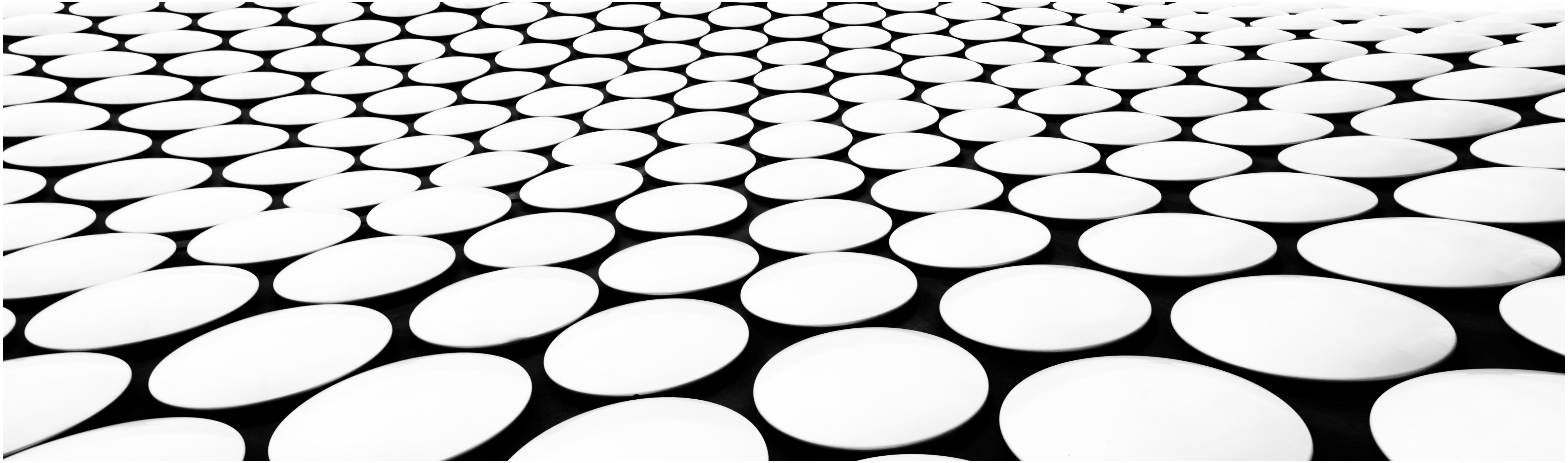
SAMPLE GAM SMOOTHS WITH THE MGCV R PACKAGE

SAM KLOESE, ACAS, CSPA, MAAA, CPCU, DTM



AGENDA

- Introduce dataOhlsson dataset
- Example Smooths
  - Thin plate smooths on driver age
  - Tensor interaction smooth on engine vehicle ratio and driver age
- Evaluating GAM exercise
  - Build a GAM on Training Data
  - Analyze Model Output
  - Show Holdout Test
- Review of Questions for GAMs

DATAOHLSSON DATASET

- Part of R's insuranceData package
  - Motorcycle Insurance Data
  - Former Swedish Insurance Company Wasa
  - Data from 1994-1998
- Preliminary adjustments
  - Columns renamed to English
  - Driver age capped between 16 and 80
  - Vehicle age capped at 40
  - Filtered for records where exposures > .01

- Data Size
  - 64,548 rows
  - 9 columns
  - 692 claims (fairly small)
  - 65,235.2 exposures
- Renamed Columns:
  - Driver Age
  - Gender
  - Parish
  - Engine Vehicle Ratio
    - (Engine power in kW x 100) / (Vehicle weight in kg + 75)
    - Rounded to the nearest lower integer
    - 75 kg represent the average driver weight
  - Vehicle Age
  - Claim-Free Bonus Class
  - Exposures
  - Claims
  - Losses

THIN PLATE SMOOTHS

THIN PLATE SMOOTHS

- Columns to smooth are wrapped in “s(   )”
- The smooth type is controlled by the bs argument
- The default smooth type is thin plate (bs = “tp”)
- The argument k determines the number of basis functions

s {mgcv} R Documentation

Defining smooths in GAM formulae

Description

Function used in definition of smooth terms within `gam` model formulae. The function does not evaluate a (spline) smooth - it exists purely to help set up a model using spline based smooths.

Usage

```
s(..., k=-1,fx=FALSE,bs="tp",m=NA,by=NA,xt=NULL,id=NULL,sp=NULL,pc=NULL)
```

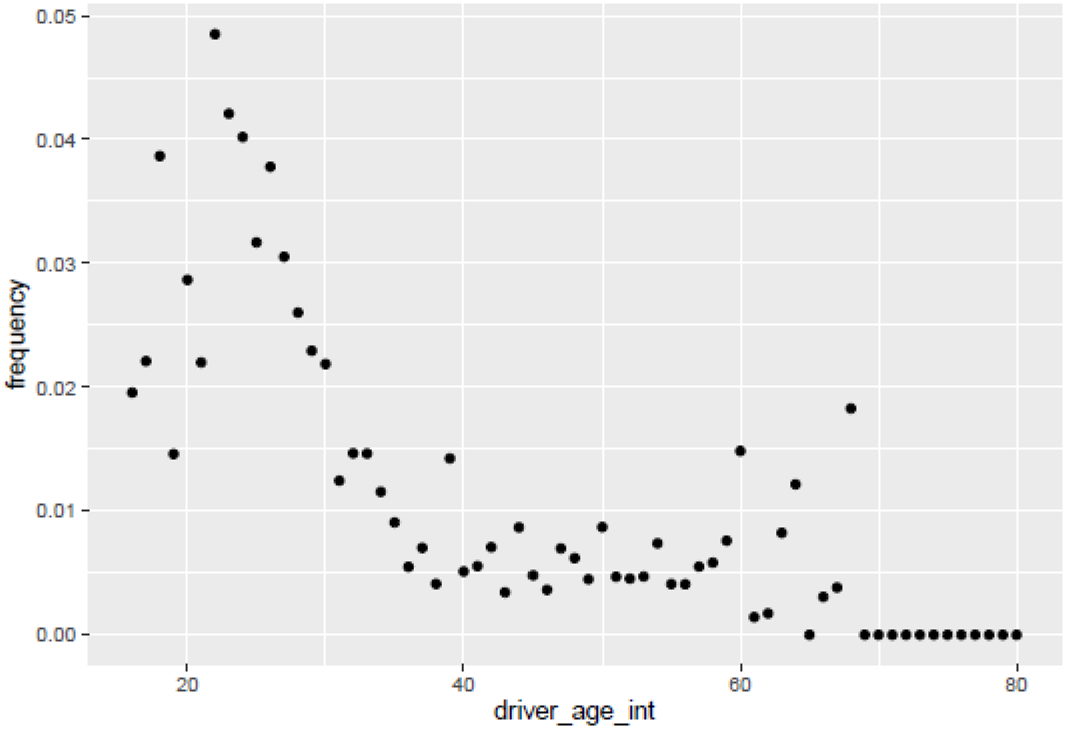
Arguments

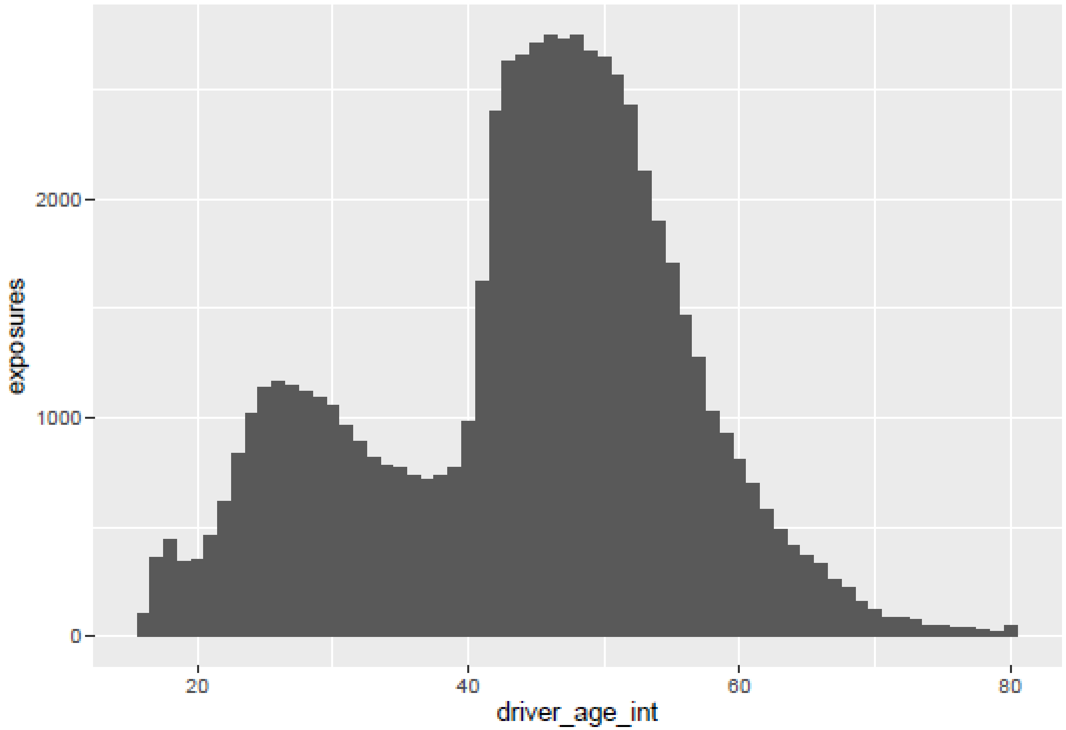
`...`	a list of variables that are the covariates that this smooth is a function of. Transformations whose form depends on the values of the data are best avoided here: e.g. `s(log(x))` is fine, but `s(I(x/sd(x)))` is not (see `predict.gam`).
`k`	the dimension of the basis used to represent the smooth term. The default depends on the number of variables that the smooth is a function of. `k` should not be less than the dimension of the null space of the penalty for the term (see `null.space.dimension`), but will be reset if it is. See `choose.k` for further information.
`fx`	indicates whether the term is a fixed d.f. regression spline (`TRUE`) or a penalized regression spline (`FALSE`).
`bs`	a two letter character string indicating the (penalized) smoothing basis to use. (eg `"tp"` for thin plate regression spline, `"cr"` for cubic regression spline). see `smooth.terms` for an over view of what is available.

