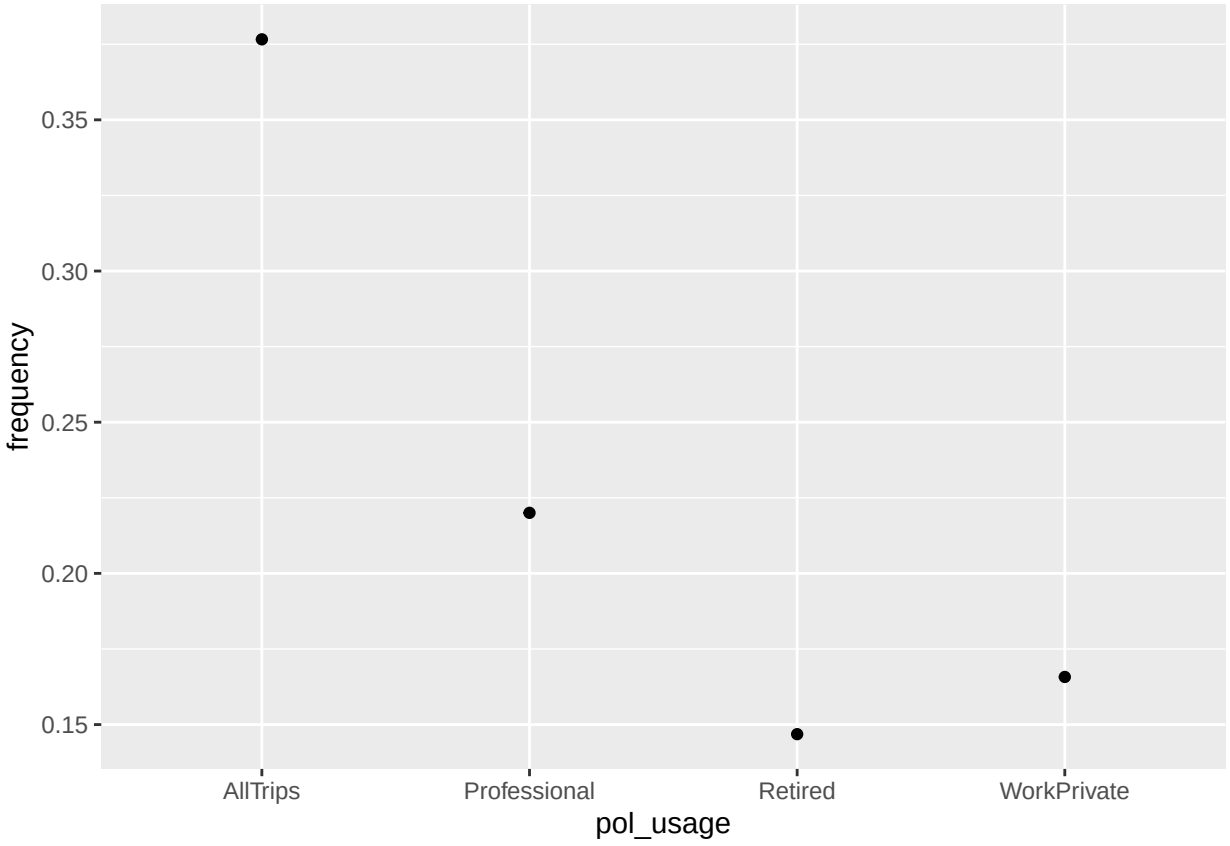

```
# Plots by Coverage Type
pol_coverage_summary <- training_data %>%
  group_by(pol_coverage) %>%
  summarize(claims = sum(claim_count),
            exposures = sum(exposures)) %>%
  ungroup() %>%
  mutate(frequency = claims/exposures)
ggplot(aes(x = pol_coverage), data = pol_coverage_summary) +
  geom_point(aes(y = frequency))
```



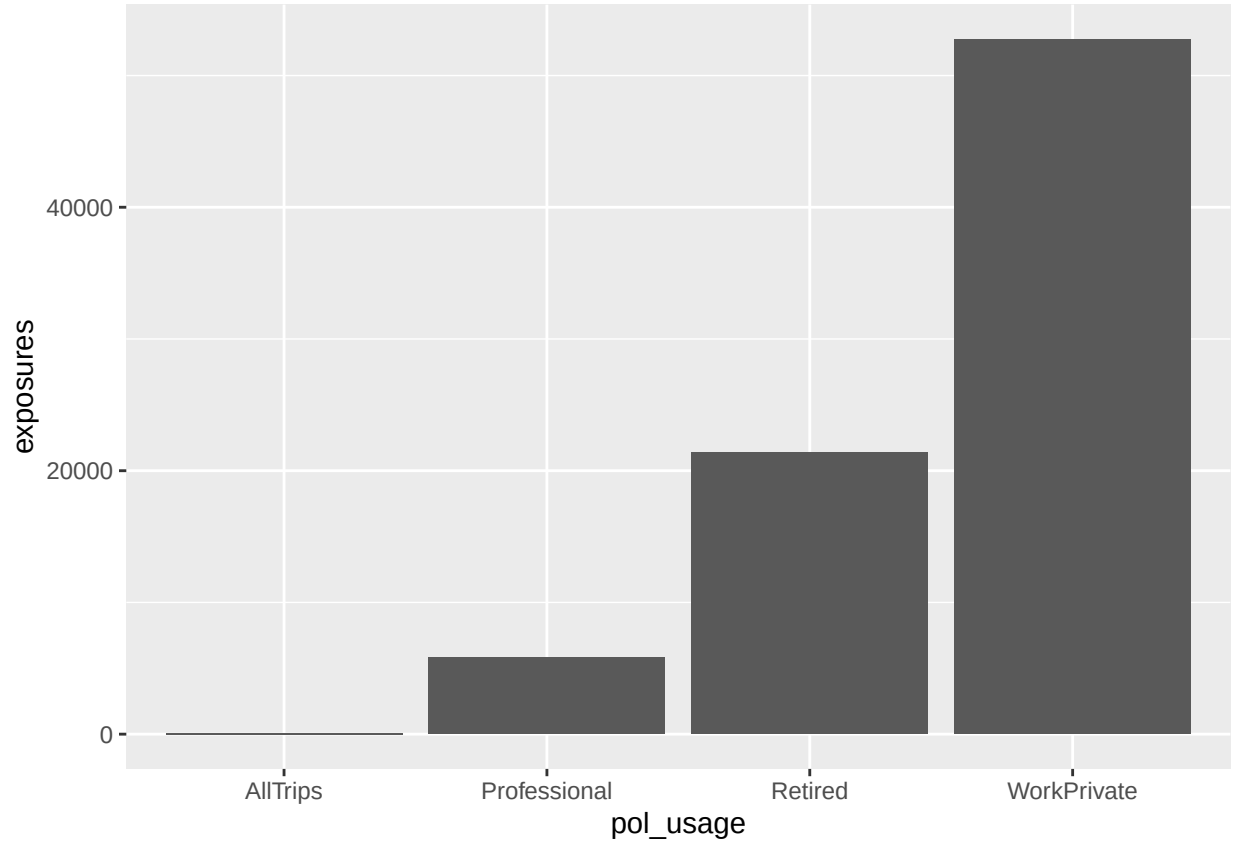

```
ggplot(aes(x = pol_coverage), data = pol_coverage_summary) +
  geom_col(aes(y = exposures))
```



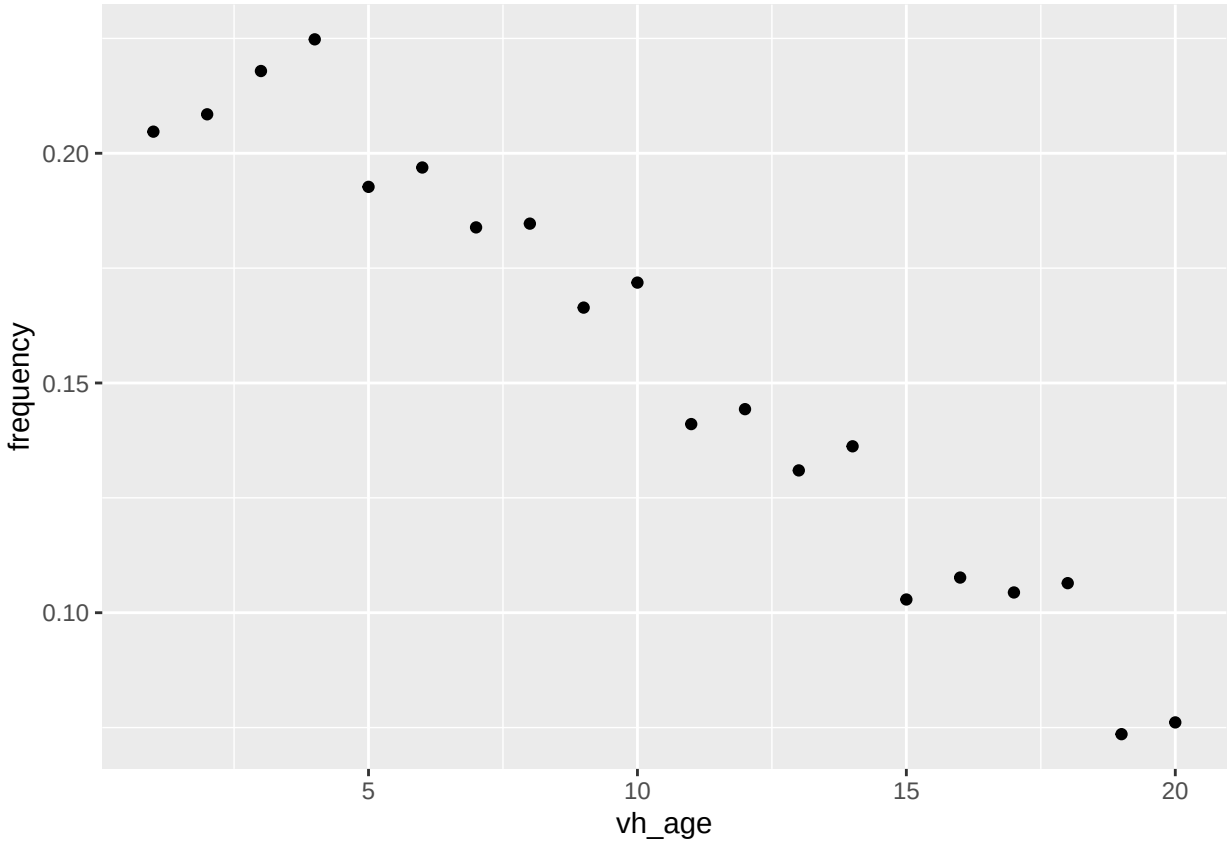

```
# Plots by Policy Usage
pol_usage_summary <- training_data %>%
  group_by(pol_usage) %>%
  summarize(claims = sum(claim_count),
            exposures = sum(exposures)) %>%
  ungroup() %>%
  mutate(frequency = claims/exposures)
ggplot(aes(x = pol_usage), data = pol_usage_summary) +
  geom_point(aes(y = frequency))
```



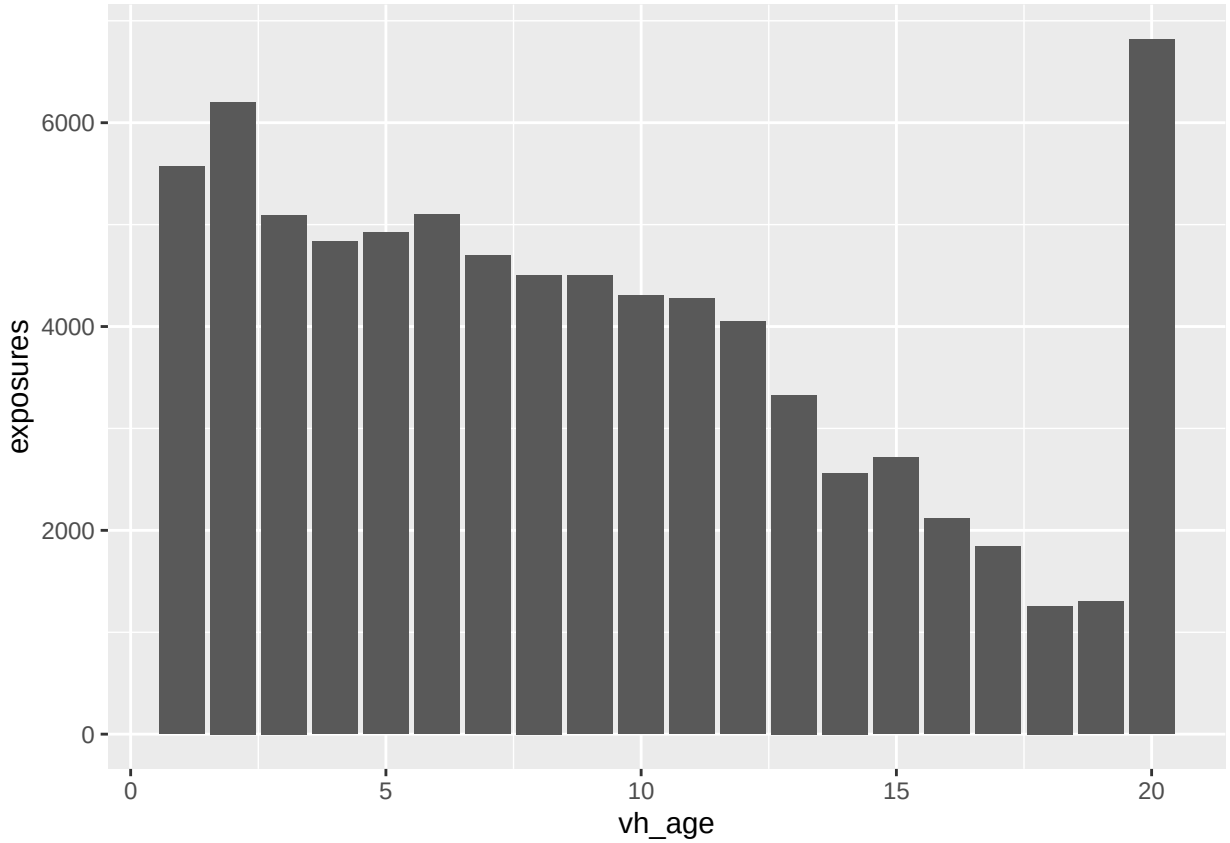

```
ggplot(aes(x = pol_usage), data = pol_usage_summary) +
  geom_col(aes(y = exposures))
```



