Check fitting methods against chosen parameters

Adam Rich

July 26, 2018

Prep Stuff

These objects were created at the same time as the smaller problem datasets. Also load required packages.

```r
(load('c:/home/git/other/ratemaking-capstone/data/created-data-full.RData'))
```

```
## [1] "losses"      "policies"    "disciplines" "inflation"   "val_date"
## [6] "states"      "odf"
```

```r
source('c:/home/git/other/ratemaking-capstone/R/resources.R')
require(tidyverse)
```

```
## Loading required package: tidyverse

## -- Attaching packages ------------------------------------------------------------------ tidyverse 1

## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## -- Conflicts --------------------------------------------------------------------- tidyverse_conflic
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
require(actuar)
```

```
## Loading required package: actuar

##
## Attaching package: 'actuar'

## The following object is masked from 'package:grDevices':
##
##     cm
```

```r
require(MASS)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
require(fitdistrplus)
```

```
## Loading required package: fitdistrplus

## Loading required package: survival
```

```r
require(tree)
```

```
## Loading required package: tree
```

Purpose

The purpose of this R file is to try out the fitting methods I expect will work and show that they provide something very close to the pre-selected frequency and severity models.

Data Checks

Make sure that column types are as expected.

```r
str(policies)
```

```
## 'data.frame':    100000 obs. of  22 variables:
##  $ index                 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ policy_number         : chr  "C1AE00092766" "C1AE00783351" "C1AE00936879" "C1AE00037943" ...
##  $ policy_year           : int  2011 2008 2013 2008 2011 2014 2009 2013 2016 2016 ...
##  $ duration_months       : num  12 12 12 12 12 12 12 12 12 12 ...
##  $ policy_month          : int  12 2 10 2 1 11 1 8 9 2 ...
##  $ inception             : chr  "201112" "200802" "201310" "200802" ...
##  $ expiration            : chr  "201212" "200902" "201410" "200902" ...
##  $ revenue_bucket        : int  3 6 1 1 2 4 1 6 1 5 ...
##  $ revenue               : num  379061 3771609 87795 59671 183667 ...
##  $ state                 : chr  "South Carolina" "California" "New Jersey" "Kentucky" ...
##  $ state_group           : chr  "Low" "High" "Mid" "Low" ...
##  $ state_relativity      : num  1 1.5 1.25 1 1.5 1 1.25 3.5 1 1.25 ...
##  $ discipline            : chr  "Landscape Architecture" "Structural Engineer" "Civil Engineer" "Stru
##  $ discipline_relativity: int  1 6 3 6 6 4 1 1 4 2 ...
##  $ discipline_group      : chr  "d1" "d6" "d3" "d6" ...
##  $ revenue_frequency     : num  0.01895 0.18858 0.00439 0.00298 0.00918 ...
##  $ expected_frequency    : num  0.019 1.6972 0.0165 0.0179 0.0827 ...
##  $ claim_count           : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ year_started          : int  2004 1989 1997 1990 2005 2004 2009 2010 2009 2001 ...
##  $ employee_count        : num  3 29 1 1 3 20 1 28 1 47 ...
##  $ use_written_contracts: chr  "N" "Y" "N" "N" ...
##  $ five_year_claims      : num  0 5 0 0 0 0 0 4 0 2 ...
```

Pre-chosen Relativites and Parameters

The base frequency is 0.05 claims per million in revenue.

State relativities are

- Low = 1.0
- Med = 1.25
- High 1.5
- For companies with revenue over $4m, override with 3.5

Discipline relativities are supposed to be:

```r
disciplines[, c('Discipline', 'Relativity')]
```

```
##                Discipline Relativity
## 1    Structural Engineer          6
```