THIN PLATE SMOOTHS

- Thin plate smooths can be used to smooth “wiggly” data
- Frequency by Driver age is well-known not to follow a straight line
  - High for the youngest drivers
  - High for the oldest drivers
  - Lower in the middle
- Experiment: Build a claim count model which uses thin plate smooths
  - Check how the smooth changes with different number of basis functions
  - Show default plot of the smooth term
  - Manually plot the predicted frequency by age

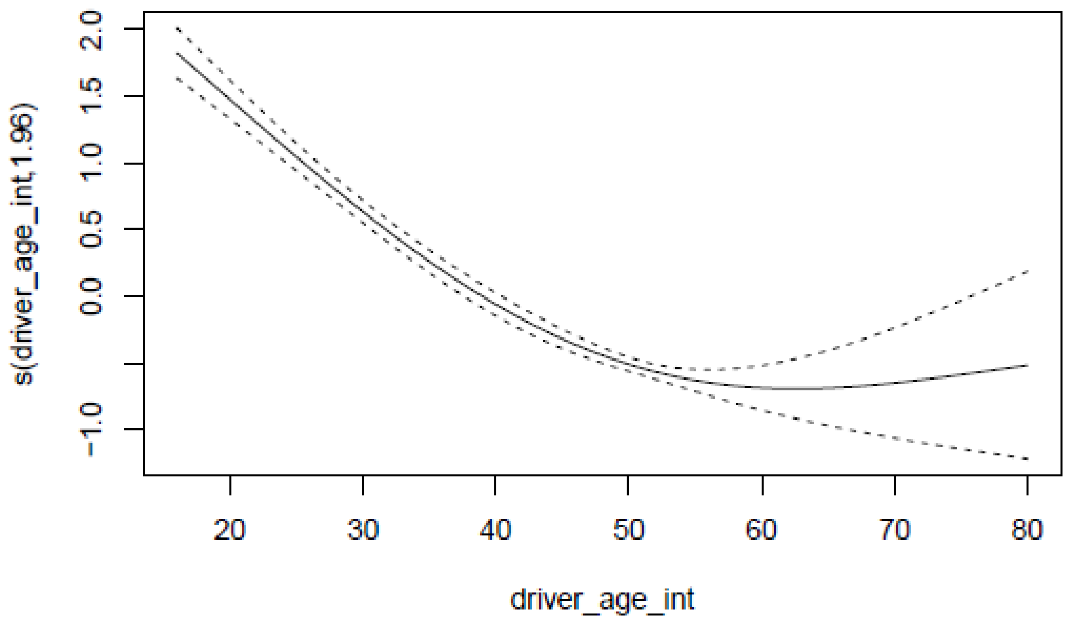
THIN PLATE SMOOTH





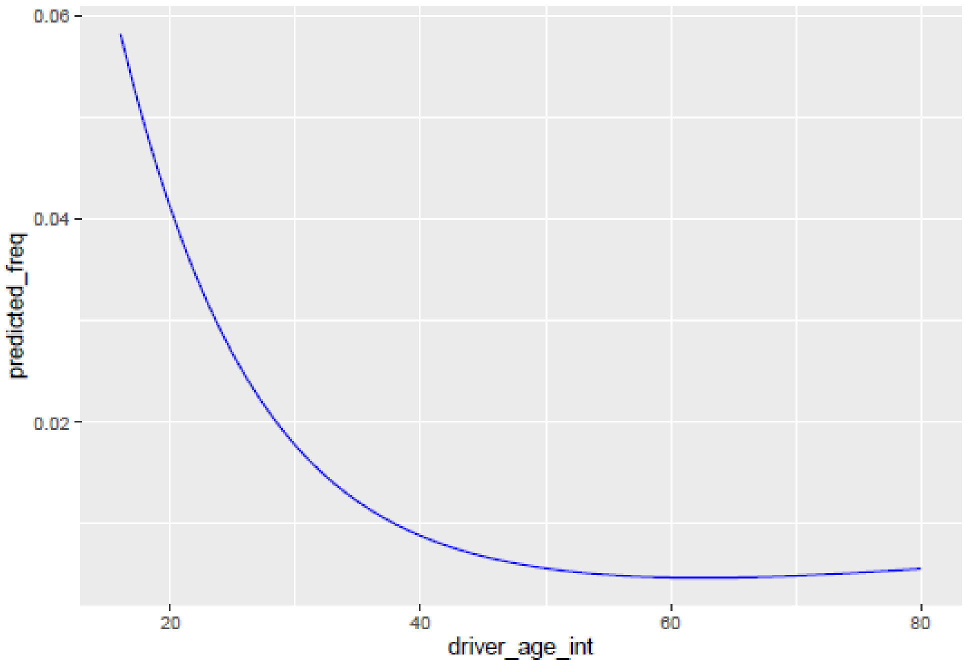
THIN PLATE SMOOTH (K = 3)

```r
gam_k3 <- gam(claims ~ s(driver_age_int, k = 3),
              family = poisson(link = "log"),
              offset = log(exposures),
              data = dataOhlsson)
```



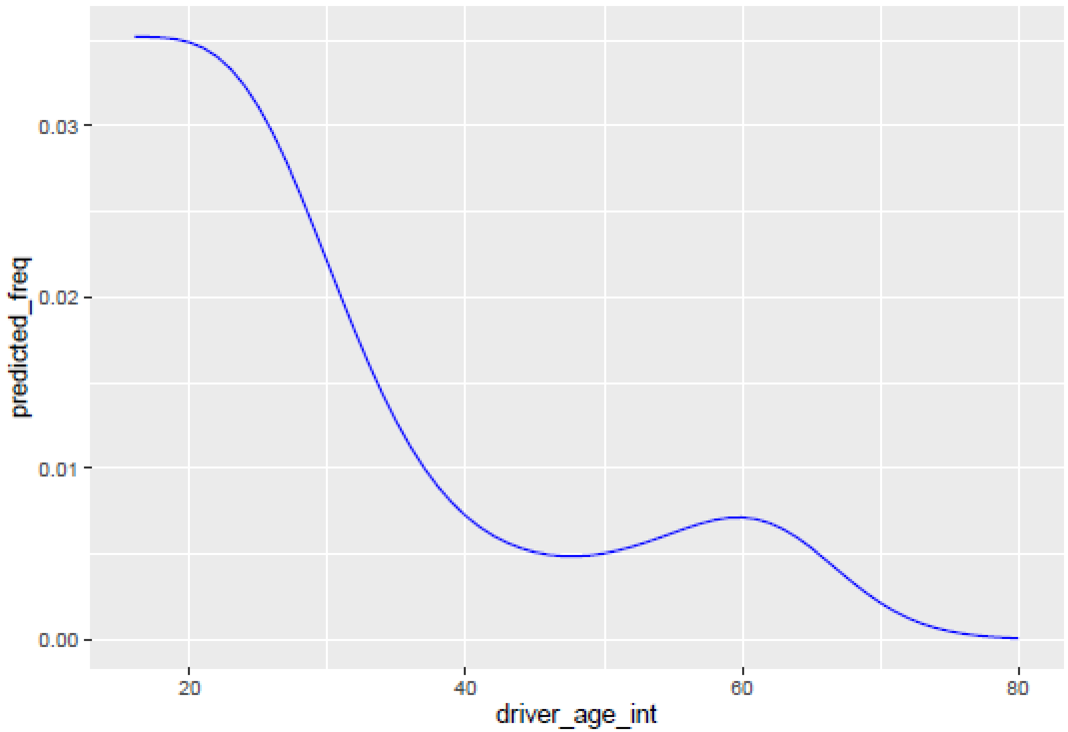

```r
coef(gam_k3)
```

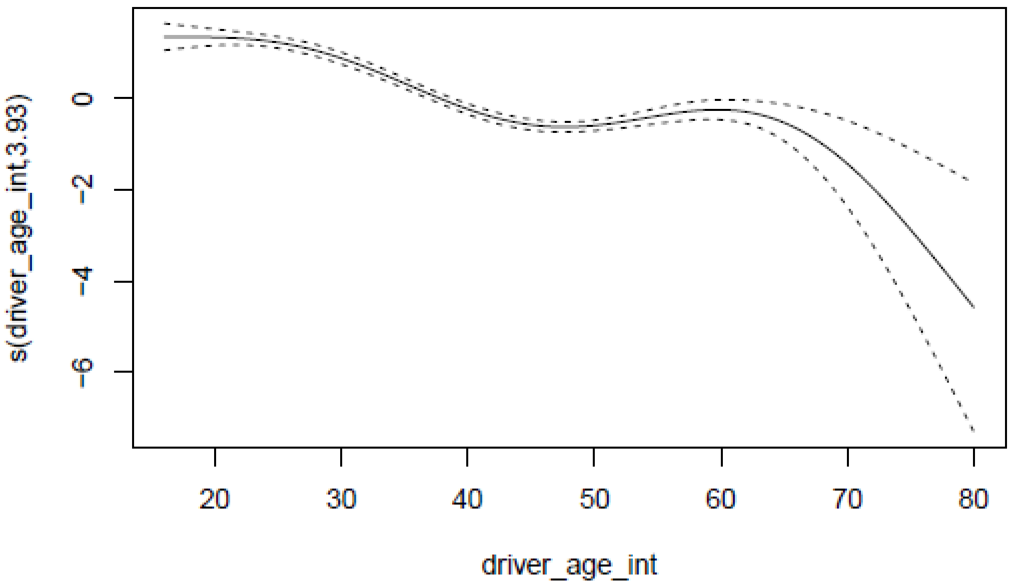
```
##        (Intercept) s(driver_age_int).1 s(driver_age_int).2
##         -4.6663586          -1.1324474          -0.4734132
```