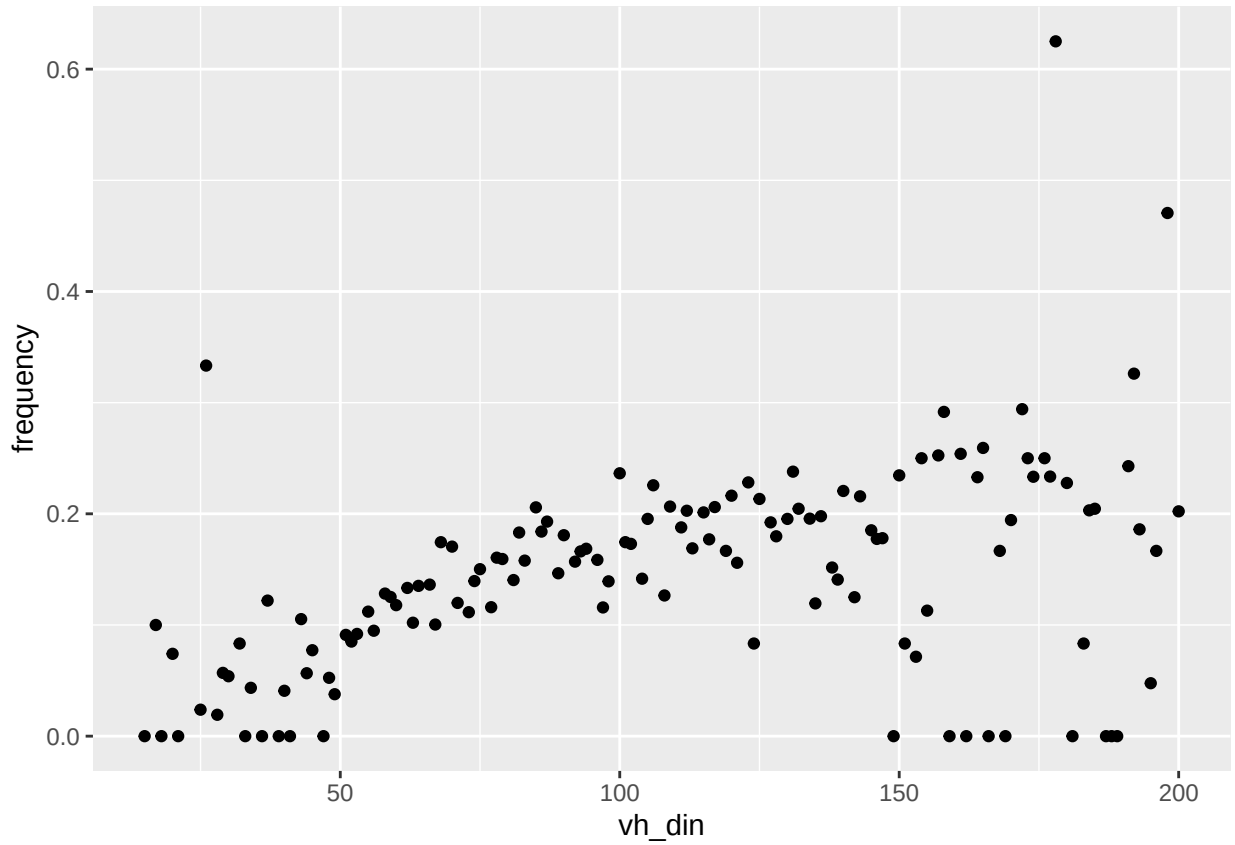

```
# Plots by Vehicle Age
vh_age_summary <- training_data %>%
  group_by(vh_age) %>%
  summarize(claims = sum(claim_count),
            exposures = sum(exposures)) %>%
  ungroup() %>%
  mutate(frequency = claims/exposures)
ggplot(aes(x = vh_age), data = vh_age_summary) +
  geom_point(aes(y = frequency))
```



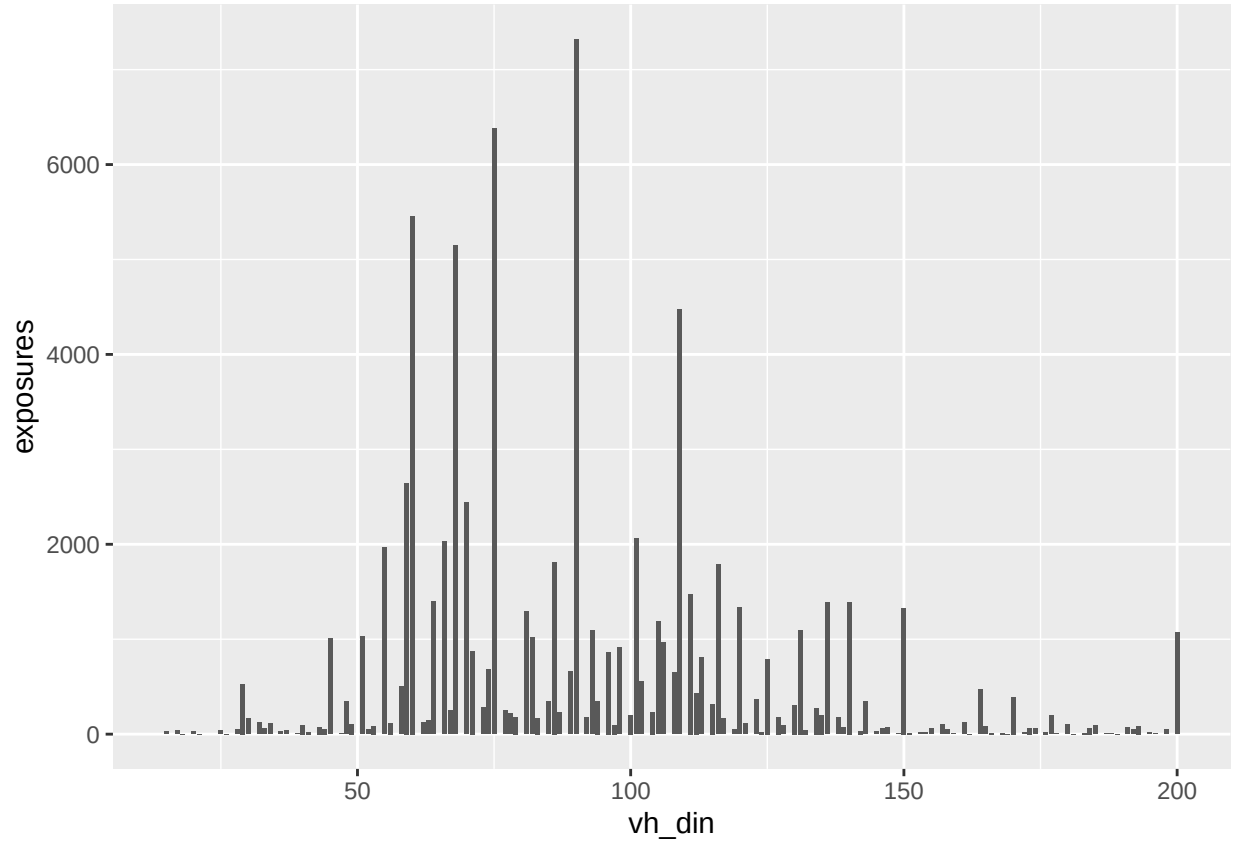

```
ggplot(aes(x = vh_age), data = vh_age_summary) +
  geom_col(aes(y = exposures))
```




```
# Plots by Vehicle DIN
vh_din_summary <- training_data %>%
  group_by(vh_din) %>%
  summarize(claims = sum(claim_count),
            exposures = sum(exposures)) %>%
  ungroup() %>%
  mutate(frequency = claims/exposures)
ggplot(aes(x = vh_din), data = vh_din_summary) +
  geom_point(aes(y = frequency))
```




```
ggplot(aes(x = vh_din), data = vh_din_summary) +
  geom_col(aes(y = exposures))
```



Build GAM

```
gam_final <- gam(claim_count ~ pol_coverage + pol_usage +
                   s(drv_age1, k = 4) + s(vh_age, k = 4) +
                   te(vh_din, vh_weight, k = 3),
                 family = poisson(link = "log"),
                 offset = log(exposures),
                 data = training_data)

coef(gam_final)
```

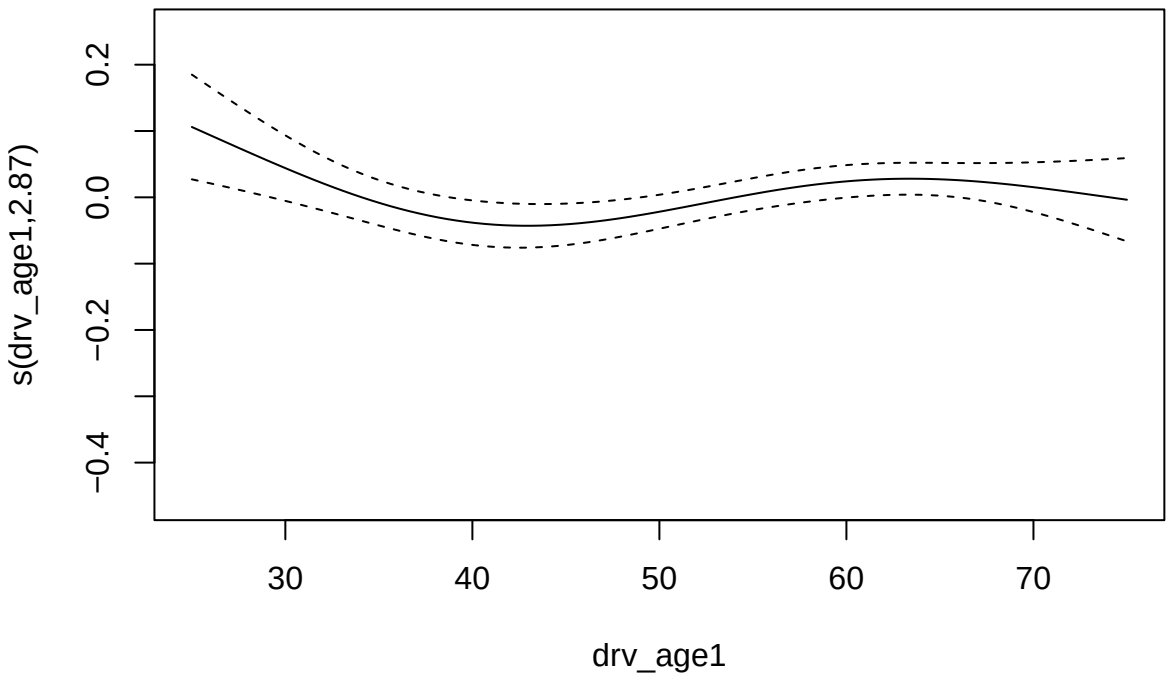
```
##            (Intercept)    pol_coverageMedian1    pol_coverageMedian2
##            -1.17695690            -0.05899137            -0.13773872
##       pol_coverageMini  pol_usageProfessional       pol_usageRetired
##            -0.59876697            -0.40513754            -0.71977724
##   pol_usageWorkPrivate          s(drv_age1).1          s(drv_age1).2
##            -0.59132851             0.10851243            -0.13639530
##          s(drv_age1).3            s(vh_age).1            s(vh_age).2
##            -0.11479283            -0.03900515             0.08484580
##            s(vh_age).3 te(vh_din,vh_weight).1 te(vh_din,vh_weight).2
##            -0.16400605            -0.21332205             0.94207594
## te(vh_din,vh_weight).3 te(vh_din,vh_weight).4 te(vh_din,vh_weight).5
##             0.26409347             0.73909859             0.63741502
```

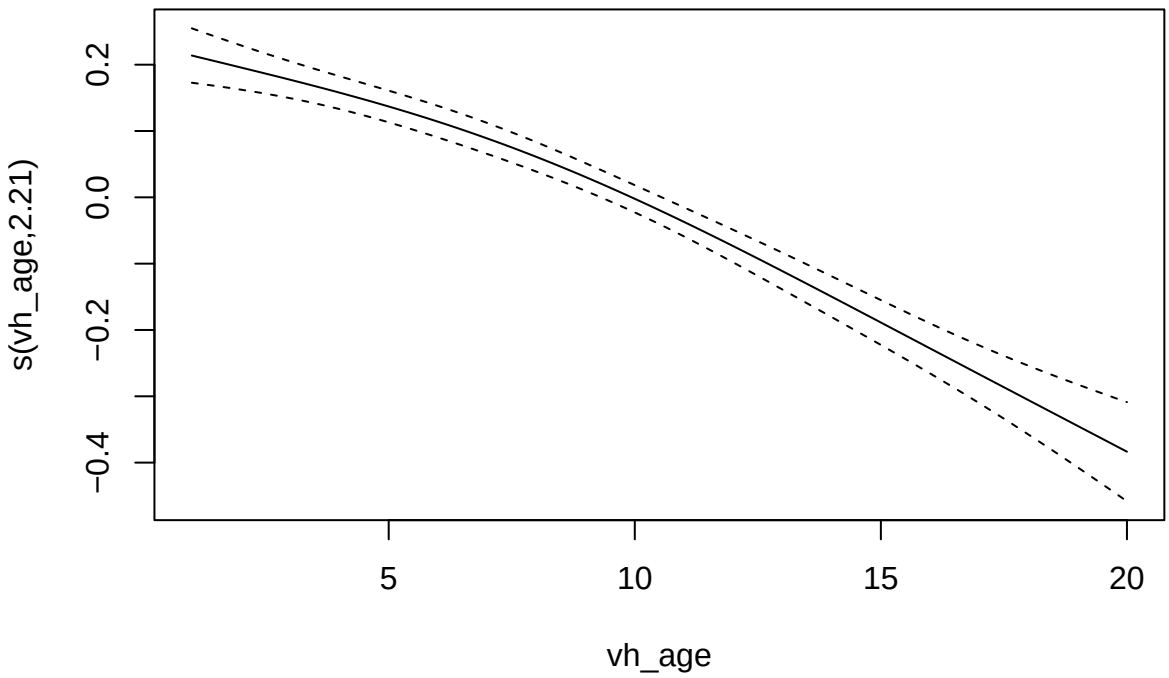
```
## te(vh_din,vh_weight).6 te(vh_din,vh_weight).7 te(vh_din,vh_weight).8
##            -0.49104169             0.14769891             0.42128466
```

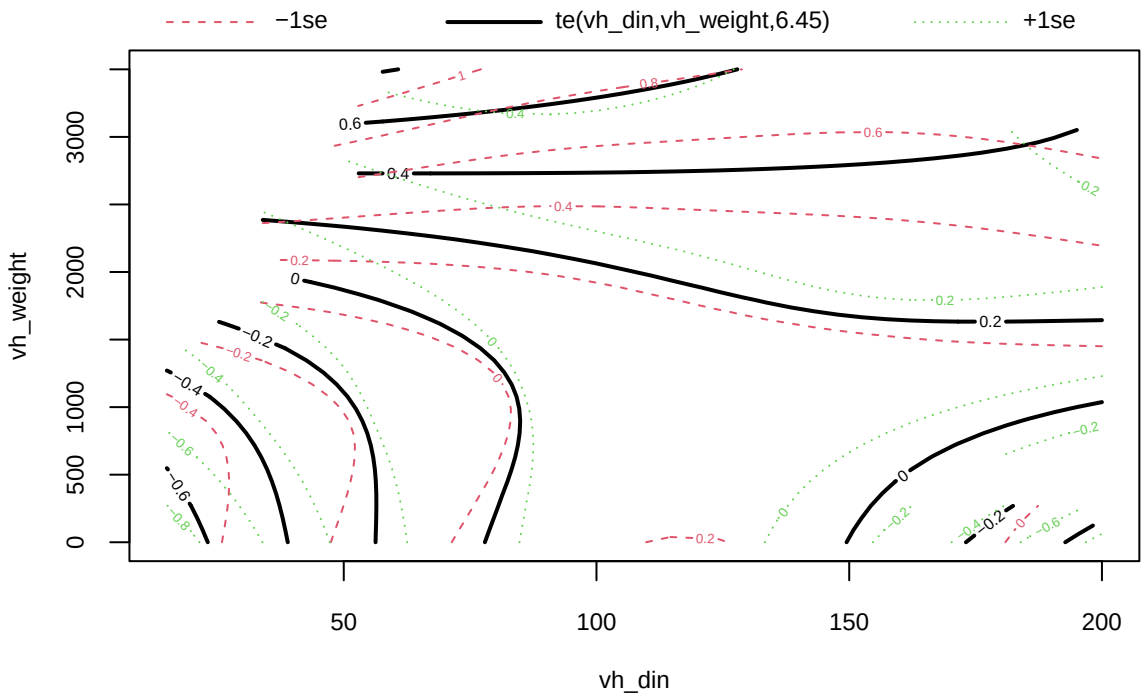
```
summary(gam_final)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## claim_count ~ pol_coverage + pol_usage + s(drv_age1, k = 4) +
##     s(vh_age, k = 4) + te(vh_din, vh_weight, k = 3)
##
## Parametric coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.17696    0.18626  -6.319 2.63e-10 ***
## pol_coverageMedian1   -0.05899    0.03944  -1.496 0.134755
## pol_coverageMedian2   -0.13774    0.02885  -4.775 1.80e-06 ***
## pol_coverageMini      -0.59877    0.05396 -11.097  < 2e-16 ***
## pol_usageProfessional -0.40514    0.18800  -2.155 0.031163 *
## pol_usageRetired      -0.71978    0.18835  -3.822 0.000133 ***
## pol_usageWorkPrivate  -0.59133    0.18624  -3.175 0.001498 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                         edf Ref.df Chi.sq p-value
## s(drv_age1)           2.870  2.988  11.75 0.00653 **
## s(vh_age)             2.207  2.591 173.96 < 2e-16 ***
## te(vh_din,vh_weight) 6.453  7.073 176.90 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0156   Deviance explained =  2.9%
## UBRE = -0.36299  Scale est. = 1         n = 79995
```