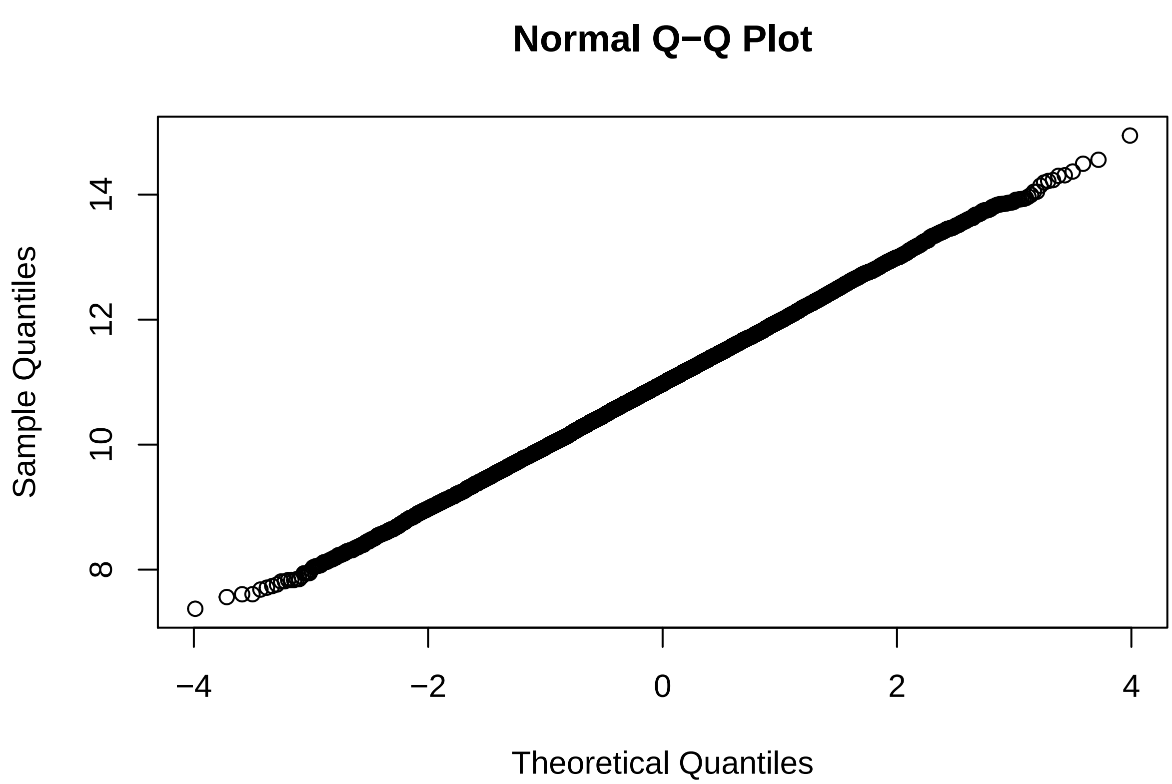
```
## 2               Architect          4
## 3          Civil Engineer          3
## 4 Mechanical Engineering          2
## 5                Surveyor          2
## 6 Landscape Architecture          1
```

Severity Curve

I'll start with the severity curve since it is probably easier than the frequency. I know that the claims come from a lognormal distribution with the same parameters, regardless of any policy characteristics.

An effective way to know whether data is from a lognormal distribution is to do a qqnorm plot on the log values.

```
qqnorm(log(losses$claim_ultimate))
```



The moments of the sample data are

```
mean_loss <- mean(losses$claim_ultimate)
sd_loss <- sd(losses$claim_ultimate)

mean_loss
```

```
## [1] 97003.42
```

```
sd_loss
```

```
## [1] 125479
```

Test a Gamma

A gamma distribution with the same moments would have these parameters (k is shape, theta is scale):

```
k <- (mean_loss / sd_loss)^2
theta <- mean_loss / k

k * theta
```

```
## [1] 97003.42
```

```
sqrt(k) * theta
```

```
## [1] 125479
```

```
losses_gamma <- rgamma(
  n = nrow(losses),
  shape = k,
  scale = theta
)

mean(losses_gamma)
```

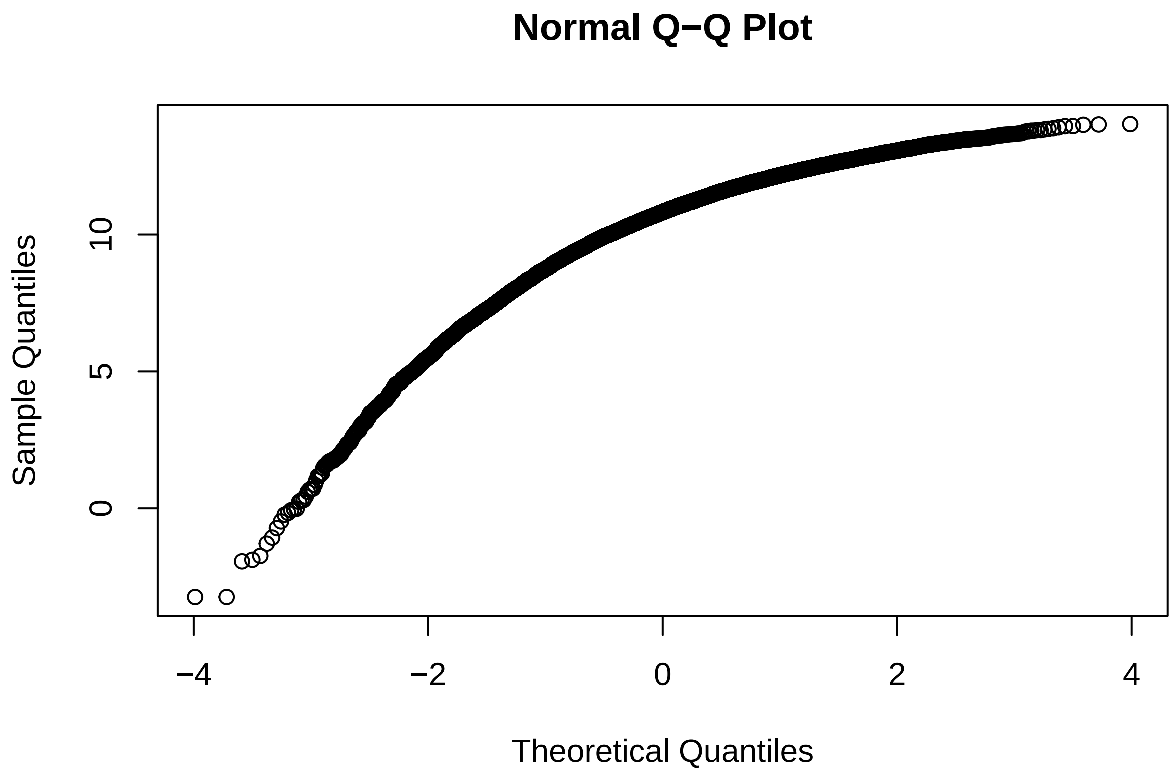
```
## [1] 97388.05
```

```
sd(losses_gamma)
```

```
## [1] 126815.2
```