THIN PLATE SMOOTH (K = 5)

```r
gam_k5 <- gam(claims ~ s(driver_age_int, k = 5),
              family = poisson(link = "log"),
              offset = log(exposures),
              data = dataOhlsson)
plot(gam_k5, residuals = FALSE, pch = 1)
```



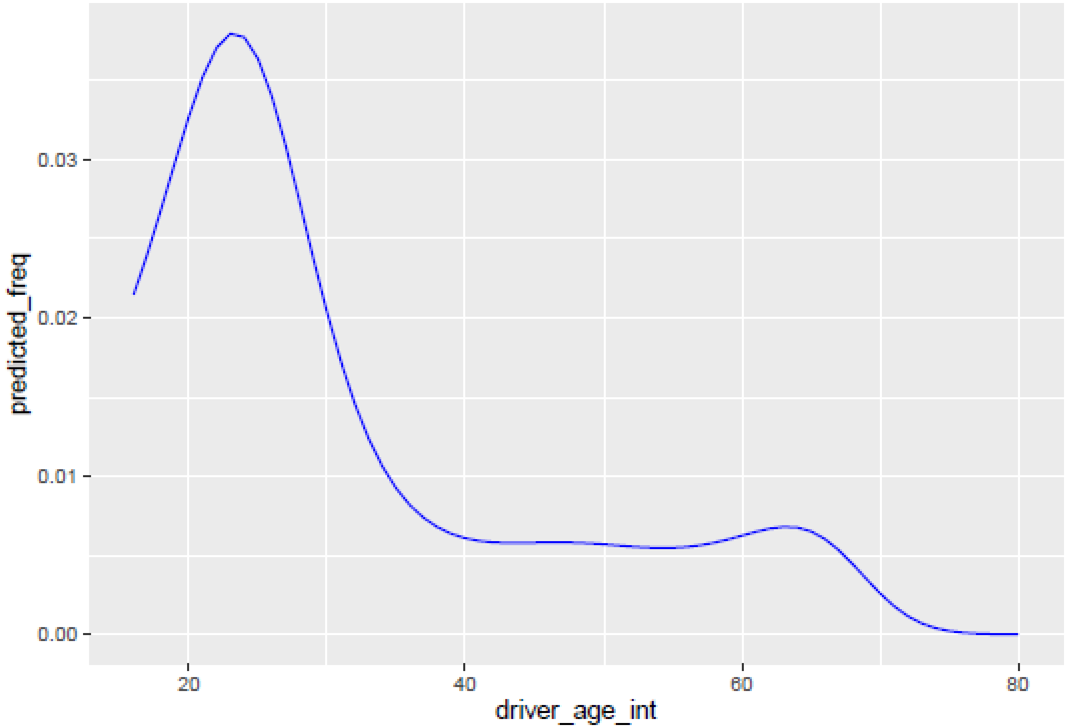


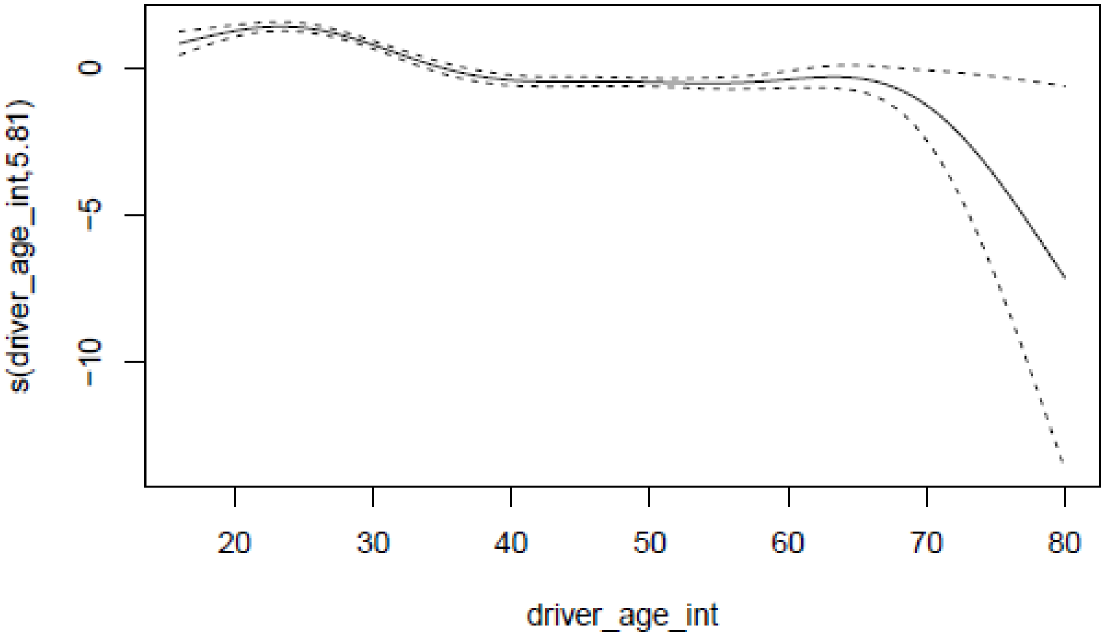

```r
coef(gam_k5)
```

```
##         (Intercept) s(driver_age_int).1 s(driver_age_int).2 s(driver_age_int).3
##           -4.690979            1.130637            3.966976            7.532659
## s(driver_age_int).4
##           -2.244242
```

THIN PLATE SMOOTH (K = 7)

```
gam_k7 <- gam(claims ~ s(driver_age_int, k = 7),
              family = poisson(link = "log"),
              offset = log(exposures),
              data = dataOhlsson)
plot(gam_k7, residuals = FALSE, pch = 1)
```