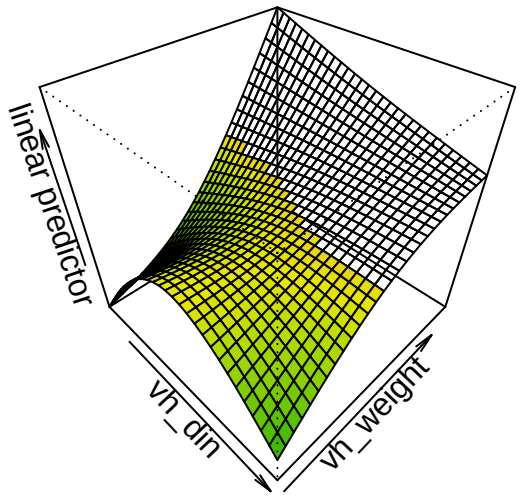
```
plot(gam_final)
```


s(drv_age1,2.87)
0.2
0.0
−0.2
−0.4
30
40
50
60
70
drv_age1

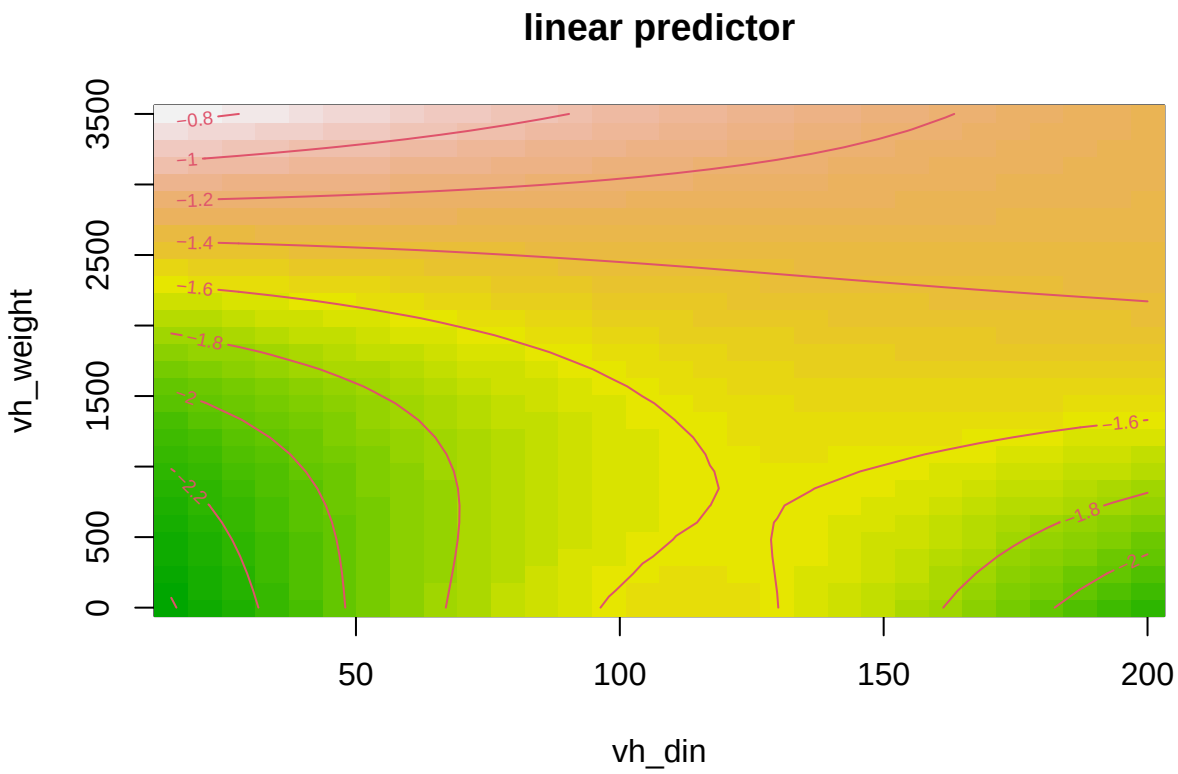

s(vh_age,2.21)
0.2
0.0
−0.2
−0.4
5
10
15
20
vh_age



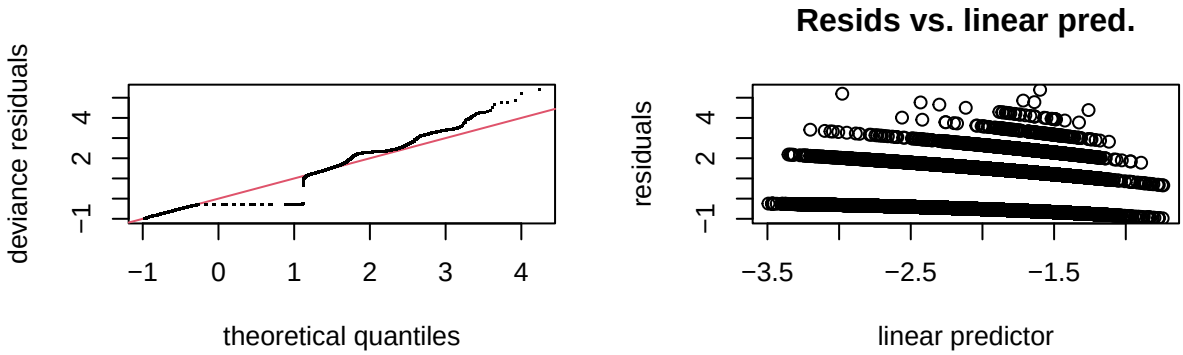

```
vis.gam(x = gam_final,
        view = c("vh_din","vh_weight"),
        plot.type = "persp",
        color = "terrain",
        theta = 45,
        phi = 40)
```

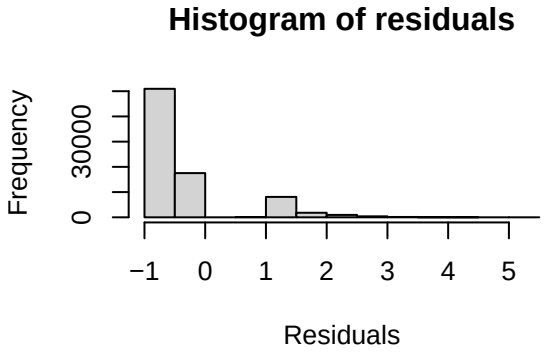


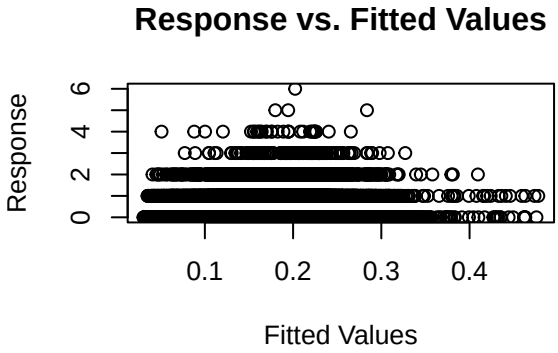

```
vis.gam(x = gam_final,
        view = c("vh_din","vh_weight"),
        plot.type = "contour",
        color = "terrain")
```




```
gam.check(gam_final)
```







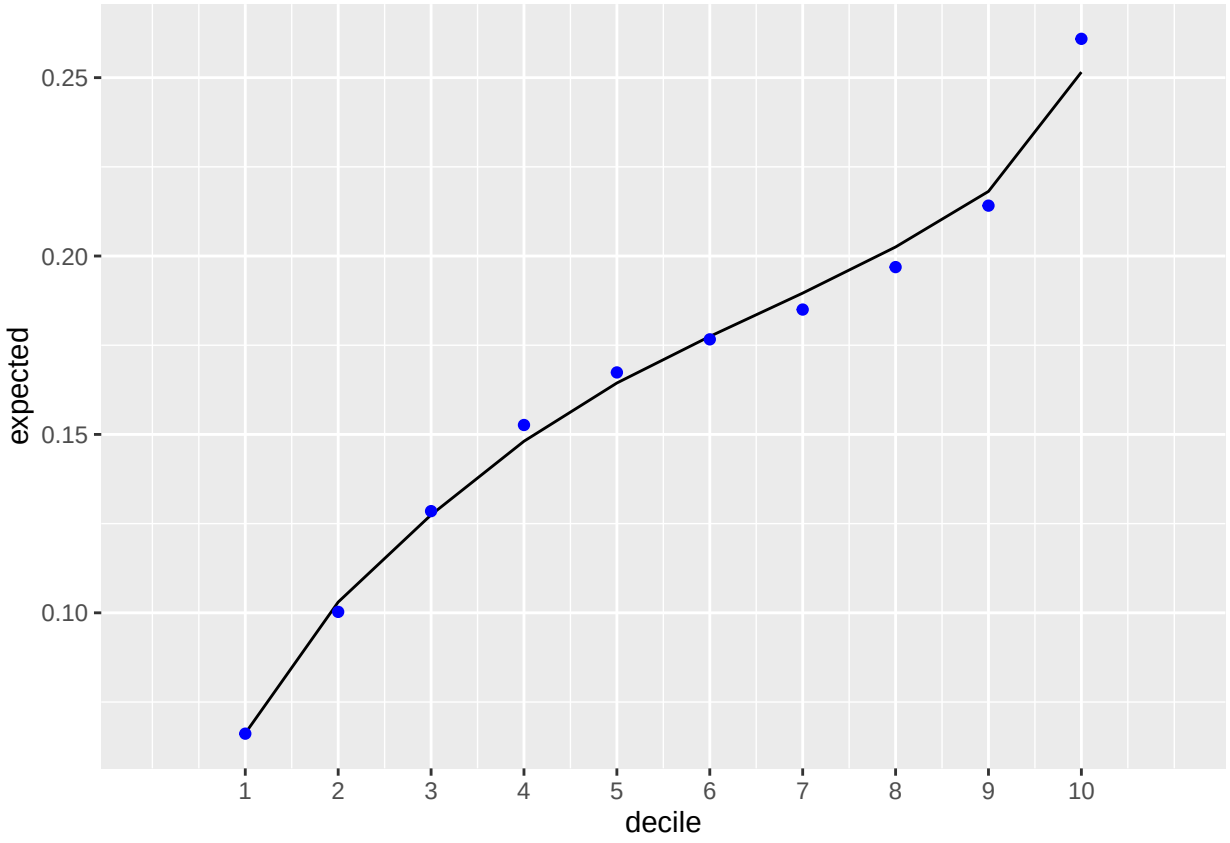

```
##
## Method: UBRE   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-3.138995e-08,5.176262e-08]
## (score -0.3629931 & scale 1).
## Hessian positive definite, eigenvalue range [2.694093e-06,8.104311e-06].
## Model rank =  21 / 21
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                        k'  edf k-index p-value
## s(drv_age1)          3.00 2.87    0.88   0.065 .
## s(vh_age)            3.00 2.21    0.91   0.760
## te(vh_din,vh_weight) 8.00 6.45    0.89   0.205
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
concurvity(gam_final, full = TRUE)
```

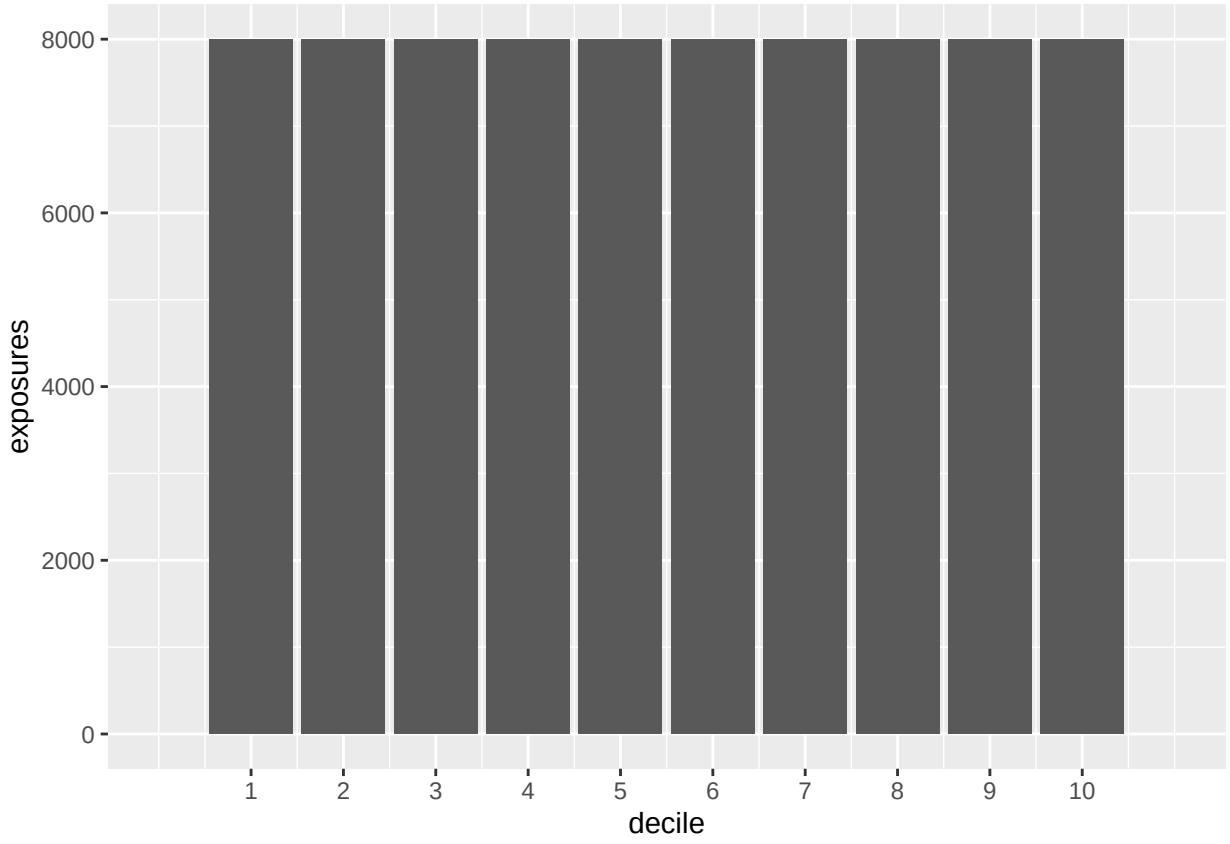
```
##               para s(drv_age1) s(vh_age) te(vh_din,vh_weight)
## worst    0.9990397  0.64082722 0.5583683            0.2826454
## observed 0.9990397  0.05038053 0.5504003            0.1831978
## estimate 0.9990397  0.42190878 0.5073782            0.1054095
```

