Sample Template - Annual Meeting 2019

Nick Hanewinckel, FSA, CERA

Contents

Package Loader

Do you have all the packages we are going to talk about? Are they shiny* and new?

*Disclaimer: this comment is not affiliated with the package 'shiny'made by rstudio.

```
library(data.table)
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

Get/Prepare Data

```
pth <- 'C:/Users/hfi/Documents/ILEC_2009-15 Data 20180601.txt'
dat <- fread(pth,stringsAsFactors = TRUE,nrows=1000000,check.names = TRUE)
```

data.table Cheat Sheets

Link1

Link2

Link3 - Longer, but good

Curious how I embedded all these cool links?

rmarkdown Cheat Sheet

See a pattern here? https://rstudio.com/resources/cheatsheets/

Demo - Speed Test

```
#Reading Test Sample Records
temp <- Sys.time()
invisible(read.csv(pth,header=TRUE,stringsAsFactors = TRUE,nrows = 1000000))
print("read.csv time: ");print(Sys.time()-temp)
```

```
## [1] "read.csv time: "
```

```
## Time difference of 1.080927 mins
```

```
temp <- Sys.time()
invisible(fread(pth,stringsAsFactors = TRUE,nrows = 1000000))
print("fread time: ");print(Sys.time()-temp)
```

```
## [1] "fread time: "
```

```
## Time difference of 2.768635 secs
```

Data Wrangle

```
setnames(dat,'Number.of.Deaths','Deaths')
setnames(dat,'Policies.Exposed','Exposure')
invisible(sapply(names(dat),function(x) setnames(dat,x,gsub('\\.','',x))))

#Get names of fields
names(dat)
```

```
##  [1] "ObservationYear"                  "CommonCompanyIndicator57"
##  [3] "PreferredIndicator"               "Gender"
##  [5] "SmokerStatus"                     "InsurancePlan"
##  [7] "IssueAge"                         "Duration"
##  [9] "AttainedAge"                      "AgeBasis"
## [11] "FaceAmountBand"                   "IssueYear"
## [13] "NumberofPreferredClasses"         "PreferredClass"
## [15] "SOAAnticipatedLevelTermPeriod"    "SOAGuaranteedLevelTermPeriod"
## [17] "SOAPostleveltermindicator"        "Select_Ultimate_Indicator"
## [19] "Deaths"                           "DeathClaimAmount"
## [21] "Exposure"                         "AmountExposed"
## [23] "ExpectedDeathQX7580EbyAmount"     "ExpectedDeathQX2001VBTbyAmount"
## [25] "ExpectedDeathQX2008VBTbyAmount"   "ExpectedDeathQX2008VBTLUbyAmount"
## [27] "ExpectedDeathQX2015VBTbyAmount"   "ExpectedDeathQX7580EbyPolicy"
## [29] "ExpectedDeathQX2001VBTbyPolicy"   "ExpectedDeathQX2008VBTbyPolicy"
## [31] "ExpectedDeathQX2008VBTLUbyPolicy" "ExpectedDeathQX2015VBTbyPolicy"
```

```
#Show row count for rows with no exposure
dat[Exposure==0,.N]
```

```
## [1] 6979
```

```
dat[AmountExposed==0,.N]
```

```
## [1] 6979
```

```
#Fix Rows with No Exposure
dat <- dat[Exposure > 0 & AmountExposed >0]

#Quantify non-level or unknown term products
table(dat$SOAGuaranteedLevelTermPeriod)
```

```
##
## 10 yr guaranteed 15 yr guaranteed 20 yr guaranteed 25 yr guaranteed
##            73134            47609            56041             1303
## 30 yr guaranteed  5 yr guaranteed   N/A (Not Term)   Not Level Term
##            23311            38725           633335            28938
##          Unknown
##            90625
```

```
#Remove non-level Term
dat <- dat[!SOAGuaranteedLevelTermPeriod %in% c('Unknown','Not Level Term')]
#Verfy Removal
table(dat$SOAGuaranteedLevelTermPeriod)
```

```
##
## 10 yr guaranteed 15 yr guaranteed 20 yr guaranteed 25 yr guaranteed
##            73134            47609            56041             1303
## 30 yr guaranteed  5 yr guaranteed   N/A (Not Term)   Not Level Term
##            23311            38725           633335                0
##          Unknown
##                0
```