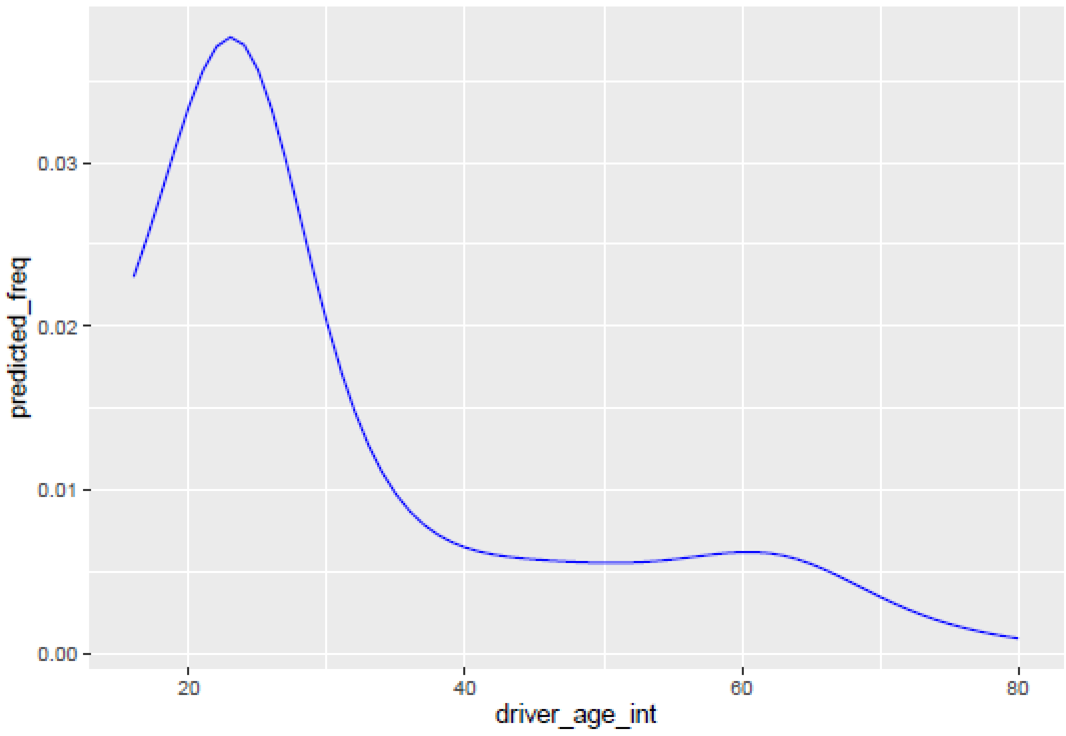


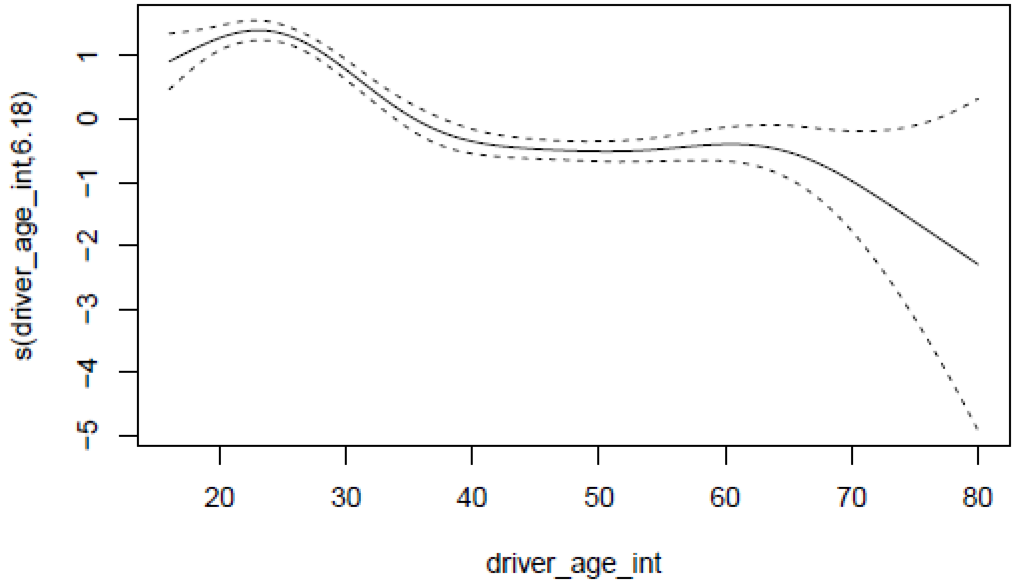

```
coef(gam_k7)
```

```
##         (Intercept) s(driver_age_int).1 s(driver_age_int).2 s(driver_age_int).3
##           -4.709409            1.709992            8.607368           -0.347413
## s(driver_age_int).4 s(driver_age_int).5 s(driver_age_int).6
##            5.910743           17.997471           -3.871071
```

THIN PLATE SMOOTH (K = 10)

```r
gam_k10 <- gam(claims ~ s(driver_age_int, k = 10),
               family = poisson(link = "log"),
               offset = log(exposures),
               data = dataOhlsson)
plot(gam_k10, residuals = FALSE, pch = 1)
```






```r
coef(gam_k10)
```

```
##         (Intercept) s(driver_age_int).1 s(driver_age_int).2 s(driver_age_int).3
##         -4.68524792          0.02755683          2.95300229          0.60707620
## s(driver_age_int).4 s(driver_age_int).5 s(driver_age_int).6 s(driver_age_int).7
##         -1.76307572         -0.40390420          1.26887501         -0.10365368
## s(driver_age_int).8 s(driver_age_int).9
##          4.97364308         -0.20512447
```

TENSOR INTERACTION SMOOTHS

TENSOR INTERACTION SMOOTHS

- Tensor interaction models pure interaction effect on top of the main effect
- Tensor smooths are useful when the 2 variables have different scales
- The relationship between driver age and engine vehicle ratio seems promising
  - Perhaps the added risk of a powerful motorcycle varies based on who is using it
  - Scales are different (Age: 16 – 80 years, EV Ratio: Classes 1 - 7
- Experiment: Build a claim count model which uses smoothed age, linear ev ratio, and a tensor interaction
  - Show how to formulate this model
  - Show visualization options

te {mgcv} R Documentation

Define tensor product smooths or tensor product interactions in GAM formulae

Description

Functions used for the definition of tensor product smooths and interactions within `gam` model formulae. `te` produces a full tensor product smooth, while `ti` produces a tensor product interaction, appropriate when the main effects (and any lower interactions) are also present.

The functions do not evaluate the smooth - they exists purely to help set up a model using tensor product based smooths. Designed to construct tensor products from any marginal smooths with a basis-penalty representation (with the restriction that each marginal smooth must have only one penalty).

Usage

```
te(..., k=NA,bs="cr",m=NA,d=NA,by=NA,fx=FALSE,
              np=TRUE,xt=NULL,id=NULL,sp=NULL,pc=NULL)
ti(..., k=NA,bs="cr",m=NA,d=NA,by=NA,fx=FALSE,
              np=TRUE,xt=NULL,id=NULL,sp=NULL,mc=NULL,pc=NULL)
```

TENSOR INTERACTION SMOOTH

```r
gam_interaction <- gam(claims ~ ev_ratio_int +
                         s(driver_age_int) +
                         ti(driver_age_int, ev_ratio_int),
                       data = dataOhlsson,
                       family = poisson(link = "log"),
                       offset = log(exposures))
```

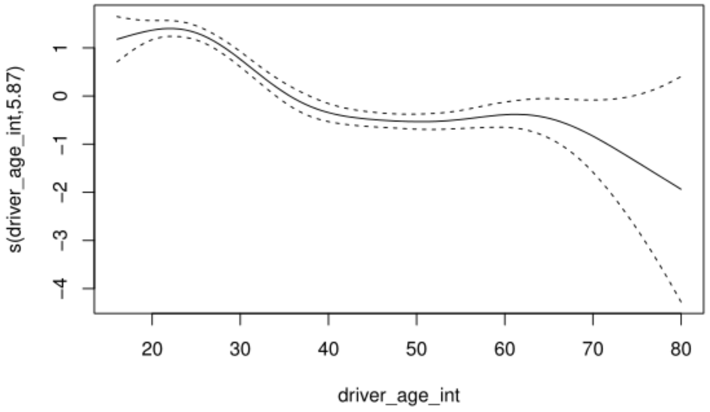
- One non-smoothed term: ev_ratio_int
- One smoothed main effect term: s(driver_age_int)
- One smoothed interaction term: ti(driver_age_int, ev_ratio_int)

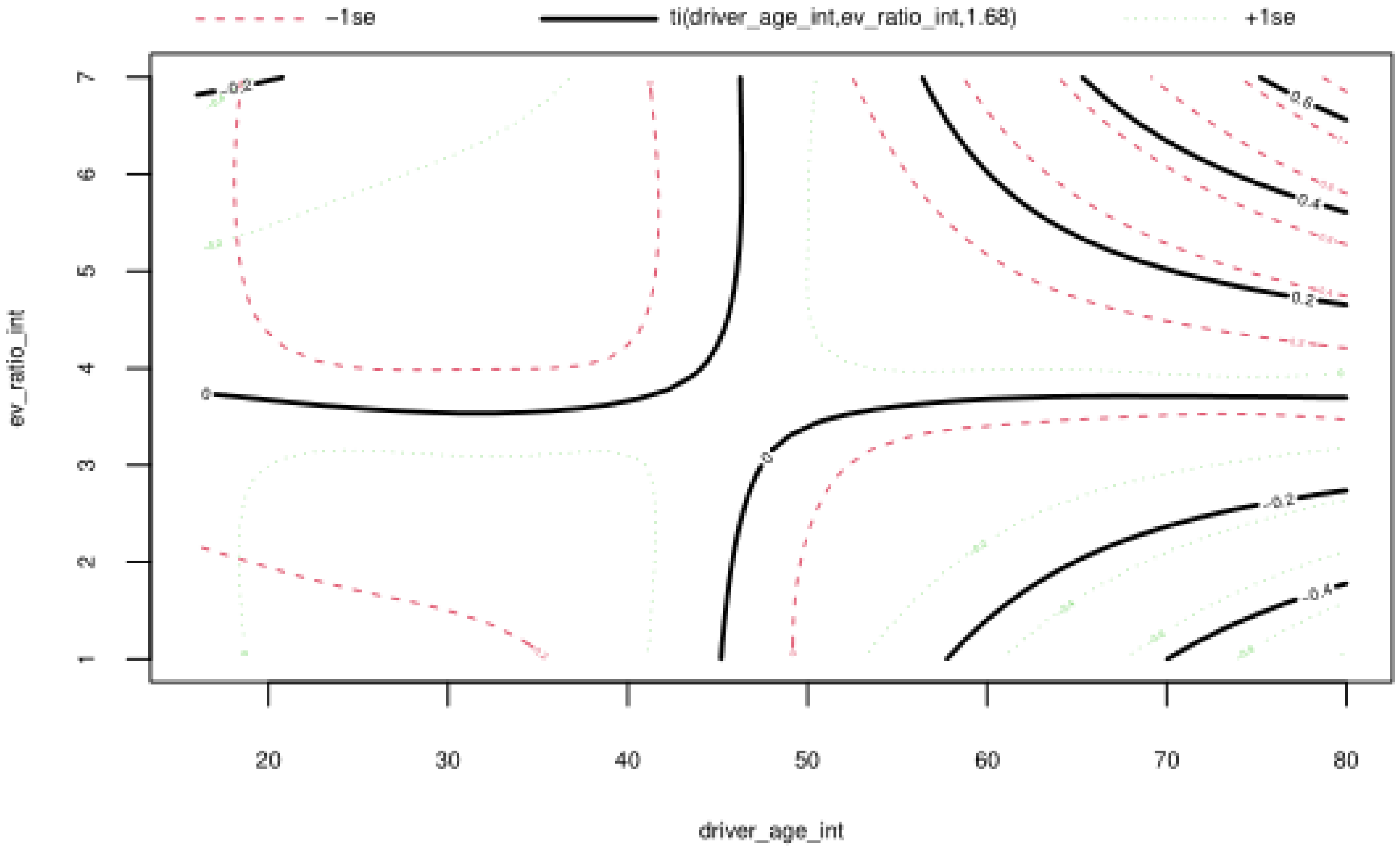
```r
coef(gam_interaction)
```

```
##                   (Intercept)                     ev_ratio_int
##                  -5.340512244                      0.166735483
##           s(driver_age_int).1              s(driver_age_int).2
##                   0.145637615                      2.418369431
##           s(driver_age_int).3              s(driver_age_int).4
##                   0.474262426                     -1.359533791
##           s(driver_age_int).5              s(driver_age_int).6
##                  -0.313843700                      0.964384322
##           s(driver_age_int).7              s(driver_age_int).8
##                  -0.070576425                      3.844865068
##           s(driver_age_int).9  ti(driver_age_int,ev_ratio_int).1

##                  -0.381079013                     -0.015337622
##  ti(driver_age_int,ev_ratio_int).2  ti(driver_age_int,ev_ratio_int).3
##                  -0.053266027                     -0.053997244
##  ti(driver_age_int,ev_ratio_int).4  ti(driver_age_int,ev_ratio_int).5
##                  -0.044982751                      0.009762226
##  ti(driver_age_int,ev_ratio_int).6  ti(driver_age_int,ev_ratio_int).7
##                   0.100779115                      0.145915637
##  ti(driver_age_int,ev_ratio_int).8  ti(driver_age_int,ev_ratio_int).9
##                   0.177692961                      0.003612855
## ti(driver_age_int,ev_ratio_int).10 ti(driver_age_int,ev_ratio_int).11
##                   0.224270919                      0.357855181
## ti(driver_age_int,ev_ratio_int).12 ti(driver_age_int,ev_ratio_int).13
##                   0.437302847                      0.009408327
## ti(driver_age_int,ev_ratio_int).14 ti(driver_age_int,ev_ratio_int).15
##                   0.374600372                      0.588426938
## ti(driver_age_int,ev_ratio_int).16
##                   0.718851900
```