Quantile Plot on Train Data

```
training_predictions = data.frame(predictions=predict(gam_final,
                                                      newdata = training_data,
                                                      type = "response"))
training_predictions <- training_data %>%
  mutate(predicted_claims = training_predictions$predictions) %>%
  mutate(predicted_frequency = predicted_claims / exposures)

total_exposures <- sum(training_predictions$exposures)
quantile_plot_data <- training_predictions %>%
  arrange(predicted_frequency) %>%
  mutate(cumulative_exposures = cumsum(exposures)) %>%
  mutate(decile = floor(cumulative_exposures/total_exposures*10)+1) %>%
  mutate(decile = if_else(decile > 10, 10, decile)) %>%
  group_by(decile) %>%
  summarize(actual = sum(claim_count) / sum(exposures),
  expected = sum(predicted_claims) / sum(exposures),
  exposures = sum(exposures))

ggplot(aes(x = decile),data = quantile_plot_data) +
geom_line(aes(y = expected)) +
  geom_point(aes(y = actual),color="blue") +
  scale_x_continuous(limits = c(0,11), breaks = seq(1,10,1))
```



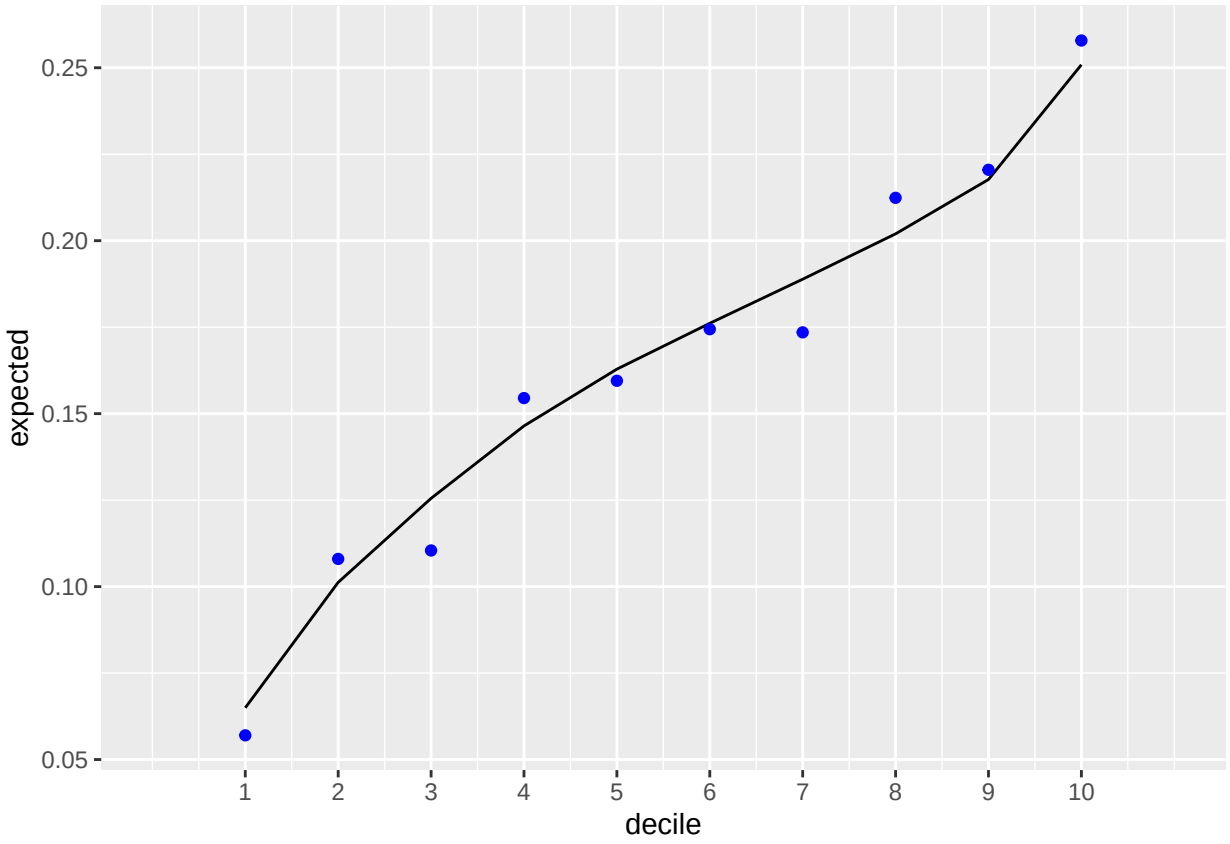
```
ggplot(aes(x = decile),data = quantile_plot_data) +
geom_col(aes(y = exposures)) +
    scale_x_continuous(limits = c(0,11), breaks = seq(1,10,1))
```



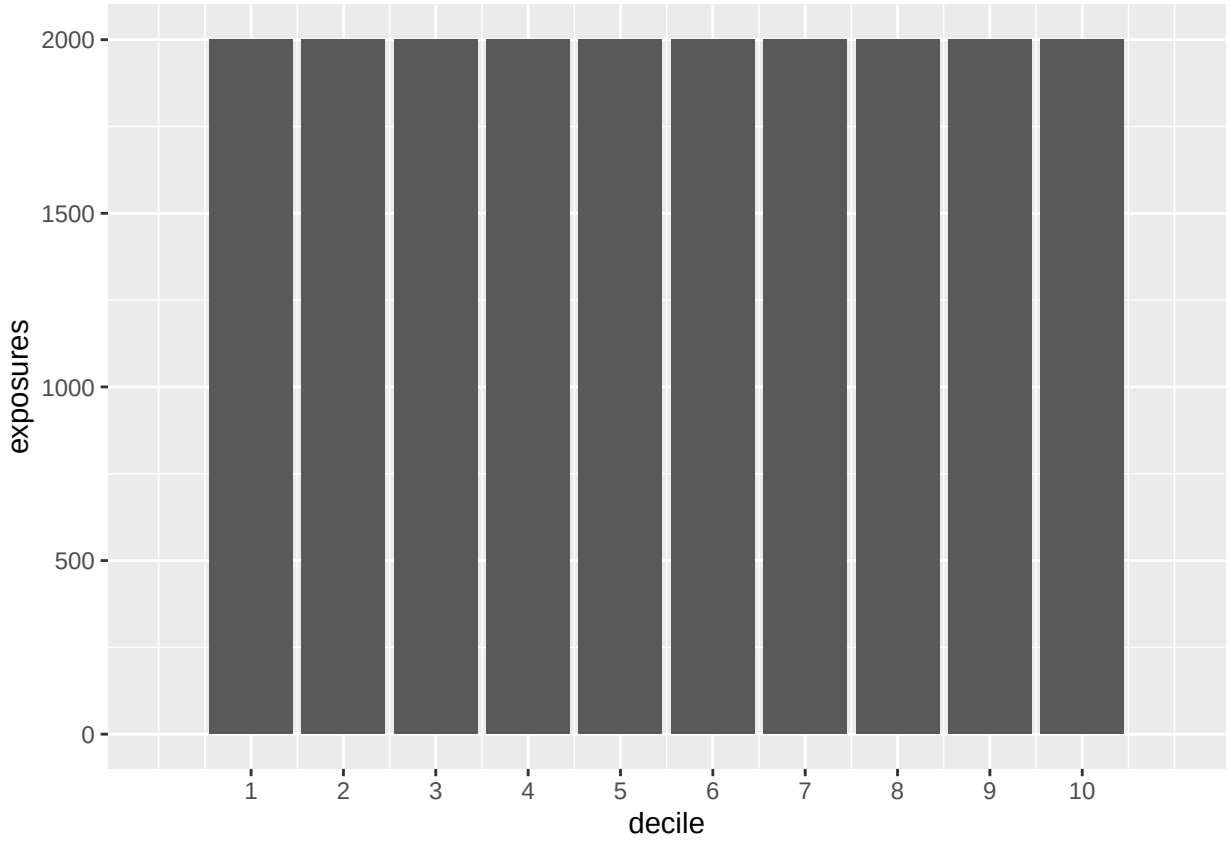
Quantile Plot on Test Data

```
testing_predictions = data.frame(predictions=predict(gam_final,
                                                     newdata = testing_data,
                                                     type = "response"))

testing_predictions <- testing_data %>%
  mutate(predicted_claims = testing_predictions$predictions) %>%
  mutate(predicted_frequency = predicted_claims / exposures)

total_exposures <- sum(testing_predictions$exposures)
quantile_plot_data <- testing_predictions %>%
  arrange(predicted_frequency) %>%
  mutate(cumulative_exposures = cumsum(exposures)) %>%
  mutate(decile = floor(cumulative_exposures/total_exposures*10)+1) %>%
  mutate(decile = if_else(decile > 10, 10, decile)) %>%
  group_by(decile) %>%
  summarize(actual = sum(claim_count) / sum(exposures),
  expected = sum(predicted_claims) / sum(exposures),
```

```
  exposures = sum(exposures))

ggplot(aes(x = decile),data = quantile_plot_data) +
geom_line(aes(y = expected)) +
  geom_point(aes(y = actual),color="blue") +
  scale_x_continuous(limits = c(0,11), breaks = seq(1,10,1))
```




```
ggplot(aes(x = decile),data = quantile_plot_data) +
geom_col(aes(y = exposures)) +
  scale_x_continuous(limits = c(0,11), breaks = seq(1,10,1))
```



Residual Plot by Predicted Value

```
min(testing_predictions$predicted_frequency)
```

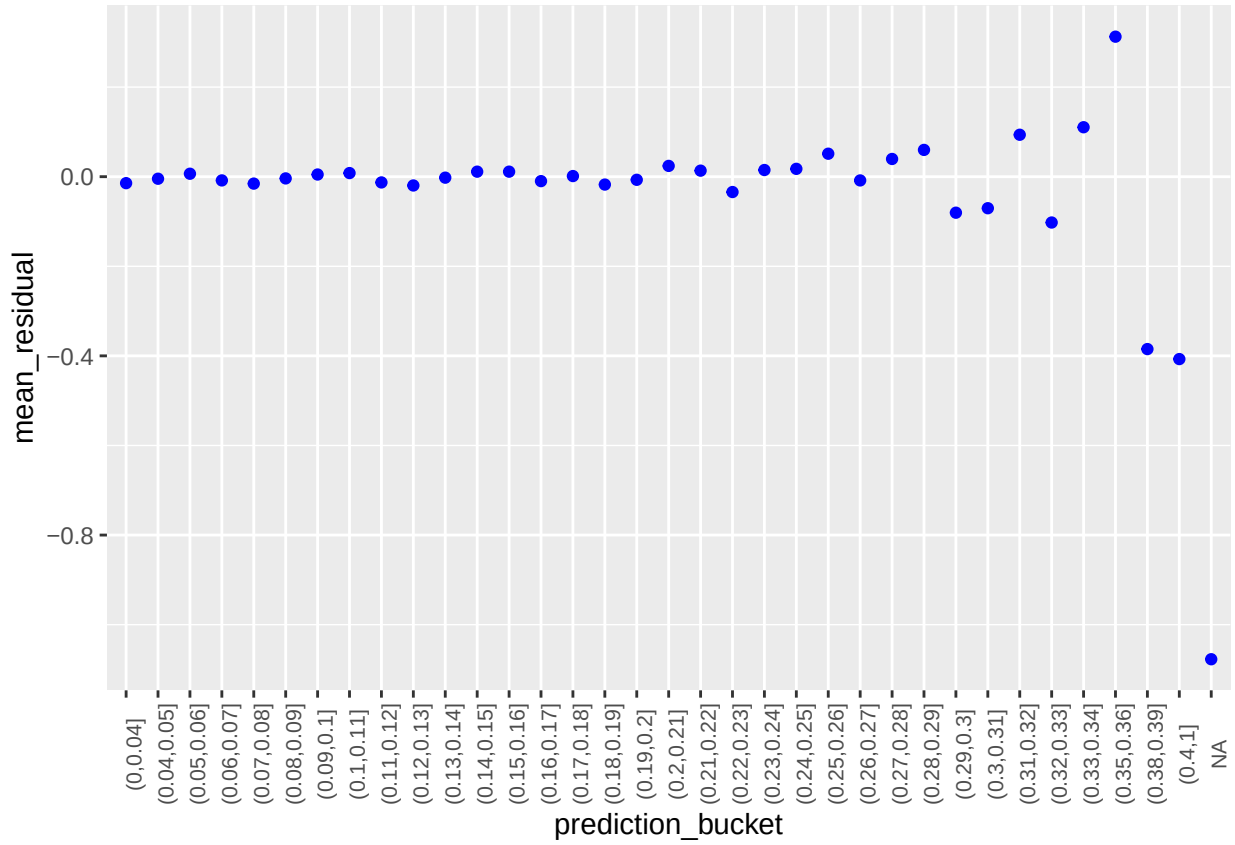
```
## [1] 0.03053216
```

```
max(testing_predictions$predicted_frequency)
```

```
## [1] 1.07716
```