```
#See Where Death > Exposure
dat[Deaths>Exposure,.N]
```

```
## [1] 1329
```

```
#I will NOT clean this, as I plan to use a poisson regresison anyway
```

Plotting

ggplot2 is your best friend!

Get your cheatsheet here: Rstudio's ggplot cheat sheet

Format Data

Data must be 'plottable' (formatted like your desired plot)

I like to make a temporary table specific to the plot. For this, I want an A/E by Count. Can you figure out how to make one by amount?

I will make a few temporary tables to show how this is necessary for similar views with more variables.

I include variables that may seem "extra", but will be used to convey additional graph information.

GET YOUR MONEY'S WORTH OUT OF INCLUDING AS MUCH INFO AS REASONABLE!

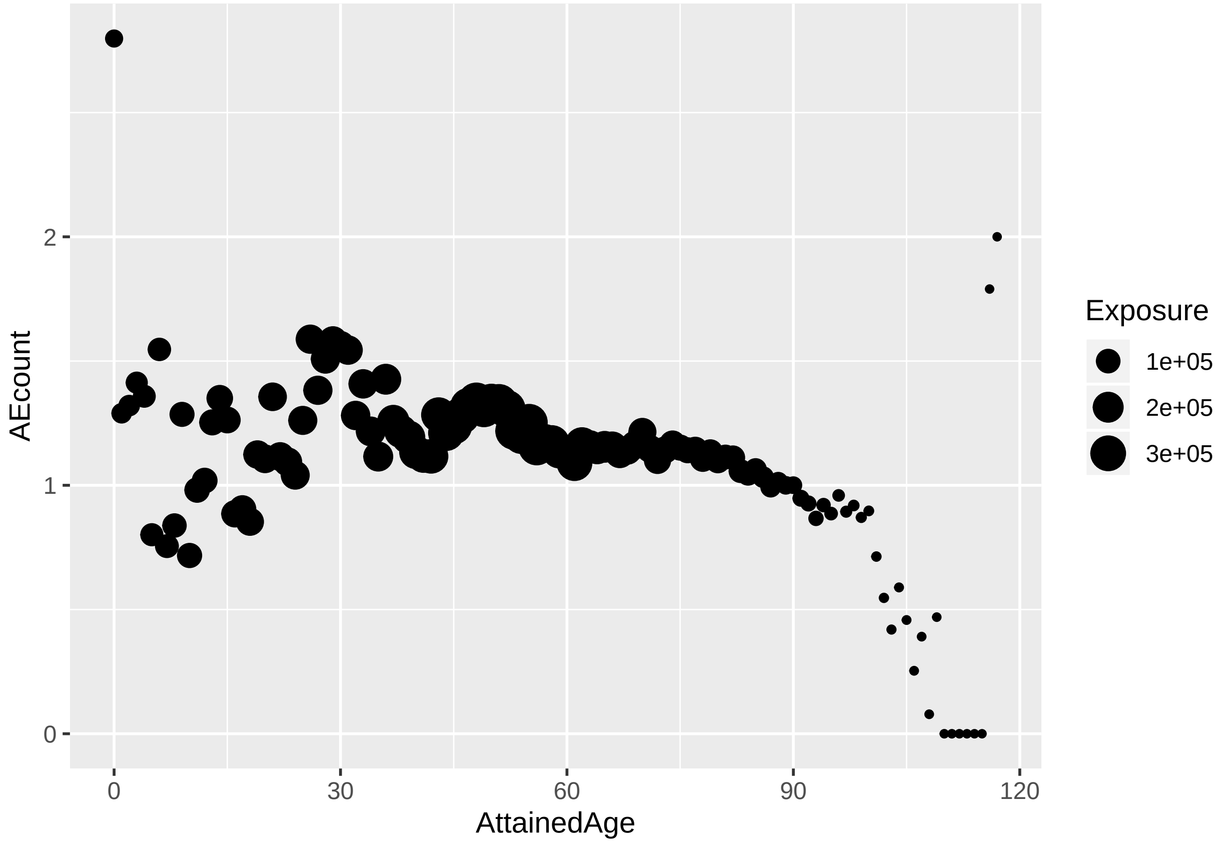
```
plotMeSimple <- dat[,.(Deaths=sum(Deaths),EDvbt15=sum(ExpectedDeathQX2015VBTbyPolicy),
                       AEcount=sum(Deaths)/sum(ExpectedDeathQX2015VBTbyPolicy),
                       Exposure=sum(Exposure)),