TENSOR INTERACTION SMOOTH

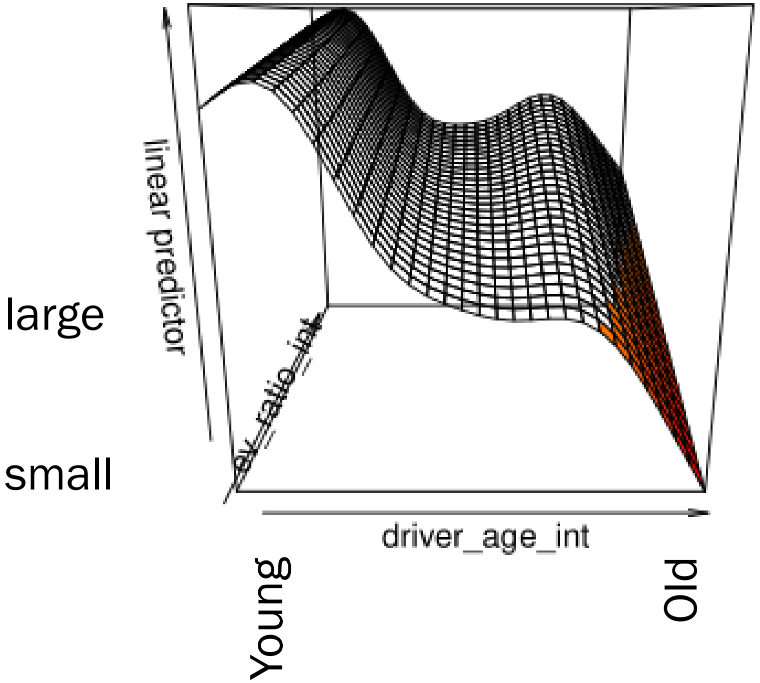
```
plot(gam_interaction)
```



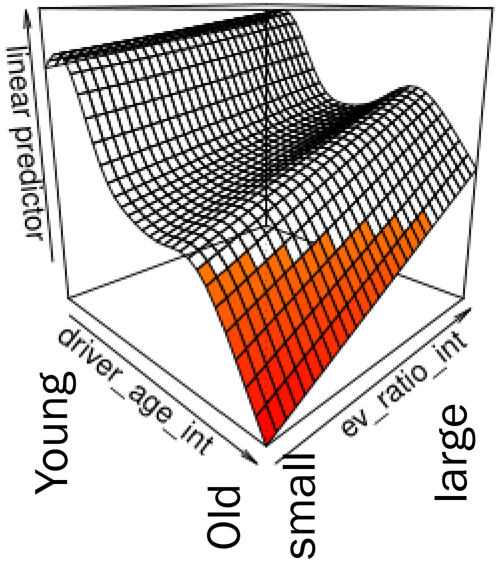


TENSOR INTERACTION SMOOTH

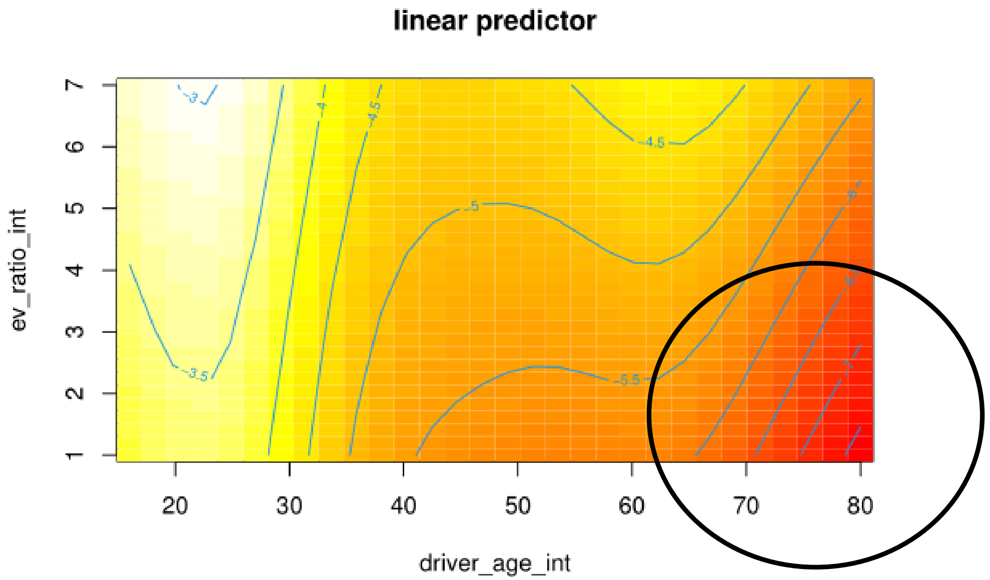
```
vis.gam(x = gam_interaction,
        view = c("driver_age_int", "ev_ratio_int"),
        plot.type = "persp")
```




```
vis.gam(x = gam_interaction,
        view = c("driver_age_int", "ev_ratio_int"),
        plot.type = "persp",
        theta = 45)
```




```
vis.gam(x = gam_interaction,
        view = c("driver_age_int", "ev_ratio_int"),
        plot.type = "contour")
```



Lowest for older drivers with small engines

EVALUATING GAM EXERCISE

TRAINING / TEST SPLIT

- Training Data (80%)
  - 80% of records with no claims
  - 80% of records with 1 or 2 claims
- Test Data (20%)
  - 20% of records with no claims
  - 20% of records with 1 or 2 claims

- Model built with the Training Data
- Decile plot built on the Test Data

SUMMARY() ON THE MODEL OBJECT

```r
gam_select <- gam(claims ~
                    ev_ratio_x1 +
                    vehicle_age_int +
                    s(driver_age_int) +
                    ti(driver_age_int, ev_ratio_int),
                  data = dataOhlsson_train,
                  family = poisson(link = "log"),
                  offset = log(exposures))

summary(gam_select)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## claims ~ ev_ratio_x1 + vehicle_age_int + s(driver_age_int) +
##     ti(driver_age_int, ev_ratio_int)
##
## Parametric coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.530023   0.165342 -27.398  < 2e-16 ***
## ev_ratio_x1      0.167210   0.037089   4.508 6.53e-06 ***
## vehicle_age_int -0.084936   0.007325 -11.595  < 2e-16 ***
## ---
## 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(driver_age_int)              5.373  6.434 321.96  <2e-16 ***
## ti(driver_age_int,ev_ratio_int) 6.809  8.602  12.53    0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0049   Deviance explained = 10.4%
## UBRE = -0.90391  Scale est. = 1         n = 49745
```

SUMMARY() ON THE MODEL OBJECT

```r
gam_select <- gam(claims ~
                    ev_ratio_x1 +
                    vehicle_age_int +
                    s(driver_age_int) +
                    ti(driver_age_int, ev_ratio_int),
                  data = dataOhlsson_train,
                  family = poisson(link = "log"),
                  offset = log(exposures))

summary(gam_select)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## claims ~ ev_ratio_x1 + vehicle_age_int + s(driver_age_int) +
##     ti(driver_age_int, ev_ratio_int)
##
## Parametric coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.530023   0.165342 -27.398  < 2e-16 ***
## ev_ratio_x1      0.167210   0.037089   4.508 6.53e-06 ***
## vehicle_age_int -0.084936   0.007325 -11.595  < 2e-16 ***
## ---
## 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(driver_age_int)              5.373  6.434 321.96  <2e-16 ***
## ti(driver_age_int,ev_ratio_int) 6.809  8.602  12.53    0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0049   Deviance explained = 10.4%
## UBRE = -0.90391  Scale est. = 1         n = 49745
```