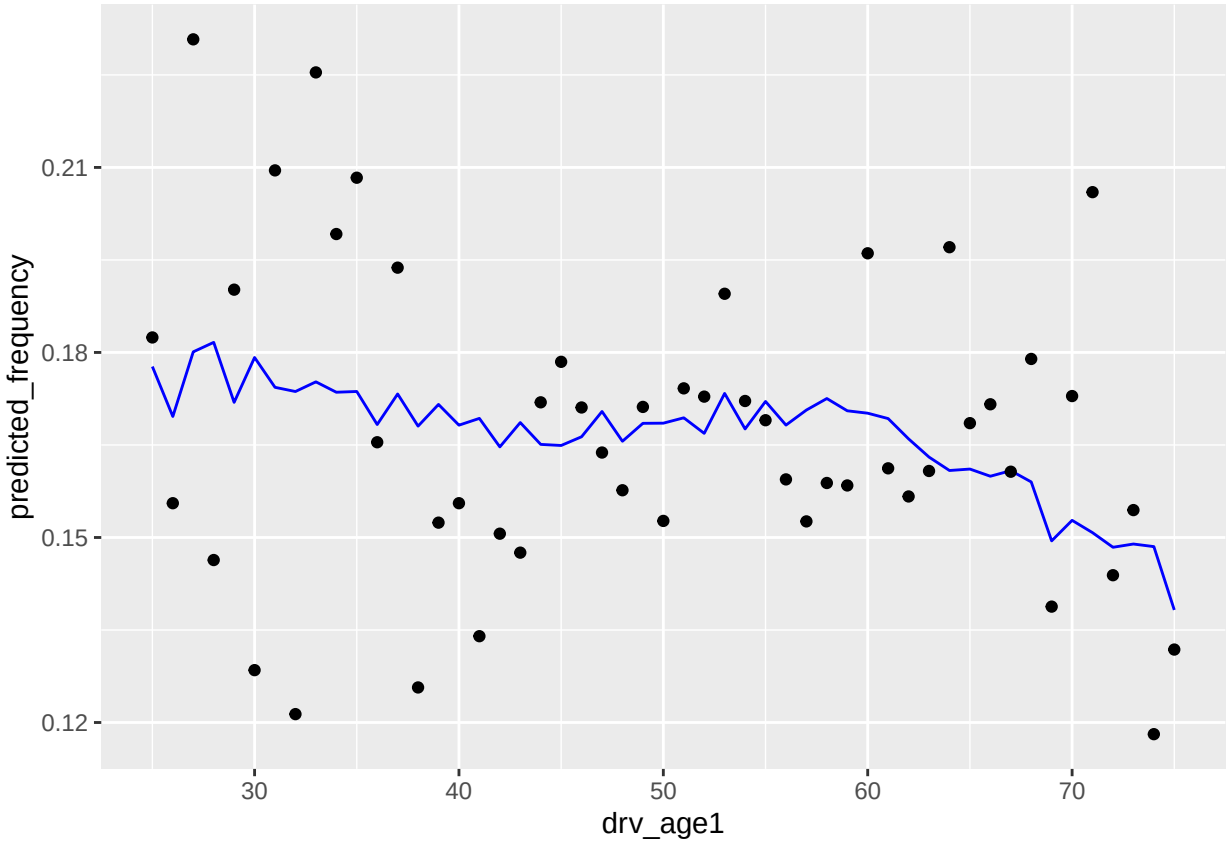
```
testing_predictions <- testing_predictions %>%
  mutate(prediction_bucket = cut(predicted_frequency,
                                 breaks = c(0,0.04,0.05,
                                            0.06,0.07,0.08,0.09,0.10,
                                            0.11,0.12,0.13,0.14,0.15,
                                            0.16,0.17,0.18,0.19,0.20,
                                            0.21,0.22,0.23,0.24,0.25,
                                            0.26,0.27,0.28,0.29,0.30,
                                            0.31,.32,0.33,0.34,0.35,
                                            0.36,0.37,0.38,0.39,0.40,
                                            1.00)))
```

```
residual_plot_data <- testing_predictions %>%
  mutate(residual = claim_count - predicted_claims) %>%
  group_by(prediction_bucket) %>%
  summarize(mean_residual = mean(residual))

ggplot(aes(x = prediction_bucket),data = residual_plot_data) +
  geom_point(aes(y=mean_residual),color = "blue")+
  theme(axis.text.x=element_text(angle=90, size=8))
```

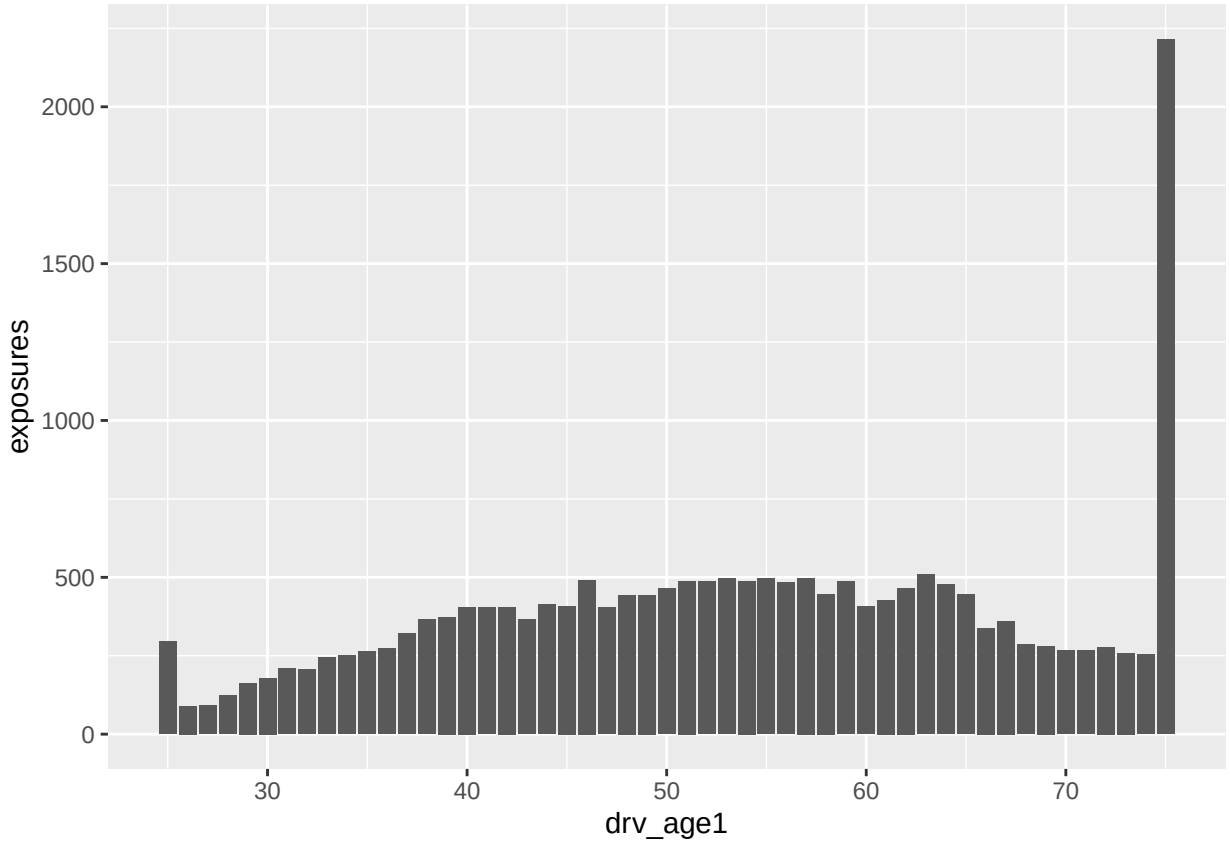


Actual vs. Expected

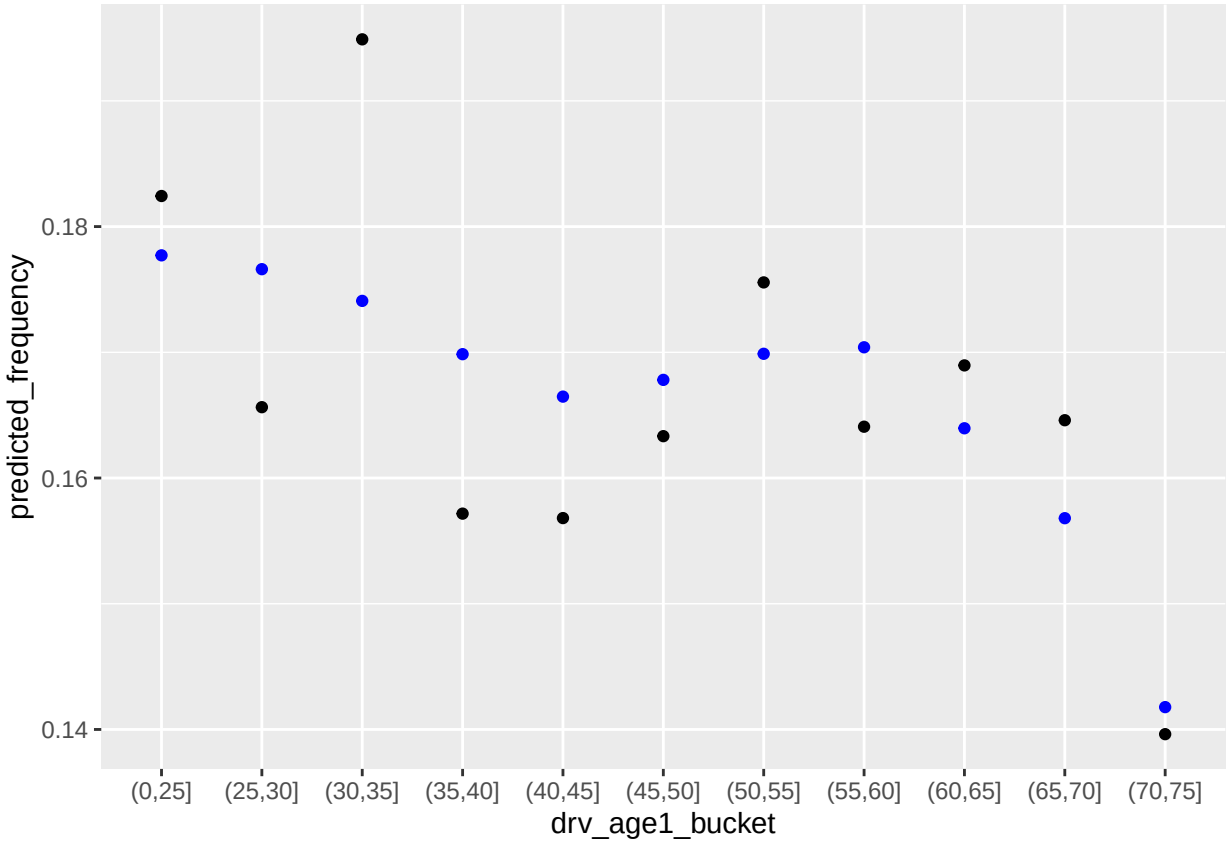
```
driver_age_a_v_e <- testing_predictions %>%
  group_by(drv_age1) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = drv_age1),data = driver_age_a_v_e) +
geom_line(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



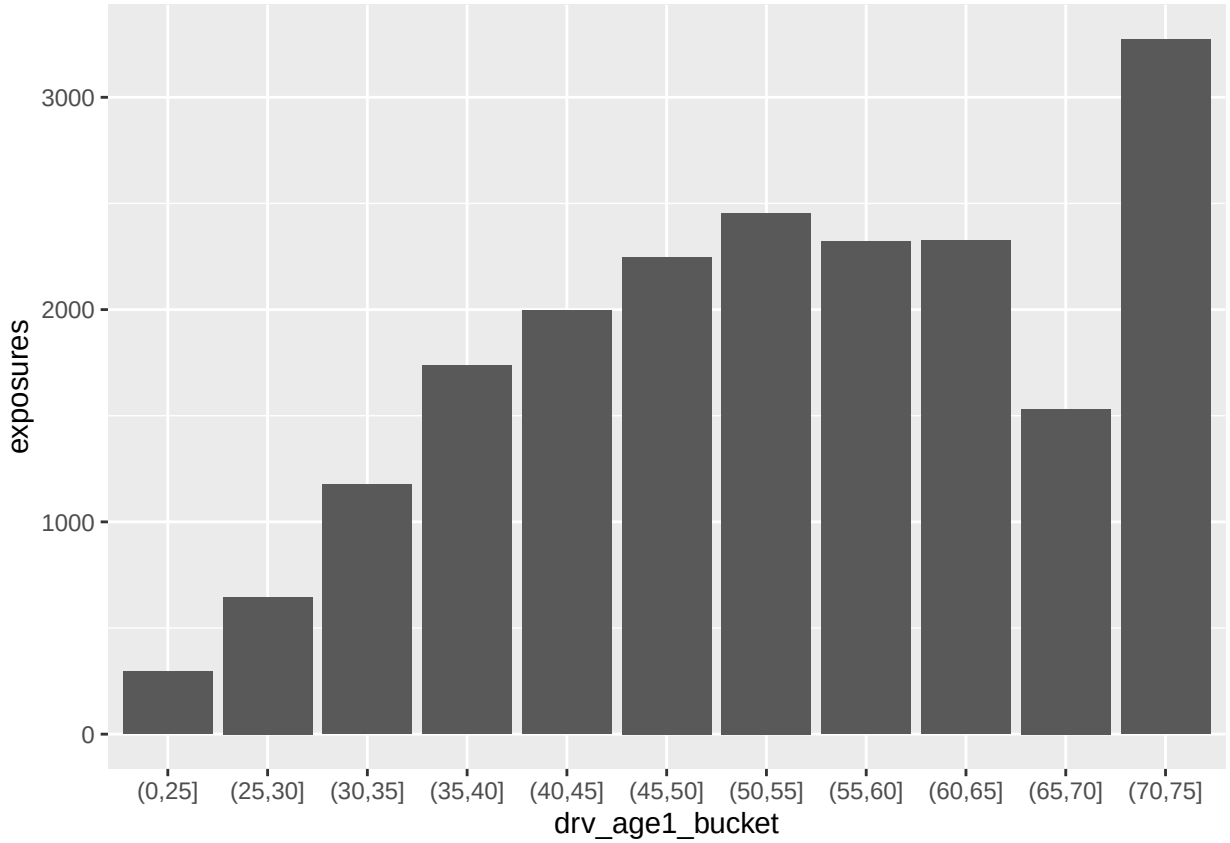

```
ggplot(aes(x = drv_age1),data = driver_age_a_v_e) +
geom_col(aes(y = exposures))
```



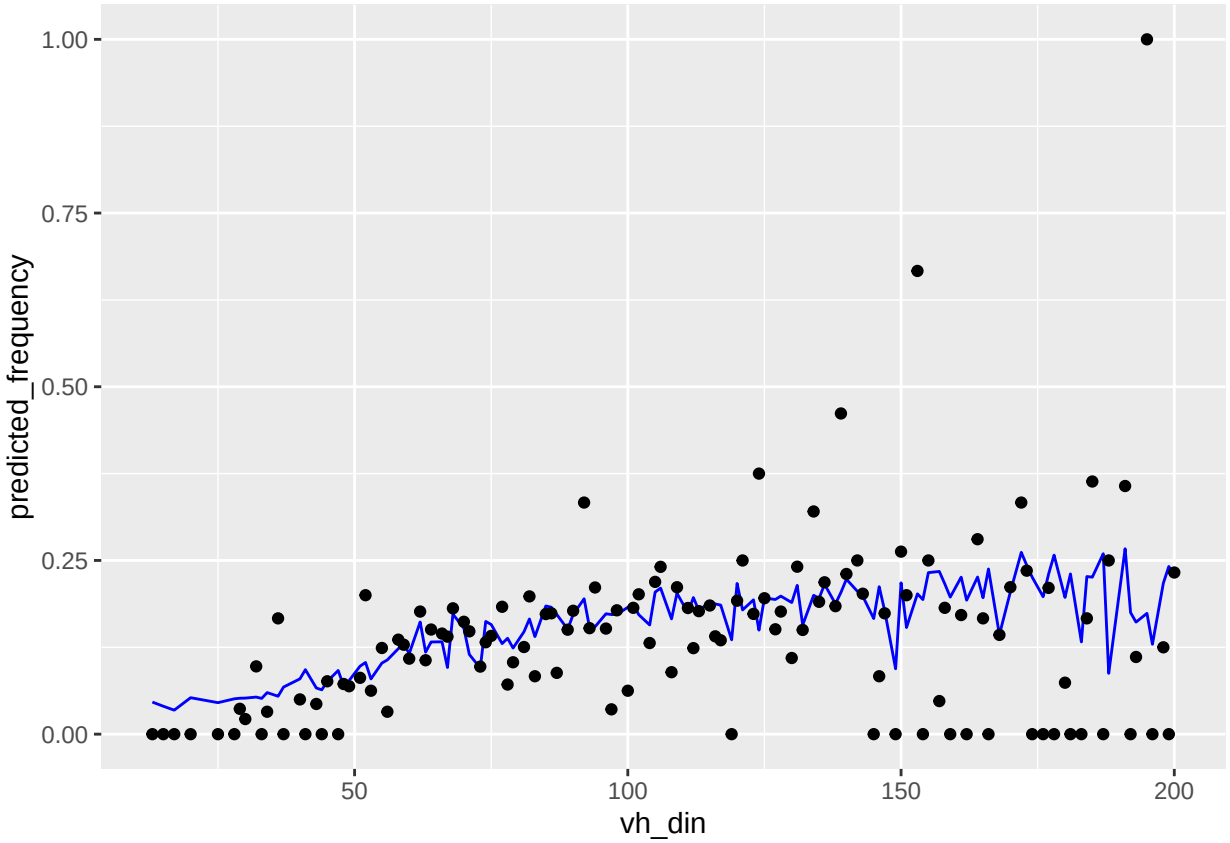