                    by=AttainedAge][order(AttainedAge)]
#Note - I did not "need" [order(AttainedAge)], but it can be nice to have data ordered!
plotMeComplex <- dat[,.(Deaths=sum(Deaths),EDvbt15=sum(ExpectedDeathQX2015VBTbyPolicy),
                       AEcount=sum(Deaths)/sum(ExpectedDeathQX2015VBTbyPolicy),
                       Exposure=sum(Exposure)),
                    by=.(Duration,Gender,SmokerStatus,InsurancePlan)]
```

Plot

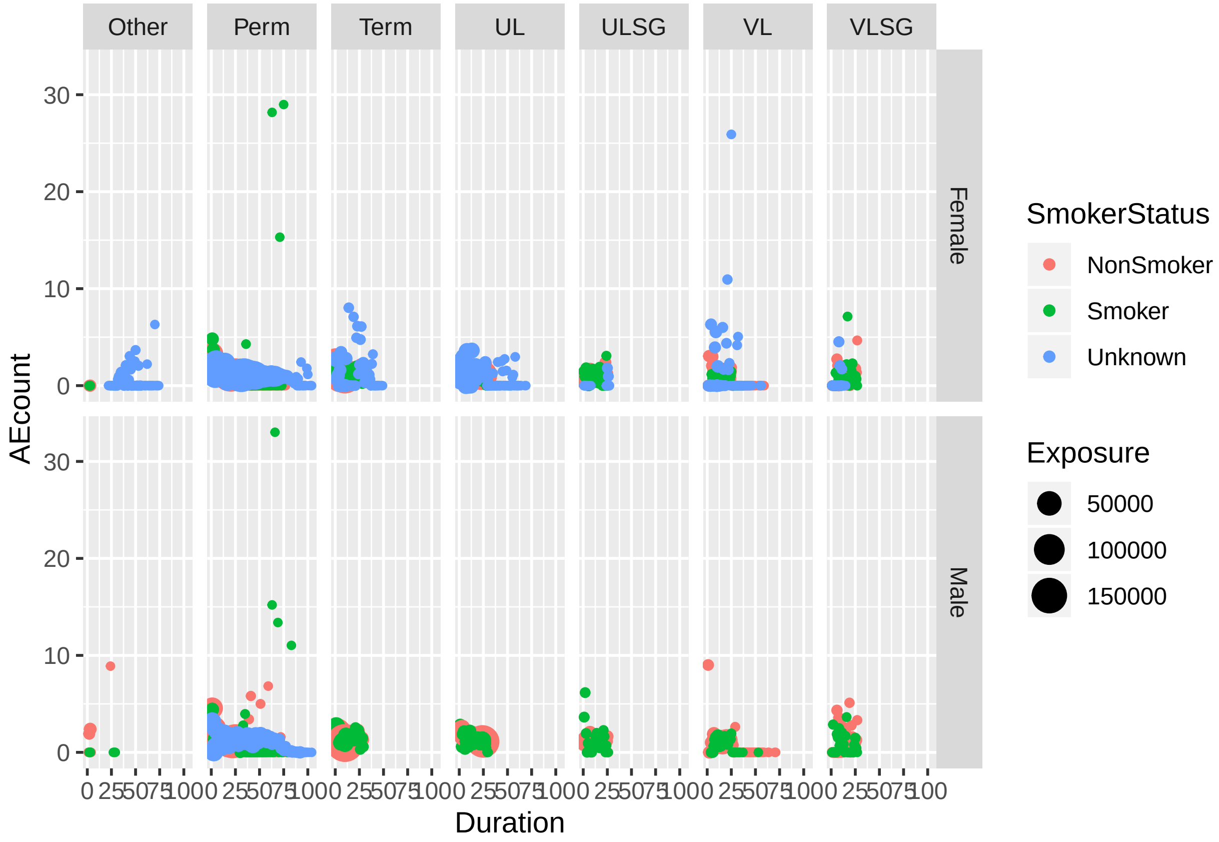
Here, note that it's imporant to use *all* the categorical variables I defined in my temporary tables. While we can't (obviously) use variables that don't exist, if we don't use all of them the plot may look like it has "extraneous" graphing. An example would be if I didn't use Gender in a plot, but it was in my table. Now I'd have two "dots" for each age.

Additional numeric variables (e.g. Exposure) are less strict. I can choose to use Exposure to indicate the size of my plots, or not, with no ill side effects.