SUMMARY() ON THE MODEL OBJECT

```r
gam_select <- gam(claims ~
                    ev_ratio_x1 +
                    vehicle_age_int +
                    s(driver_age_int) +
                    ti(driver_age_int, ev_ratio_int),
                  data = dataOhlsson_train,
                  family = poisson(link = "log"),
                  offset = log(exposures))

summary(gam_select)
```

```
##
## Family: poisson
## Link function: log
##
## Formula:
## claims ~ ev_ratio_x1 + vehicle_age_int + s(driver_age_int) +
##     ti(driver_age_int, ev_ratio_int)
##
## Parametric coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.530023   0.165342 -27.398  < 2e-16 ***
## ev_ratio_x1      0.167210   0.037089   4.508 6.53e-06 ***
## vehicle_age_int -0.084936   0.007325 -11.595  < 2e-16 ***
## ---
## 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(driver_age_int)               5.373  6.434 321.96  <2e-16 ***
## ti(driver_age_int,ev_ratio_int) 6.809  8.602  12.53    0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0049   Deviance explained = 10.4%
## UBRE = -0.90391  Scale est. = 1         n = 49745
```

SUMMARY() ON THE MODEL OBJECT

```r
gam_select <- gam(claims ~
                    ev_ratio_x1 +
                    vehicle_age_int +
                    s(driver_age_int) +
                    ti(driver_age_int, ev_ratio_int),
                  data = dataOhlsson_train,
                  family = poisson(link = "log"),
                  offset = log(exposures))

summary(gam_select)
```

- ti(driver_age_int, ev_ratio_int) has a p-value of 0.15

```
##
## Family: poisson
## Link function: log
##
## Formula:
## claims ~ ev_ratio_x1 + vehicle_age_int + s(driver_age_int) +
##     ti(driver_age_int, ev_ratio_int)
##
## Parametric coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.530023   0.165342 -27.398  < 2e-16 ***
## ev_ratio_x1      0.167210   0.037089   4.508 6.53e-06 ***
## vehicle_age_int -0.084936   0.007325 -11.595  < 2e-16 ***
## ---
## 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(driver_age_int)               5.373  6.434 321.96  <2e-16 ***
## ti(driver_age_int,ev_ratio_int) 6.809  8.602  12.53    0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0049   Deviance explained = 10.4%
## UBRE = -0.90391  Scale est. = 1         n = 49745
```

SUMMARY() ON THE MODEL OBJECT

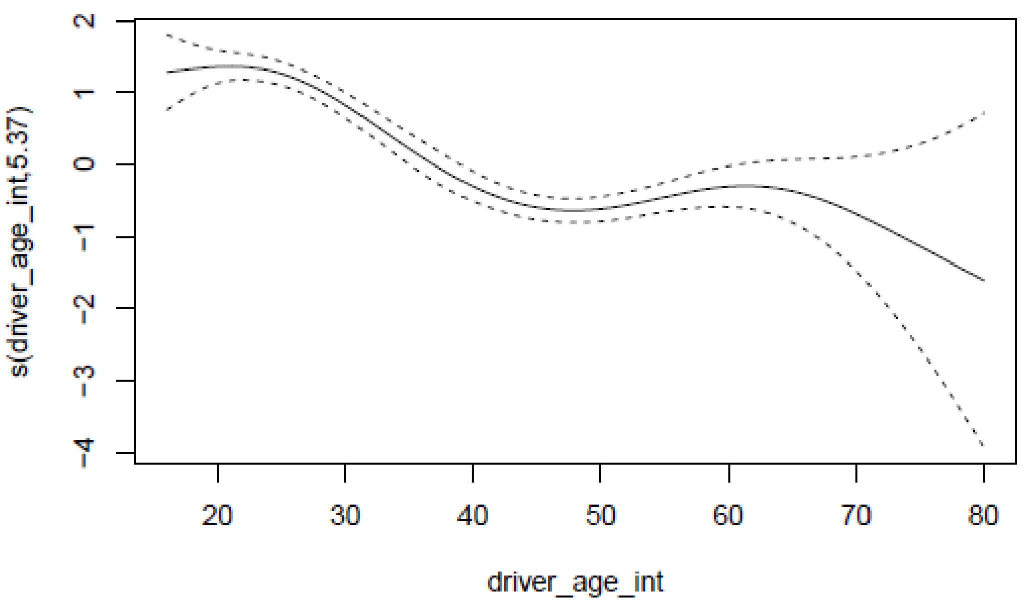
```
gam_select <- gam(claims ~
                    ev_ratio_x1 +
                    vehicle_age_int +
                    s(driver_age_int) +
                    ti(driver_age_int, ev_ratio_int),
                  data = dataOhlsson_train,
                  family = poisson(link = "log"),
                  offset = log(exposures))

summary(gam_select)
```

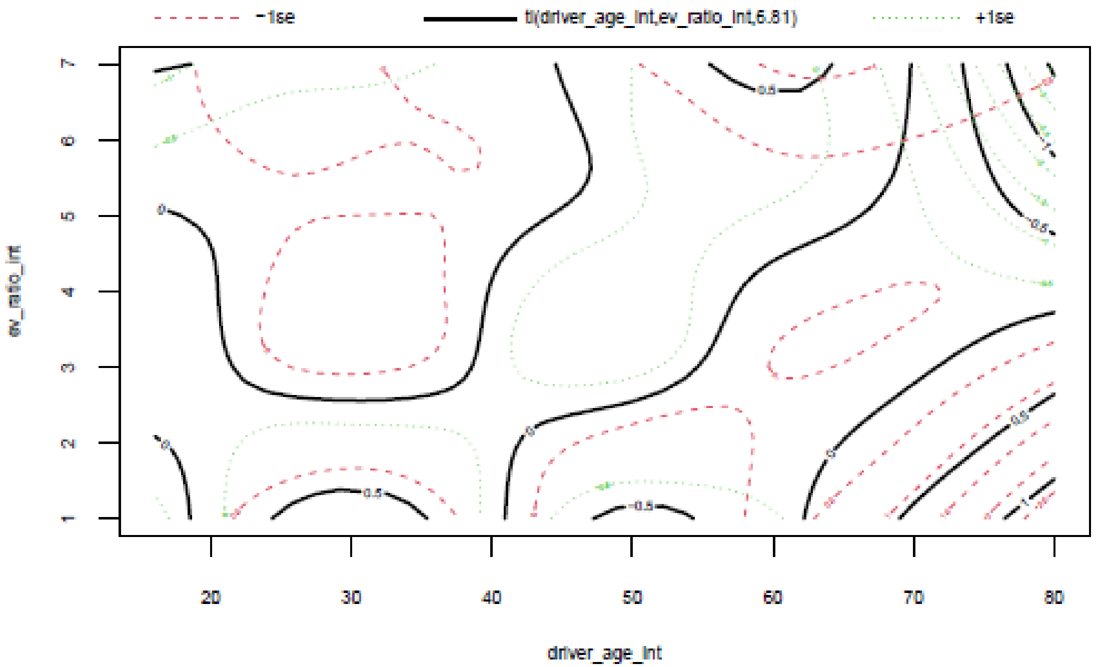
- ti(driver_age_int, ev_ratio_int) has a p-value of 0.15
- R-sq. (adj) = 0.0049

```
##
## Family: poisson
## Link function: log
##
## Formula:
## claims ~ ev_ratio_x1 + vehicle_age_int + s(driver_age_int) +
##     ti(driver_age_int, ev_ratio_int)
##
## Parametric coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -4.530023   0.165342 -27.398  < 2e-16 ***
## ev_ratio_x1      0.167210   0.037089   4.508 6.53e-06 ***
## vehicle_age_int -0.084936   0.007325 -11.595  < 2e-16 ***
## ---
## 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(driver_age_int)                5.373  6.434 321.96  <2e-16 ***
## ti(driver_age_int,ev_ratio_int) 6.809  8.602  12.53    0.15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0049   Deviance explained = 10.4%
## UBRE = -0.90391  Scale est. = 1         n = 49745
```