```
driver_age_a_v_e2 <- testing_predictions %>%
  mutate(drv_age1_bucket = cut(drv_age1,
                               breaks = c(0,25,30,35,40,45,50,
                                          55,60,65,70,75,200))) %>%
  group_by(drv_age1_bucket) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = drv_age1_bucket),data = driver_age_a_v_e2) +
geom_point(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



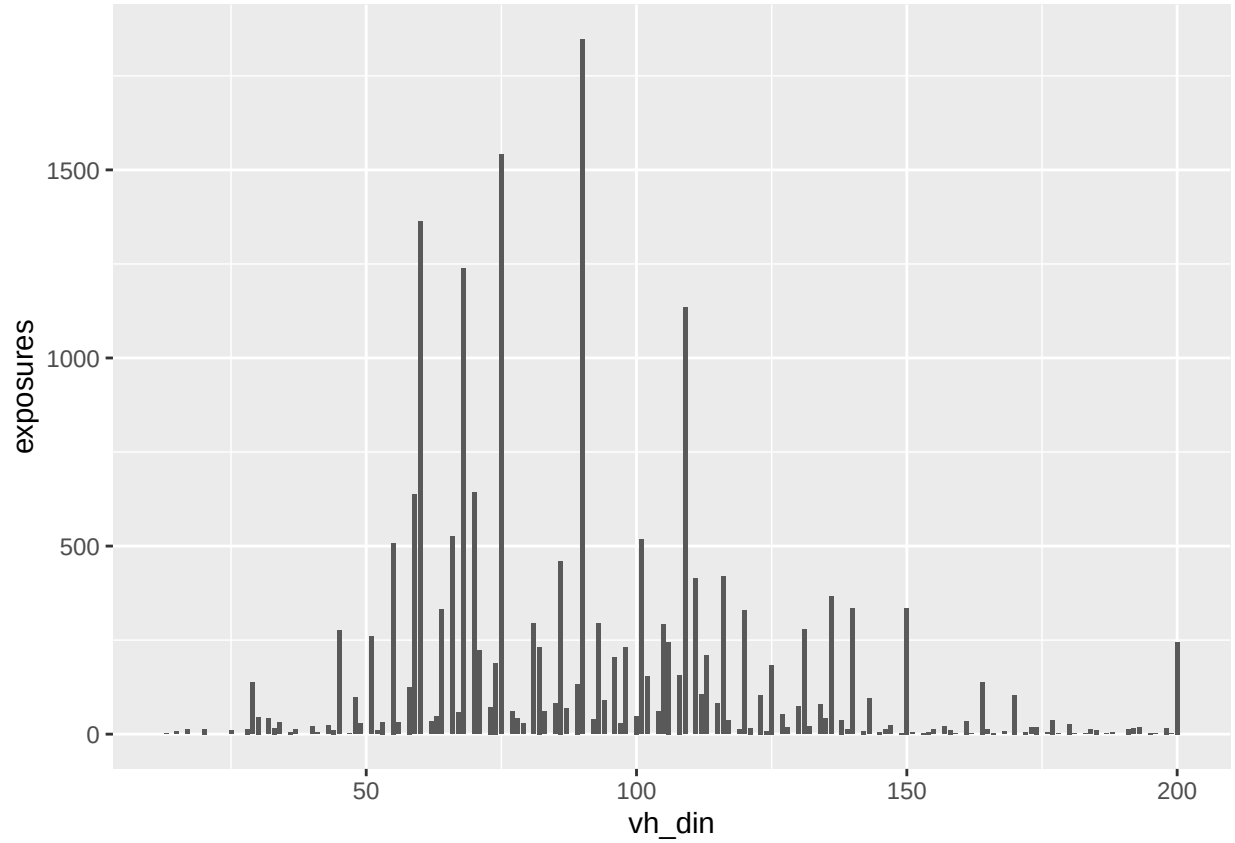

```
ggplot(aes(x = drv_age1_bucket),data = driver_age_a_v_e2) +
geom_col(aes(y = exposures))
```



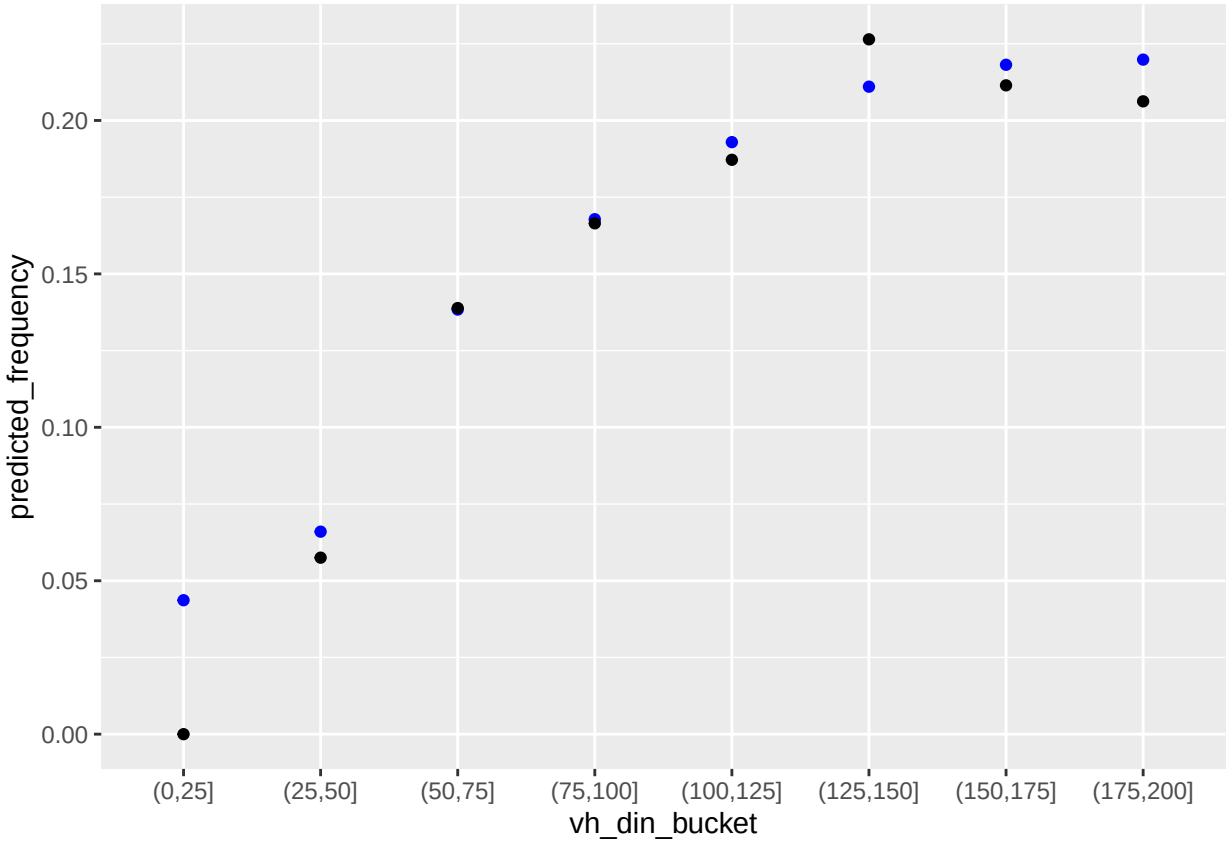

```
# Vehicle DIN (Metric Horse Power)

vh_din_a_v_e <- testing_predictions %>%
  group_by(vh_din) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = vh_din),data = vh_din_a_v_e) +
geom_line(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



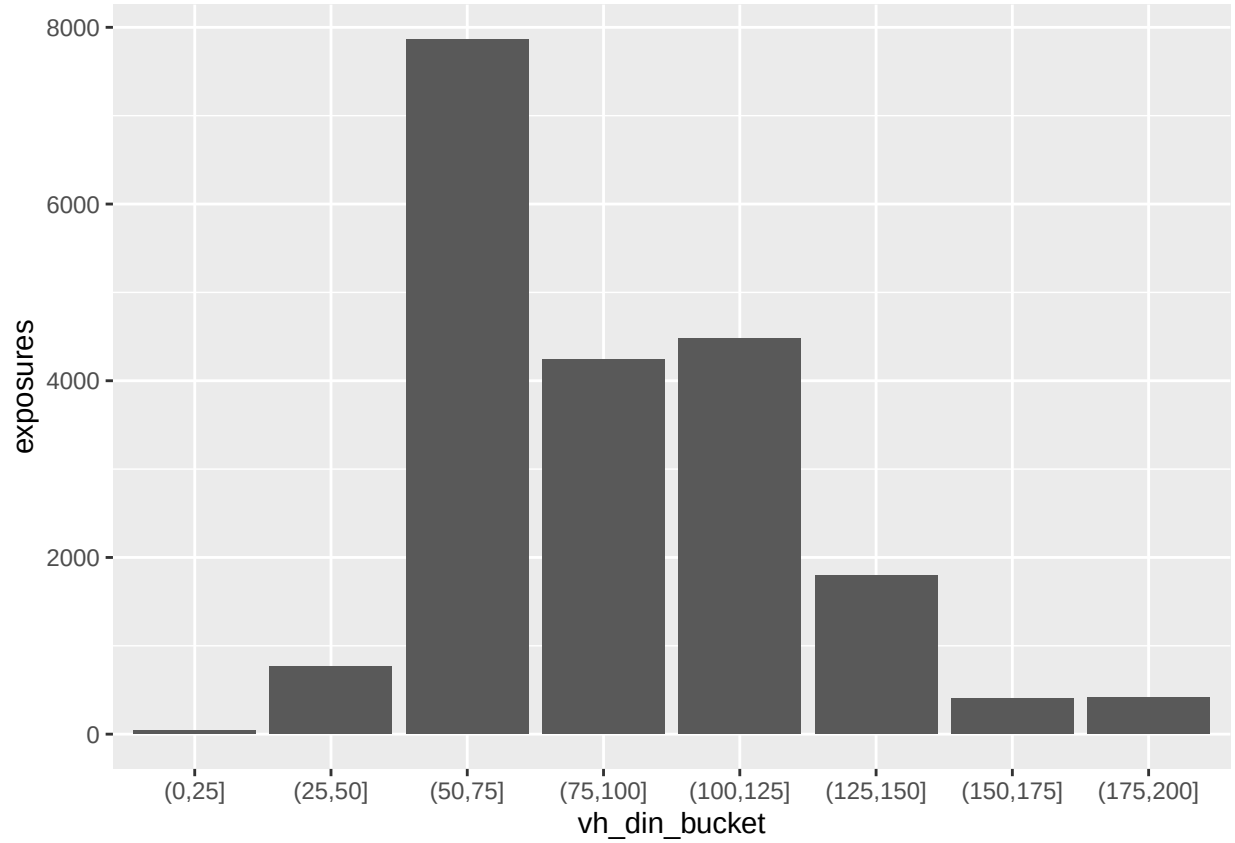

```
ggplot(aes(x = vh_din),data = vh_din_a_v_e) +
geom_col(aes(y = exposures))
```



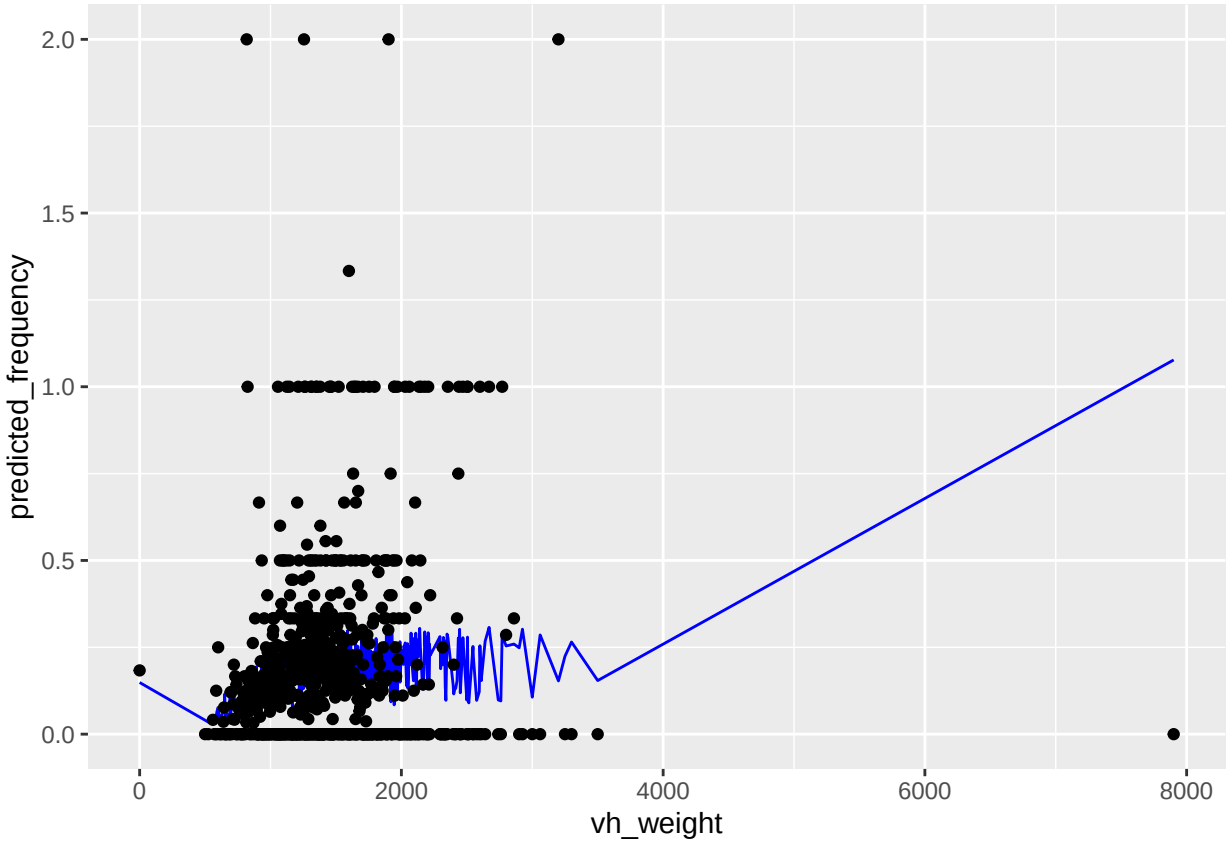

```
vh_din_a_v_e2 <- testing_predictions %>%
  mutate(vh_din_bucket = cut(vh_din,
                             breaks = c(0,25,50,75,100,
                                        125,150,175,200,1000))) %>%
  group_by(vh_din_bucket) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = vh_din_bucket),data = vh_din_a_v_e2) +
geom_point(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