```
ggplot(data=plotMeSimple,aes(x=AttainedAge,y=AEcount,size=Exposure)) +
  geom_point()
```



```
ggplot(data=plotMeComplex,aes(x=Duration,y=AEcount,size=Exposure,color=SmokerStatus)) +
  geom_point() +
  facet_grid(Gender~InsurancePlan)
```

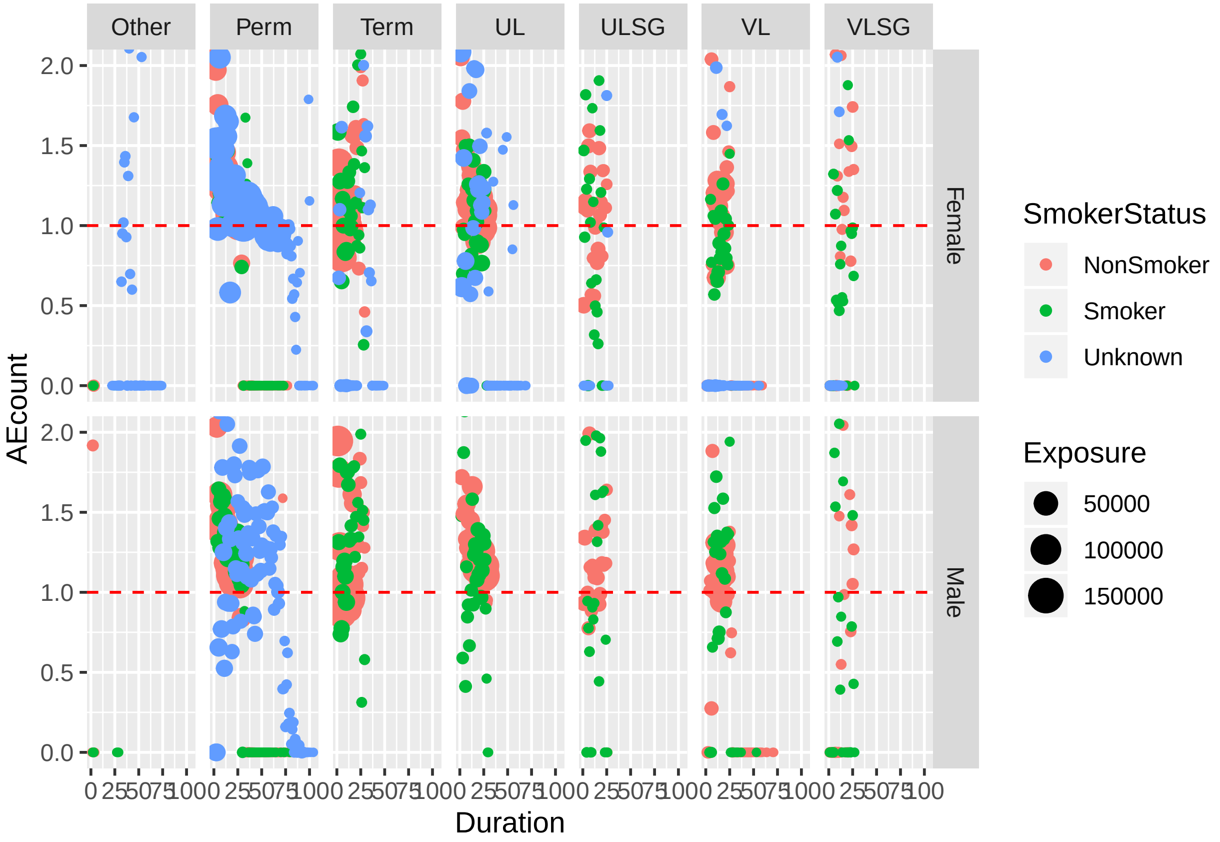


Wait

I can't see that last one?

Okay, sometimes we have to get creative when we look at how a graph turns out:

```
ggplot(data=plotMeComplex,aes(x=Duration,y=AEcount,size=Exposure,color=SmokerStatus)) +
  geom_point() +
  facet_grid(Gender~InsurancePlan)+
  coord_cartesian(ylim=c(0,2)) +
  geom_hline(yintercept=1,color='red',linetype='dashed')
```



Takeaways

This is not a great dataset…to be expected with a small subset of a full dataset. A big takeaway challenge - how can we add CIs so we can tell when an A/E is meaningful? A hint…?qpois.

Model

Now the heart of what many want to do - modelling. Since this is a somewhat incomplete dataset, this gives us a chance to see how to use a predictive model (glm) to adjust a given basis.

Let's assume that this company uses the 2015 VBT table as its baseline assumptions. They would like to know what factors or multiples they could apply to this table to make it fit there business better.

Offset

We will fit a model to 'correct' the basis by using that basis as an Offset.

What is an offset?

In Regression, offsets are variables that always receive a coefficient of 1. The model will not "solve" for their coefficient, but will always assume it's one.

Why do we usually use it

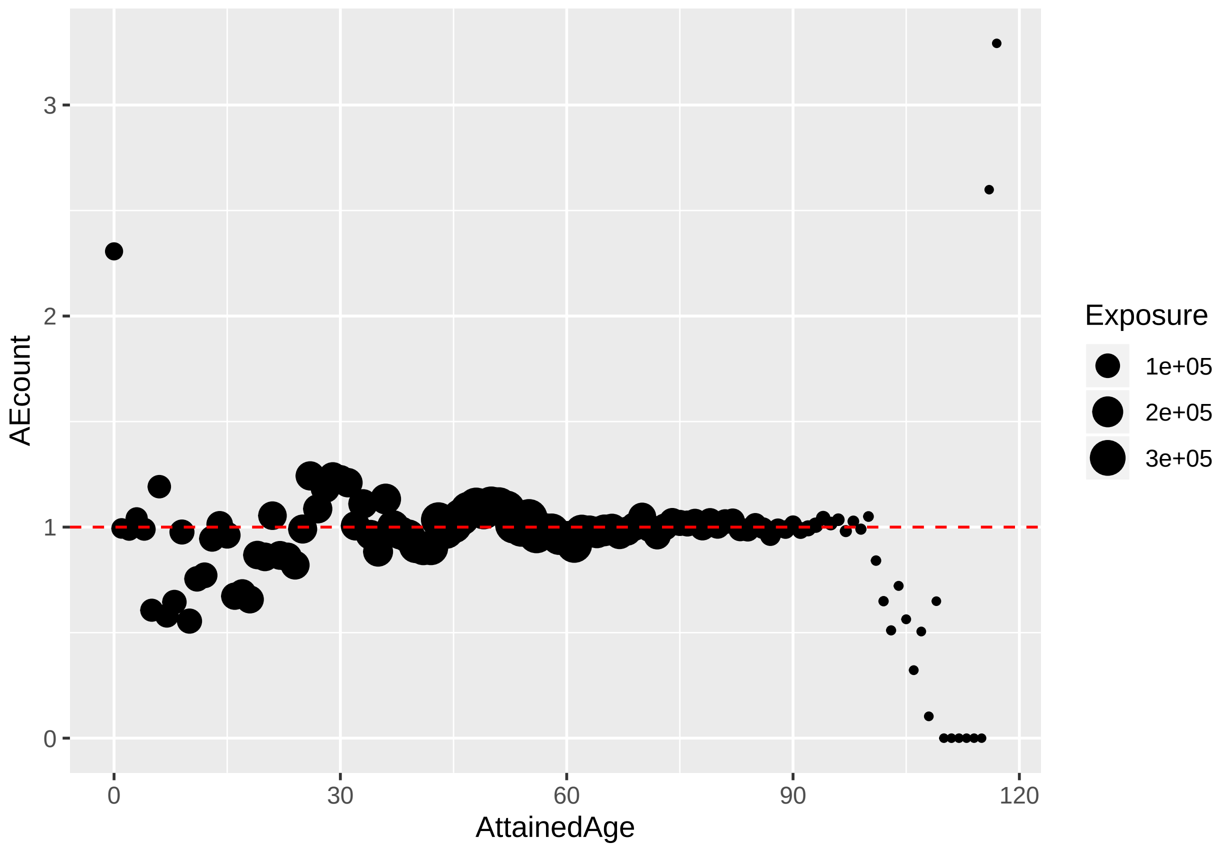
A common use for offset is for the "exposure" in, say, a poisson model. This means $e^{MortalityRate}$ always gets multiplied by $e^{Exposure}$ just like in a poisson regression. This implies our $Offset = ln(Exposure)$

How will we use it now

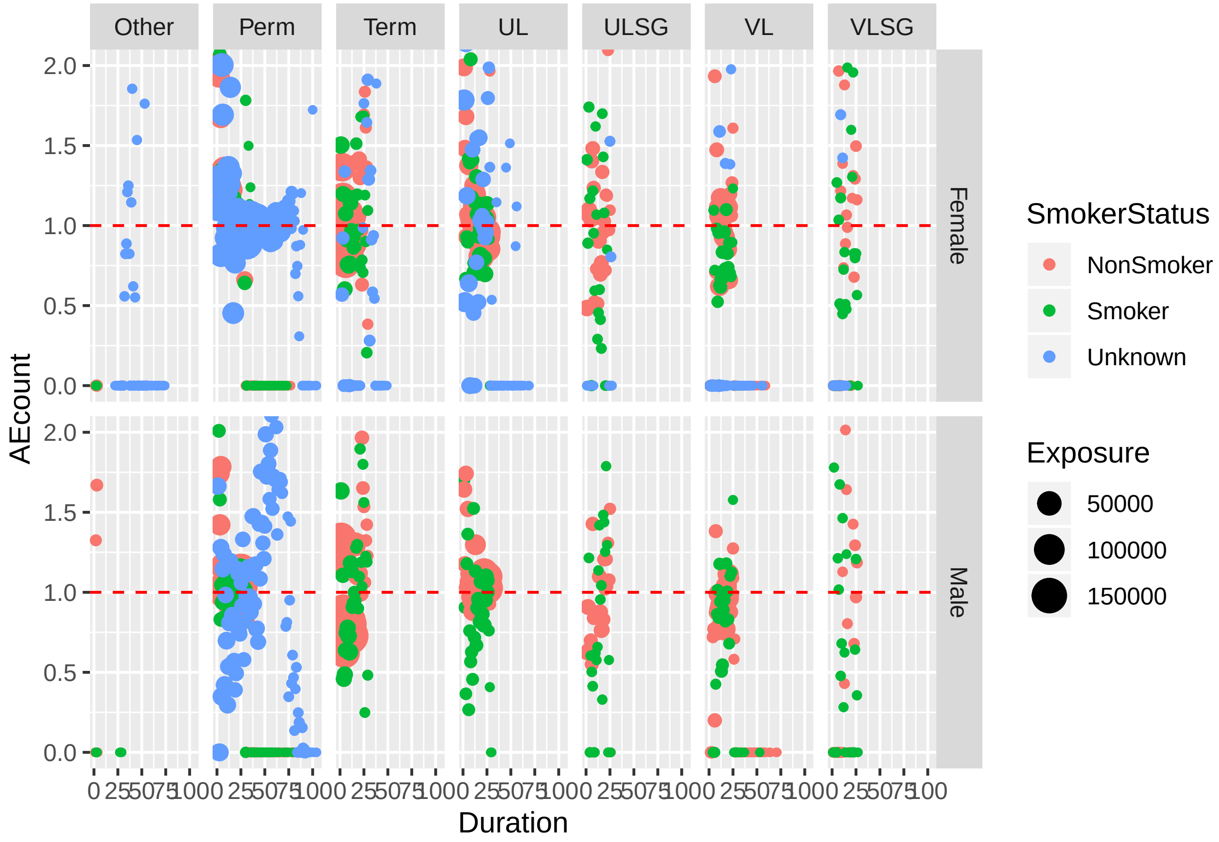
With the magic of algebra, if we use $Offset = ln(ExpectedDeaths)$ we essentially make our offset= $log(HazardRate * Exposure)$ and our model will solve for the coefficients that take the Mortality Assumption to an adjusted Mortality Assumption by solving for Deaths.

In Action