PLOT() ON THE MODEL OBJECT

```
plot(gam_select)
```



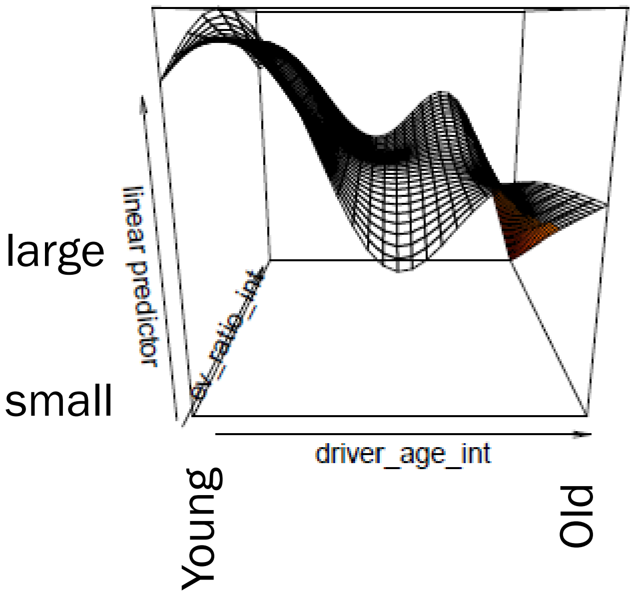


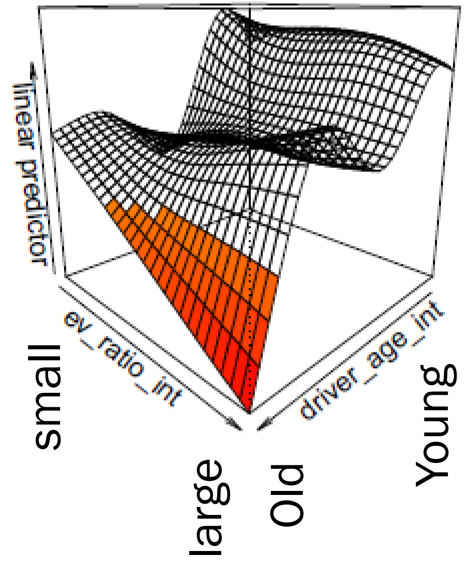
VIS.GAM() ON THE INTERACTED TERMS

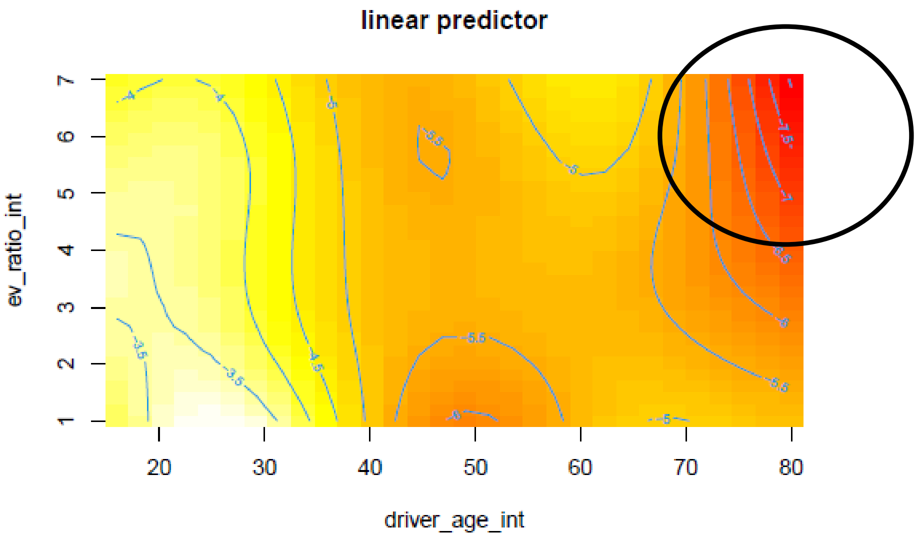
```
vis.gam(x = gam_select,
        view = c("driver_age_int", "ev_ratio_int"),
        plot.type = "persp")
```

```
vis.gam(x = gam_select,
        view = c("driver_age_int", "ev_ratio_int"),
        plot.type = "persp",
        theta = 135)
```

```
vis.gam(x = gam_select,
        view = c("driver_age_int", "ev_ratio_int"),
        plot.type = "contour")
```







Lowest for older drivers with large engines

GAM.CHECK() AUTOMATED RESULTS

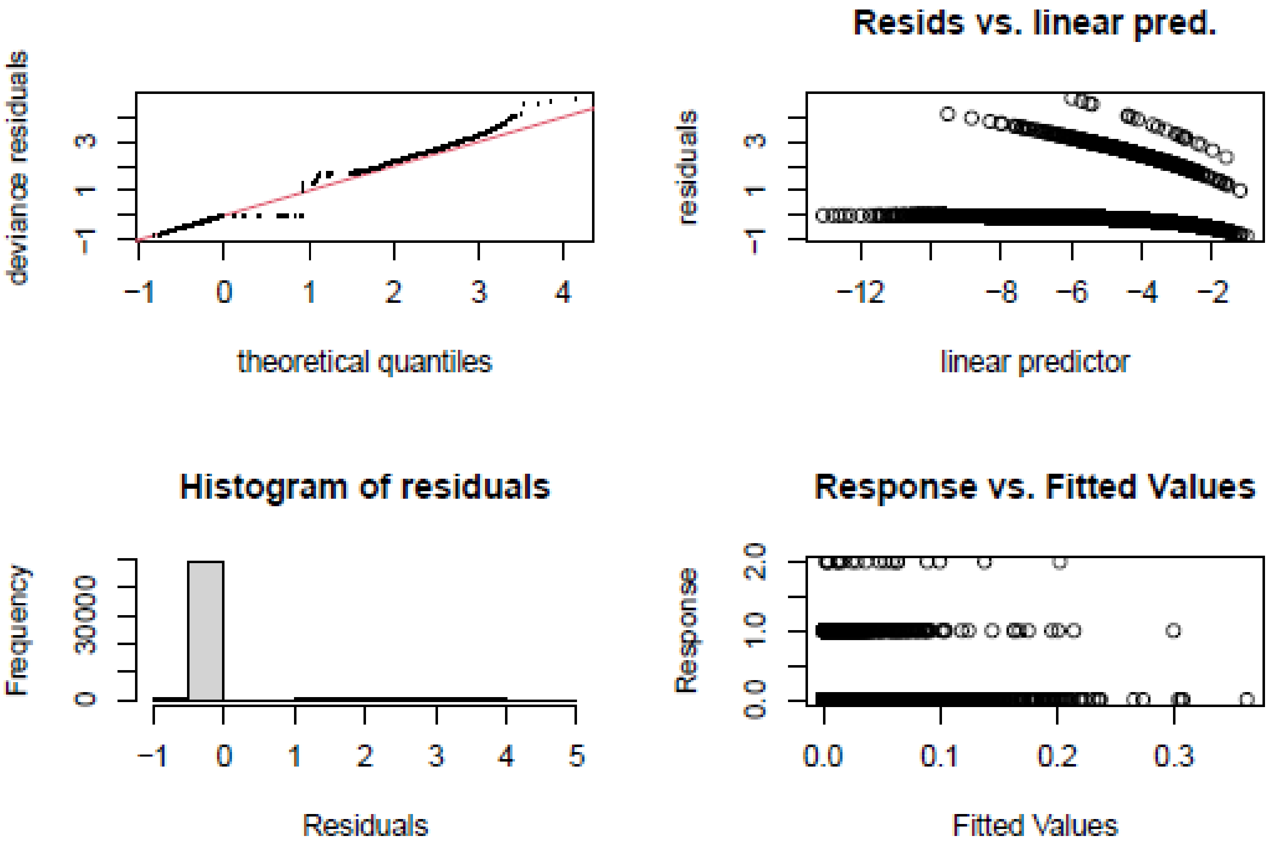
- We want HIGH values for these p-values
  - We are checking if one of the smooths is predictive of the residuals
  - We sure hope it isn't!
- We want k-index approximately 1

```
##
## Method: UBRE   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-1.086673e-08,2.286584e-08]
## (score -0.9039131 & scale 1).
## Hessian positive definite, eigenvalue range [8.398131e-06,3.118174e-05].
## Model rank =  28 / 28
##
## 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(driver_age_int)                9.00  5.37    0.89    0.12
## ti(driver_age_int,ev_ratio_int) 16.00  6.81    0.92    0.60
```

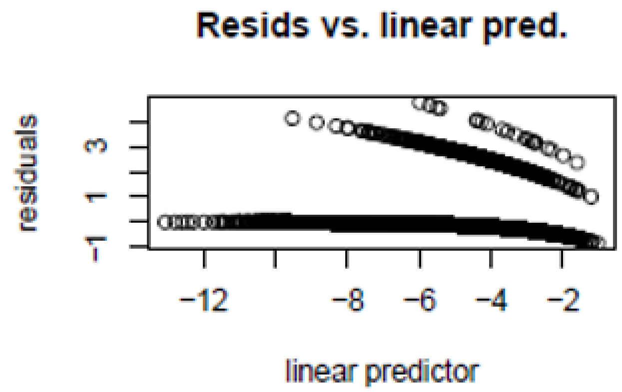
GAM.CHECK() AUTOMATED PLOTS

```
gam.check(gam_select)
```