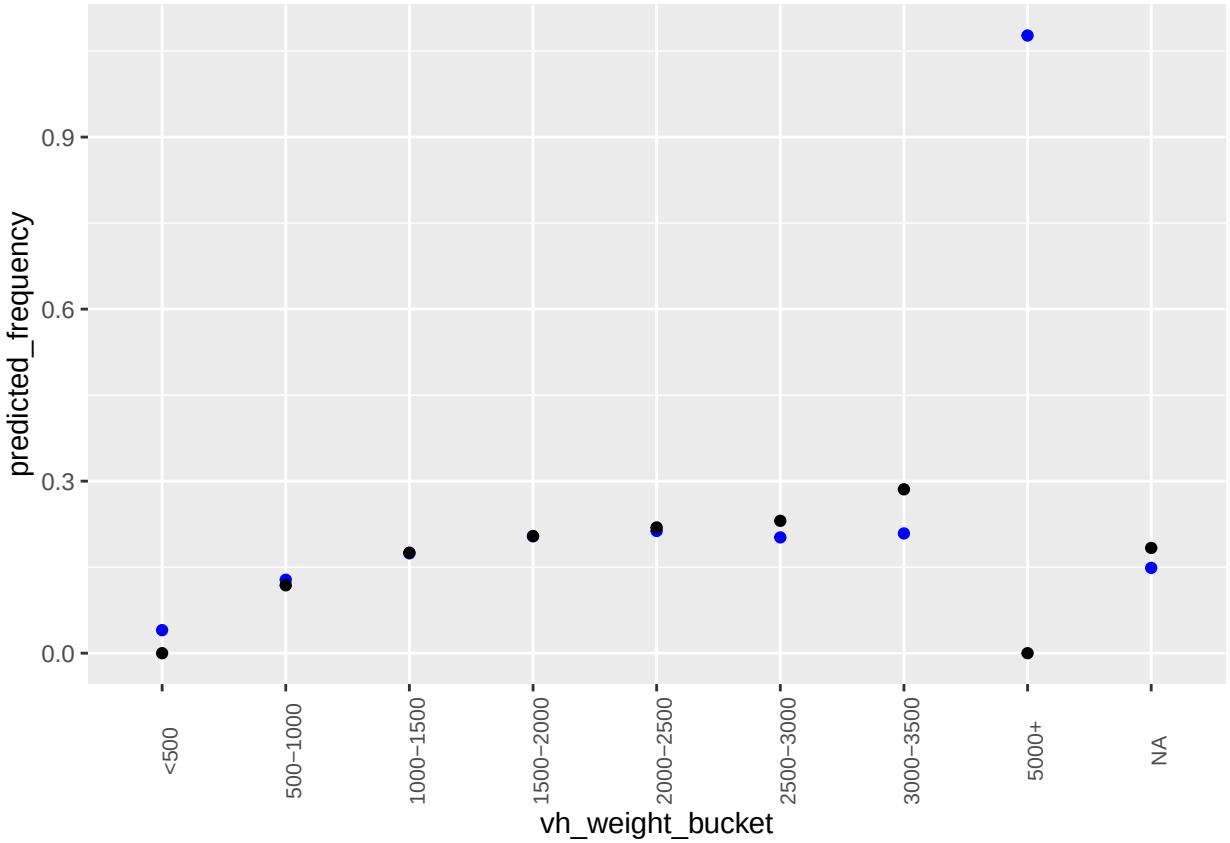

```
ggplot(aes(x = vh_din_bucket),data = vh_din_a_v_e2) +
geom_col(aes(y = exposures))
```



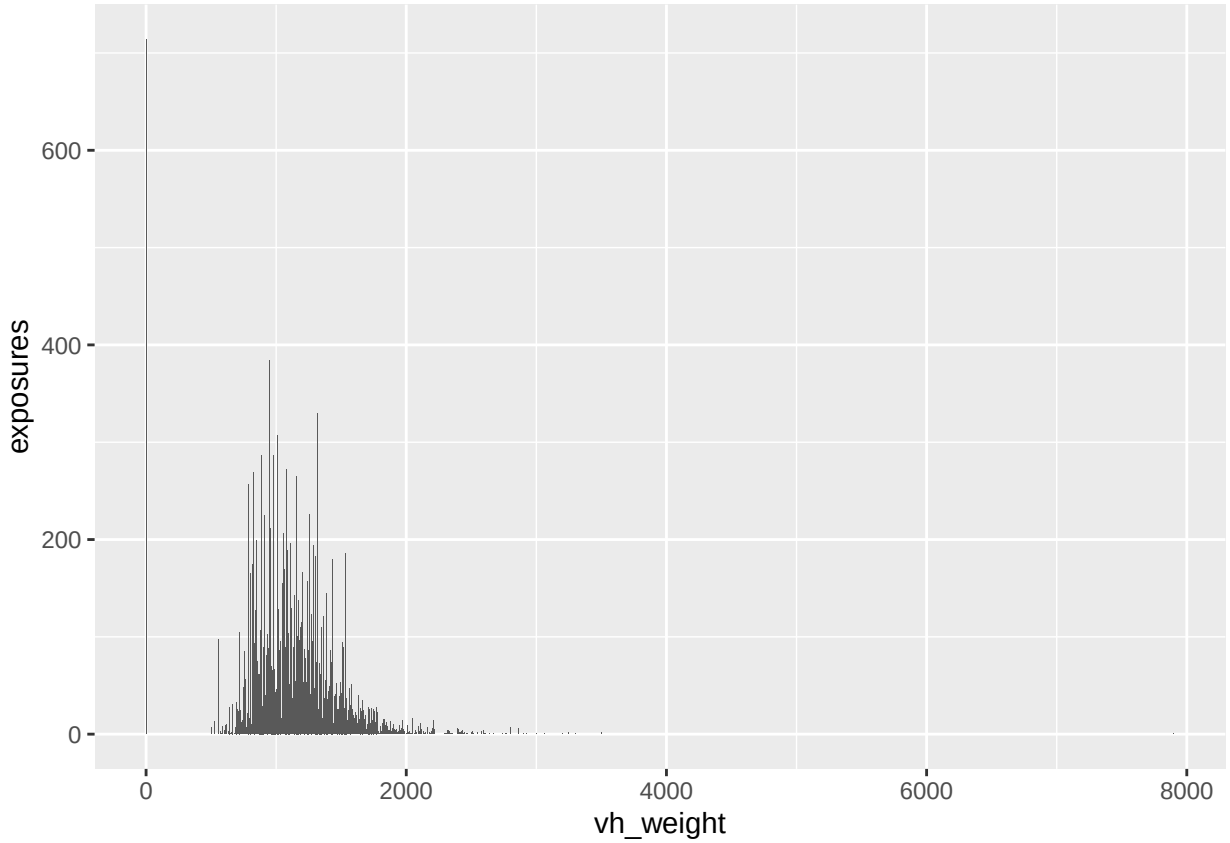

```
# Vehicle Weight

vh_weight_a_v_e <- testing_predictions %>%
  group_by(vh_weight) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = vh_weight),data = vh_weight_a_v_e) +
geom_line(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



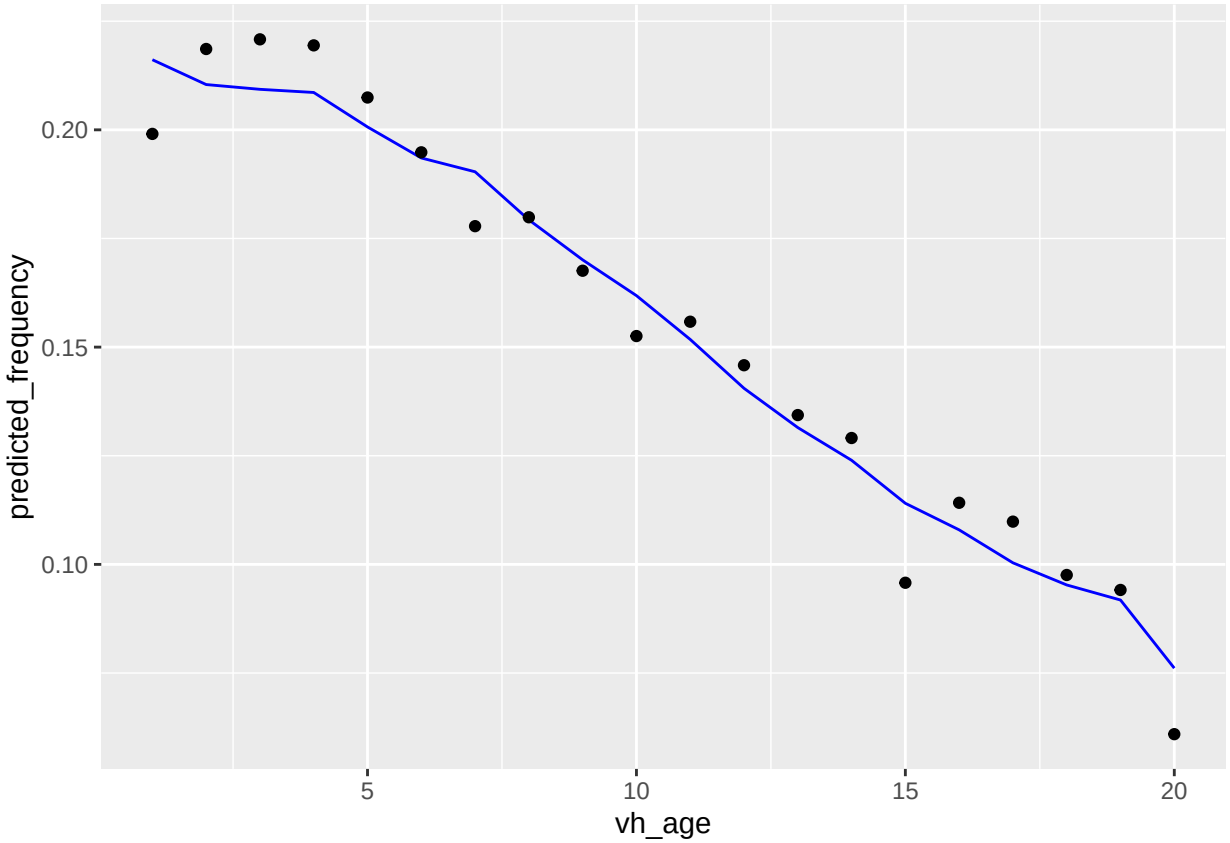

```
vh_weight_a_v_e2 <- testing_predictions %>%
  mutate(vh_weight_bucket = cut(vh_weight,
                                breaks = c(0,500,1000,1500,2000,2500,
                                           3000,3500,4000,4500,5000,
                                           10000),
                                labels = c("<500","500-1000","1000-1500","1500-2000","2000-2500",
                                           "2500-3000","3000-3500","3500-4000","4000-4500","4500-5000",
                                           "5000+"))) %>%
  group_by(vh_weight_bucket) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = vh_weight_bucket),data = vh_weight_a_v_e2) +
geom_point(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency)) +
  theme(axis.text.x=element_text(angle=90, size=8))
```



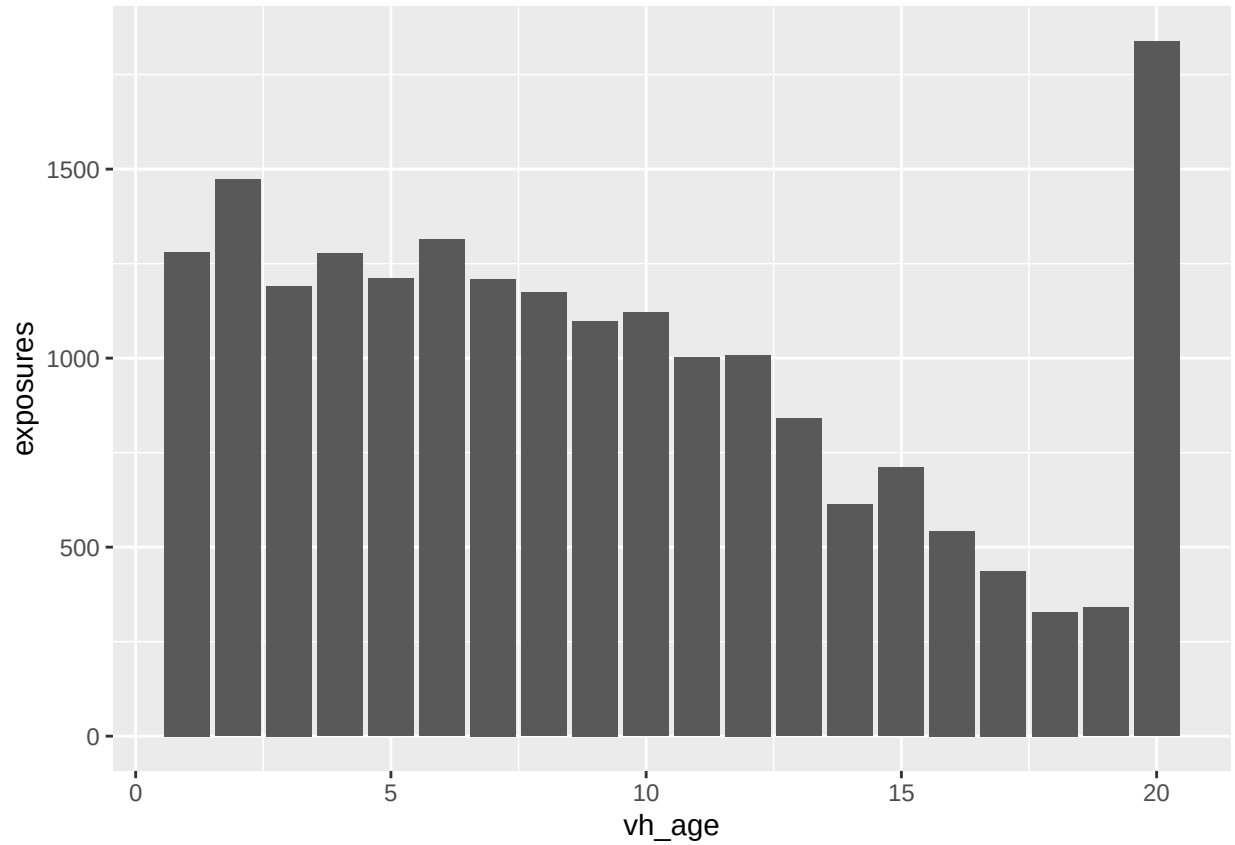