```r
model <- glm(Deaths ~ (AttainedAge + Duration + Gender + SmokerStatus)^2,
             data=dat,offset=log(ExpectedDeathQX2015VBTbyPolicy),family='poisson')

dat[,predDeaths:=predict(model,newdata=dat,type='response')]
dat[,modelQX:=predDeaths/Exposure]

#Duplicate prior plots
plotMeSimple <- dat[,.(Deaths=sum(Deaths),EDmodel=sum(predDeaths),
                       AEcount=sum(Deaths)/sum(predDeaths),
                       Exposure=sum(Exposure)),
                    by=AttainedAge]
plotMeComplex <- dat[,.(Deaths=sum(Deaths),EDmodel=sum(predDeaths),
                       AEcount=sum(Deaths)/sum(predDeaths),
                       Exposure=sum(Exposure)),
                    by=.(Duration,Gender,SmokerStatus,InsurancePlan)]
ggplot(data=plotMeSimple,aes(x=AttainedAge,y=AEcount,size=Exposure)) +
  geom_point()+ geom_hline(yintercept=1,color='red',linetype='dashed')
```



```
ggplot(data=plotMeComplex,aes(x=Duration,y=AEcount,size=Exposure,color=SmokerStatus)) +
  geom_point() +
  facet_grid(Gender~InsurancePlan)+
  coord_cartesian(ylim=c(0,2)) +
  geom_hline(yintercept=1,color='red',linetype='dashed')
```



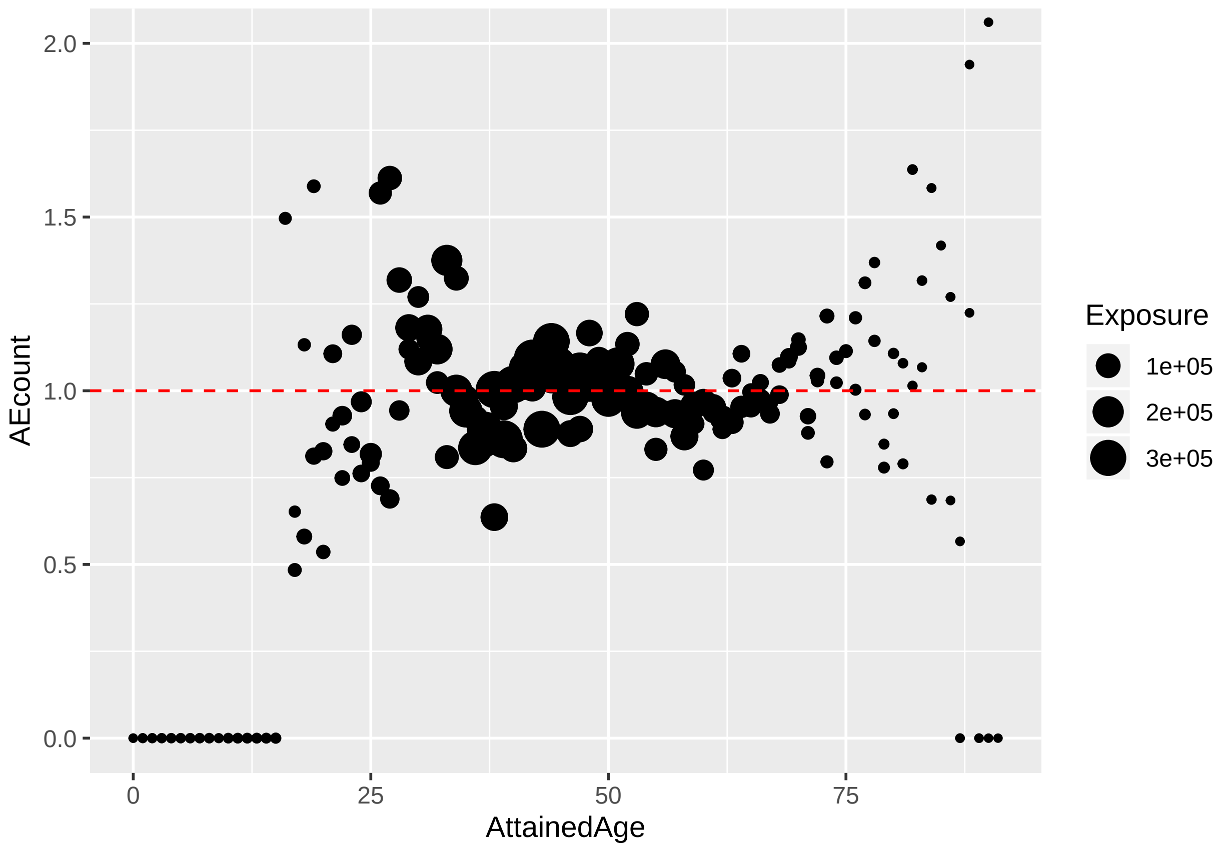
These Results Are. . . fine?

Yes! They are not great! Why?

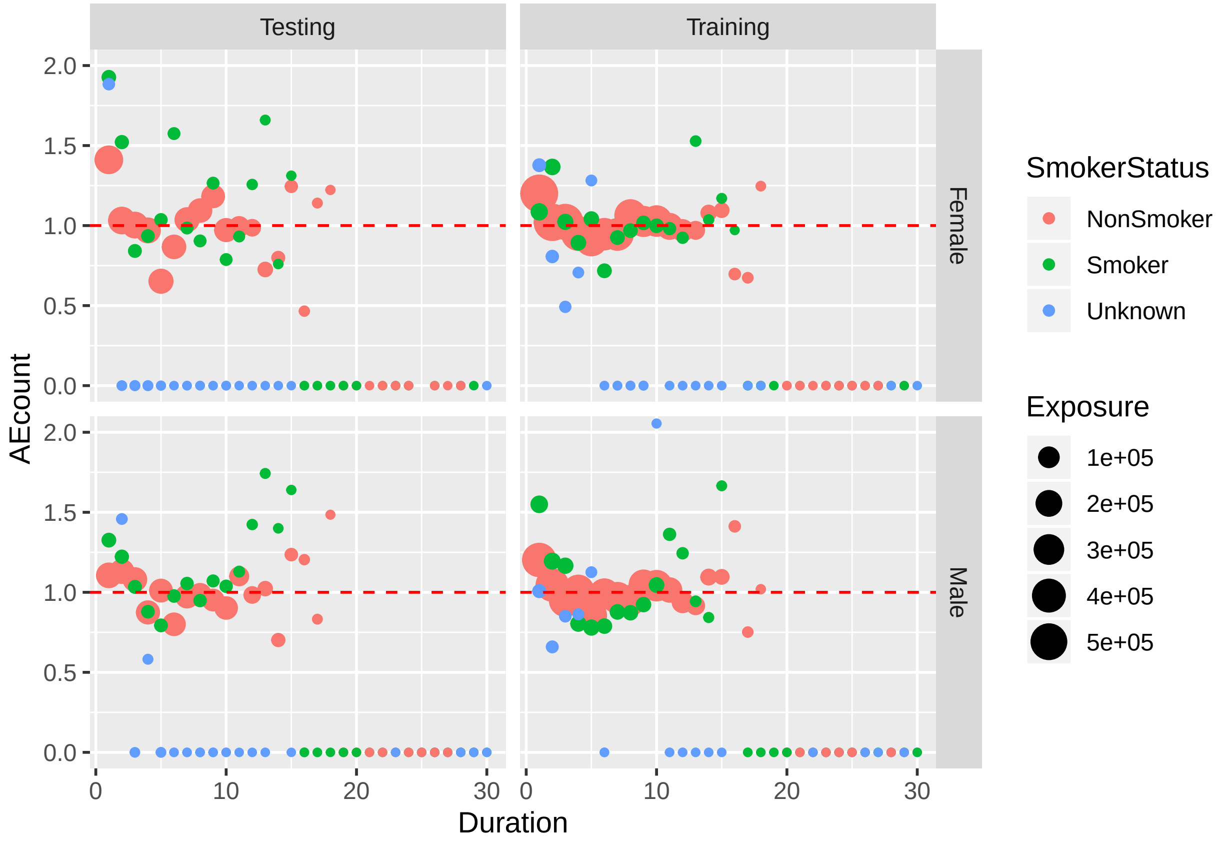
- Using a subset of data
  - Not much we can do about this
  - For fun we'll add more records to see if this helps
- Using a wide range of data
  - Let's focus just on term
- Overfit!
  - I may have over-specified this model and its interactions
  - I didn't use a testing/training set, which would catch it
  - I should have (perhaps) used a penalized model for so many variables
- Other Data Issues
  - Remove post-level term