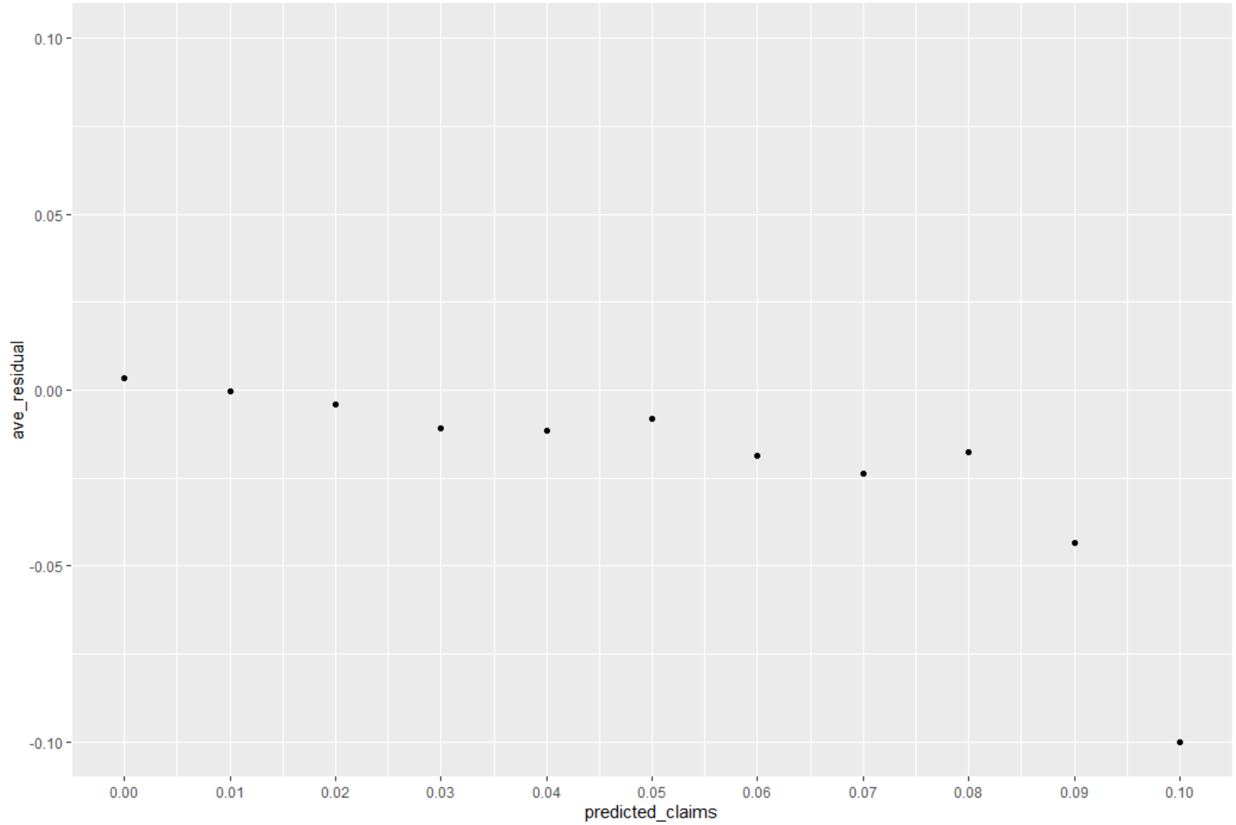
- These plots can be challenging to evaluate for discrete values
- The residual plots look odd because the target variable is either 0, 1, or 2



AVERAGE RESIDUAL BY PREDICTED VALUE



- Here, predicted mean frequency is rounded to the nearest 0.01
- We are hoping for average residuals to be randomly distributed around zero
- This plot fails this test



CONCURVITY() ON THE MODEL OBJECT

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

- Worst, observed, estimate are different measurements of concurvity
- Worst is the most pessimistic
- Rule of thumb: Worst case concurvity > 0.8 is too much

```
##                para s(driver_age_int) ti(driver_age_int,ev_ratio_int)
## worst    0.912225        0.30529430                      0.29609896
## observed 0.912225        0.03634251                      0.01815659
## estimate 0.912225        0.05137036                      0.02160891
```

CONCURVITY() ON THE MODEL OBJECT

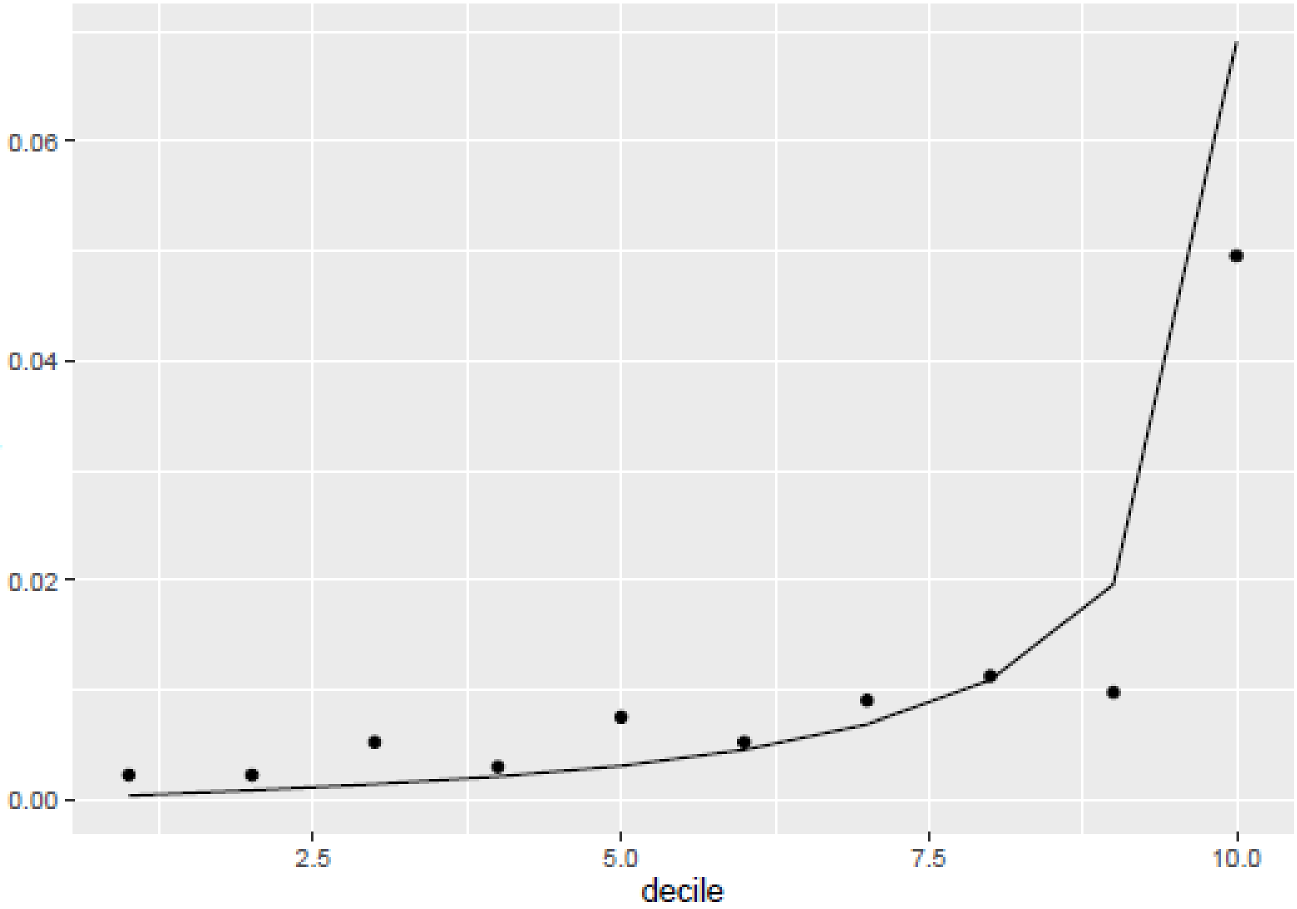
```
concurvity(gam_select, full = TRUE)
```

```
##               para s(driver_age_int) ti(driver_age_int,ev_ratio_int)
## worst     0.912225        0.30529430                      0.29609896
## observed  0.912225        0.03634251                      0.01815659
## estimate  0.912225        0.05137036                      0.02160891
```

- Worst, observed, estimate are different measurements of concurvity
- Worst is the most pessimistic
- Rule of thumb: Worst case concurvity > 0.8 is too much

CONCLUSION

- Positives
  - Concurvity Metrics appear okay
- Negatives
  - The ti(driver_age, ev_ratio) smooth p-value from the summary = 0.15 (too high)
  - Driver age smooth plot is counterintuitive for 16 year old drivers
  - The s(driver age) p-value from gam.check() = 0.12 (too low)
  - The average residual by predicted value plot show the residuals do not have a mean of zero and higher predictions have a negative residual
  - The decile plot on test data confirms that our highest predictions are too high



HOW TO IMPROVE?

- Adjust number of basis functions with k
- Adjust the smoothing parameters (wiggliness penalty) with sp
- Try alternate smoothing functions with bs
- Use a larger, more credible dataset!

QUESTIONS FOR GAM'S

- Provide GAM Output [from summary()]
  - Effective Degrees of Freedom (EDF)
  - Reference Degrees of Freedom (RDF)
  - Chi-sq or F Statistic
  - **P-values (hopefully low)**
  - **Adjusted R-Squared (hopefully closer to 1.0)**
  - **Deviance Explained**
  - Scale Estimate
  - **N – Number of Observations (hopefully large – credible)**

- Provide GAM Output from the gam.check() statement.
  - Method
  - Optimizer
  - **Convergence Iterations (hopefully converged)**
  - Hessian Eigenvalues, Eigenvalue Range
  - Model Rank
  - Basis Dimension (k')
  - EDF
  - **K-index (hopefully close to 1.0)**
  - **P-values (hopefully high)**

QUESTIONS FOR GAM'S

- Rationales
  - Type / number of smooth terms
  - K value(s)
  - Smoothing parameter value(s)
  - Optimization method
- **Correlation matrix non-smoothed terms**
- **Concurvity metrics for smoothed terms**
- **AIC after each term in the model**

- Plots
  - **Plot of the smooth terms** (since there are no betas) [plot()]
    - Include confidence intervals
  - **Visualizations for interactions** [vis.gam()]
  - **Residual plots**
    - gam.check() plots (useful for continuous target variables)
    - Average residual by predicted value (or bucket)
- Lift charts
  - Lorenz curve
  - **Quantile Plot**