```
ggplot(aes(x = vh_weight),data = vh_weight_a_v_e) +
geom_col(aes(y = exposures))
```



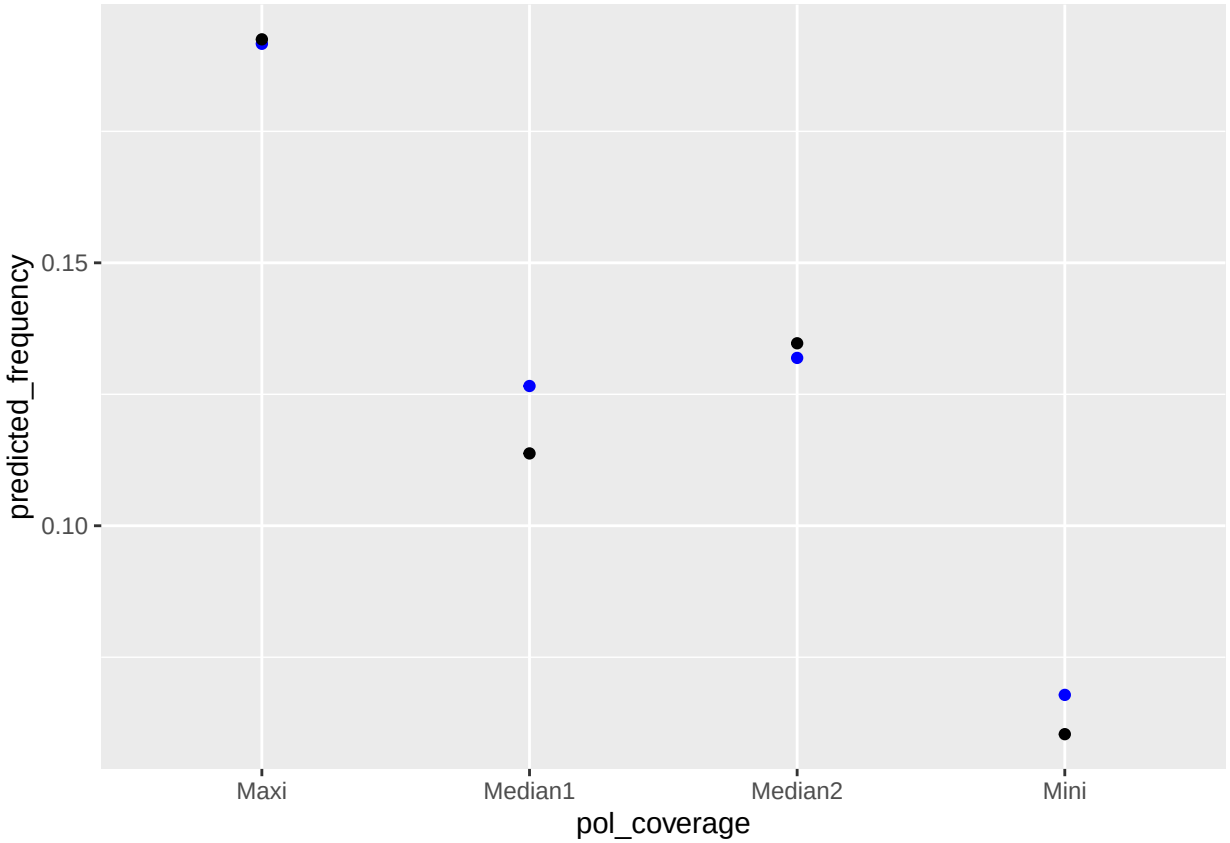

```
# Vehicle Age

vh_age_a_v_e <- testing_predictions %>%
  group_by(vh_age) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = vh_age),data = vh_age_a_v_e) +
geom_line(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



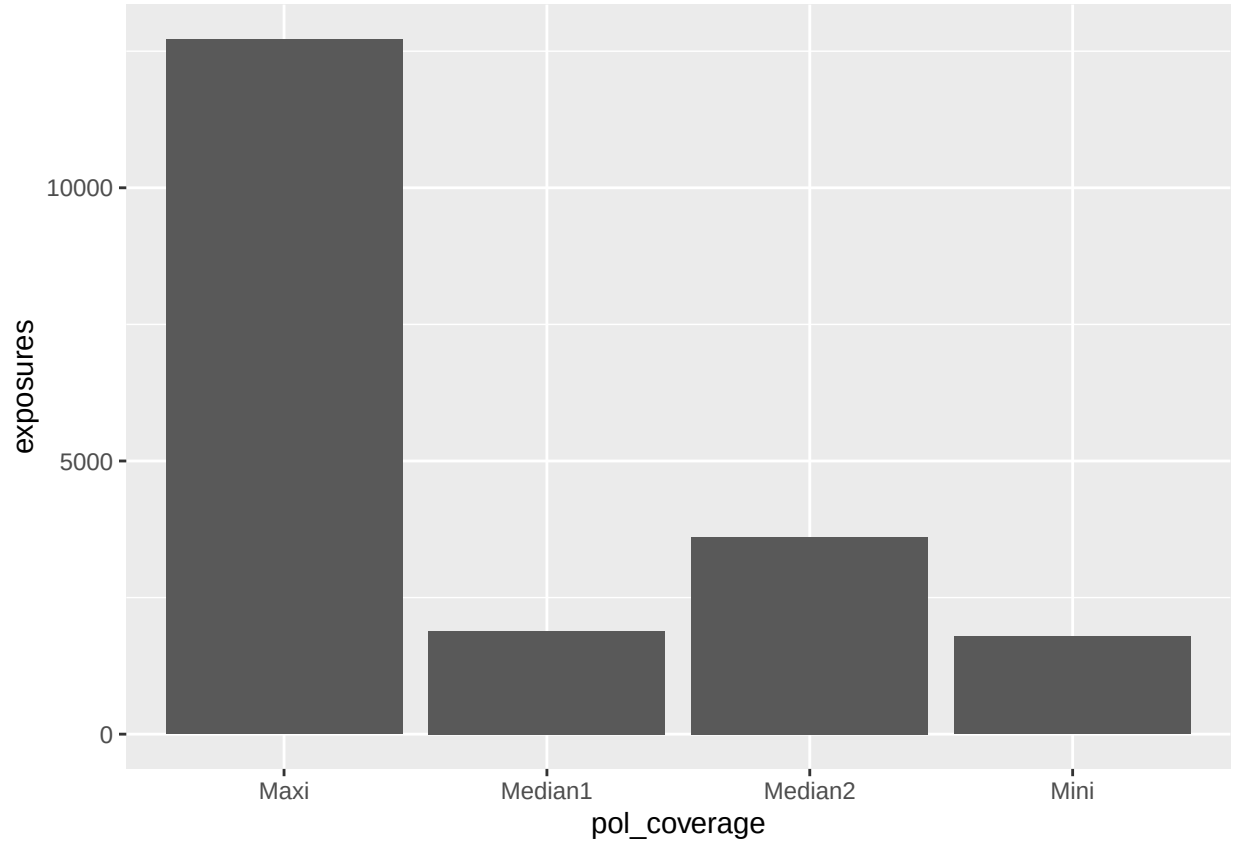

```
ggplot(aes(x = vh_age),data = vh_age_a_v_e) +
geom_col(aes(y = exposures))
```



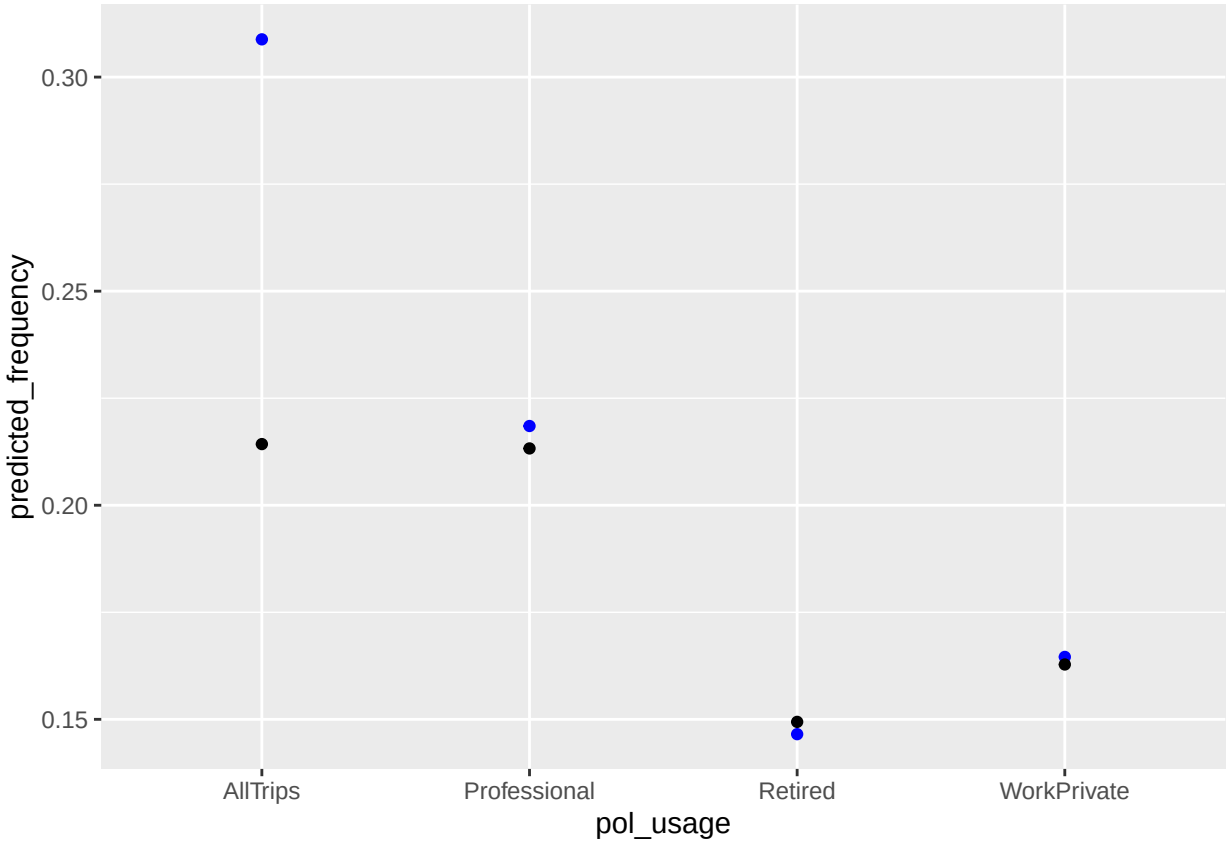

```
# Policy Coverage

pol_coverage_a_v_e <- testing_predictions %>%
  group_by(pol_coverage) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = pol_coverage),data = pol_coverage_a_v_e) +
geom_point(aes(y = predicted_frequency), color = "blue") +
  geom_point(aes(y = actual_frequency))
```



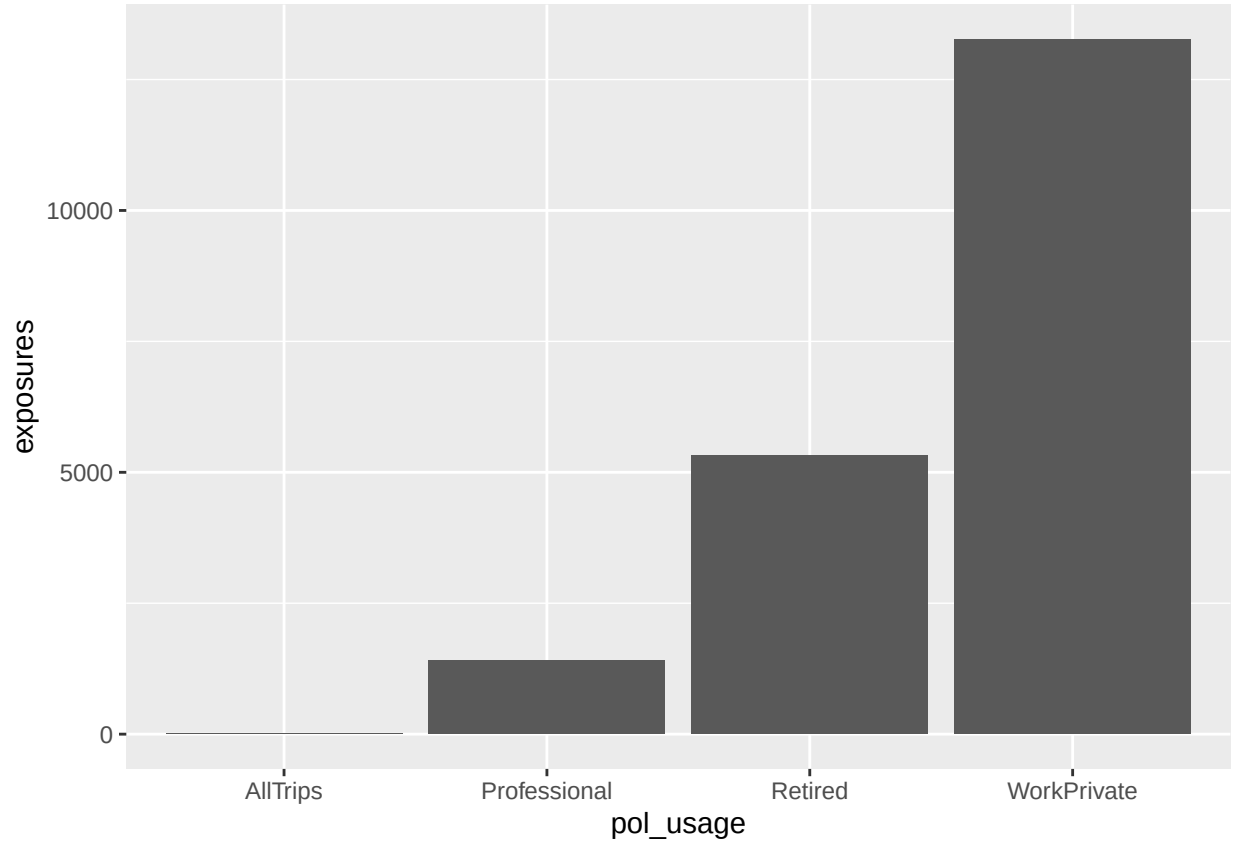

```
ggplot(aes(x = pol_coverage),data = pol_coverage_a_v_e) +
geom_col(aes(y = exposures))
```




```
# Policy Usage

pol_usage_a_v_e <- testing_predictions %>%
  group_by(pol_usage) %>%
  summarize(predicted_frequency = sum(predicted_claims)/sum(exposures),
            actual_frequency = sum(claim_count)/sum(exposures),
            exposures = sum(exposures))

ggplot(aes(x = pol_usage),data = pol_usage_a_v_e) +
geom_point(aes(y = predicted_frequency),color = "blue") +
  geom_point(aes(y = actual_frequency))
```




```
ggplot(aes(x = pol_usage),data = pol_usage_a_v_e) +
geom_col(aes(y = exposures))
```



Incremental AIC

```
gam_step1 <- gam(claim_count ~ pol_coverage,
                 family = poisson(link = "log"),
                 offset = log(exposures),
                 data = training_data)

aic_1 <- AIC(gam_step1)

gam_step2 <- gam(claim_count ~ pol_coverage + pol_usage,
                 family = poisson(link = "log"),
                 offset = log(exposures),
                 data = training_data)

aic_2 <- AIC(gam_step2)

gam_step3 <- gam(claim_count ~ pol_coverage + pol_usage +
                   s(drv_age1, k = 4),
                 family = poisson(link = "log"),
                 offset = log(exposures),
                 data = training_data)

aic_3 <- AIC(gam_step3)
```

```r
gam_step4 <- gam(claim_count ~ pol_coverage + pol_usage +
                   s(drv_age1, k = 4) + s(vh_age, k = 4),
                 family = poisson(link = "log"),
                 offset = log(exposures),
                 data = training_data)

aic_4 <- AIC(gam_step4)

gam_step5 <- gam(claim_count ~ pol_coverage + pol_usage +
                   s(drv_age1, k = 4) + s(vh_age, k = 4) +
                   te(vh_din, vh_weight, k = 3),
                 family = poisson(link = "log"),
                 offset = log(exposures),
                 data = training_data)

aic_5 <- AIC(gam_step5)

aic_1
```

```
## [1] 75572.66
```

```r
aic_2
```

```
## [1] 75421.85
```

```r
aic_3
```

```
## [1] 75406.44
```

```r
aic_4
```

```
## [1] 75117.55
```

```r
aic_5
```

```
## [1] 74954.88
```