```
#ReRead bigger data
dat <- fread(pth,stringsAsFactors = TRUE,nrows=5000000,check.names = TRUE)
setnames(dat,'Number.of.Deaths','Deaths')
setnames(dat,'Policies.Exposed','Exposure')
invisible(sapply(names(dat),function(x) setnames(dat,x,gsub('\\.','',x))))
#Fix Rows with No Exposure
dat <- dat[Exposure > 0 & AmountExposed >0]
#Term Only
dat <- dat[InsurancePlan == 'Term']
#Remove non-level Term
```

```
dat <- dat[!SOAGuaranteedLevelTermPeriod %in% c('Unknown','Not Level Term')]
#Remove post-level term
dat <- dat[SOAPostleveltermindicator == 'Within Level Term']
#Split Test/Train
trainingRows <- createDataPartition(dat$AttainedAge,p=.7,list=FALSE)
dat[,Set:='Testing']
dat[trainingRows,Set:='Training']

model <- glm(Deaths ~ (AttainedAge + Duration + Gender + SmokerStatus)^2,
             data=dat[Set=='Training'],offset=log(ExpectedDeathQX2015VBTbyPolicy),family='poisson')

dat[,predDeaths:=predict(model,newdata=dat,type='response')]
dat[,modelQX:=predDeaths/Exposure]

#Duplicate prior plots
plotMeSimple <- dat[,.(Deaths=sum(Deaths),EDmodel=sum(predDeaths),
                       AEcount=sum(Deaths)/sum(predDeaths),
                       Exposure=sum(Exposure)),
                    by=.(AttainedAge,Set)]
plotMeComplex <- dat[,.(Deaths=sum(Deaths),EDmodel=sum(predDeaths),
                       AEcount=sum(Deaths)/sum(predDeaths),
                       Exposure=sum(Exposure)),
                    by=.(Duration,Gender,SmokerStatus,Set)]
ggplot(data=plotMeSimple,aes(x=AttainedAge,y=AEcount,size=Exposure)) +
  geom_point()+ geom_hline(yintercept=1,color='red',linetype='dashed')+
  coord_cartesian(ylim=c(0,2))
```



```r
ggplot(data=plotMeComplex,aes(x=Duration,y=AEcount,size=Exposure,color=SmokerStatus)) +
  geom_point() +
  facet_grid(Gender~Set)+
  coord_cartesian(ylim=c(0,2)) +
  geom_hline(yintercept=1,color='red',linetype='dashed')
```



```
dat[,.(AECount=sum(Deaths)/sum(predDeaths)),by=Set]
```

```
##          Set   AECount
## 1: Training 1.0000000
## 2:  Testing 0.9981906
```