The `qqnorm` plot for this just doesn't fit a straight line.

```
qqnorm(log(losses_gamma))
```



Severity Parameters

The actual lognormal parameters used are

```
(p <- unique(losses[, c('sev_mu', 'sev_sigma')]))
```

```
##     sev_mu sev_sigma
## 1 10.96163         1
```

```
mu_actual <- p[, 1]
sigma_actual <- p[, 2]
```

Using the `fitdistrplus` package, we get these fitted parameters:

```
(b <- fitdistrplus::fitdist(losses$claim_ultimate, 'lnorm'))
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate  Std. Error
## meanlog 10.975047 0.008224922
## sdlog    1.007679 0.005815873
```

```
mu_fitted <- b$estimate[1] %>% unname
sigma_fitted <- b$estimate[2] %>% unname
```

A function for lognormal limited expected value is

```r
levlnorm <- function(x, mu, sigma) {
  exp(mu + sigma^2/2) *
    pnorm((log(x) - mu - sigma^2) / sigma) +
    x * (1 - plnorm(x, mu, sigma))
}
```

The expected values, unlimited and then limited at $1m, $5m, and $10m are:

```r
actual_U <- exp(mu_actual + sigma_actual^2/2)
actual_1 <- levlnorm(1e6, mu_actual, sigma_actual)
actual_2 <- levlnorm(2e6, mu_actual, sigma_actual)
actual_5 <- levlnorm(3e6, mu_actual, sigma_actual)
```

Check using "brute force"

```r
integrate(
  lower = 1,
  upper = 100e6,
  f = function(t) {t * dlnorm(t, mu_actual, sigma_actual)}
)
```

```
## 95000 with absolute error < 6.9
```

```r
integrate(
  lower = 1,
  upper = 100e6,
  f = function(t) {pmin(t, 1e6) * dlnorm(t, mu_actual, sigma_actual)}
)
```

```
## 94131.01 with absolute error < 11
```

```r
integrate(
  lower = 1,
  upper = 100e6,
  f = function(t) {pmin(t, 2e6) * dlnorm(t, mu_actual, sigma_actual)}
)
```

```
## 94873.54 with absolute error < 7.8
```

```r
integrate(
  lower = 1,
  upper = 100e6,
  f = function(t) {pmin(t, 5e6) * dlnorm(t, mu_actual, sigma_actual)}
)
```

```
## 94994.83 with absolute error < 7.9
```

The severities are so low in this distribution that the limits of $1m, $2m, and $5m are basically the same price.

Frequency Prep

There is an engineered break at $4m in revenue. Below that, revenue, state group and discipline are all predictive. Above that, state group is no longer predictive.

One thing that is really important to check before doing a GLM on frequency is whether a Poisson error is appropriate. This check is done by comparing the mean and variance of the observations.

```r
mean(policies$claim_count)
```

```
## [1] 0.1501
```

```r
var(policies$claim_count)
```

```
## [1] 0.4131541
```

```r
var(policies$claim_count) / mean(policies$claim_count)
```

```
## [1] 2.752526
```

They are *not* close so Poisson is not appropriate. However, a negative binomial distribution has variance greater than the mean.

The problem with the glm.nb is that it assumes that theta is a constant. But, I know that var(X) = o.d.f * E[X] where o.d.f. is a constant. So, I need a family with a log link that allows for this linear relationship between variance and mean. Quasipoisson does this.

Create some new columns in the dataset Re-base the factors, too:

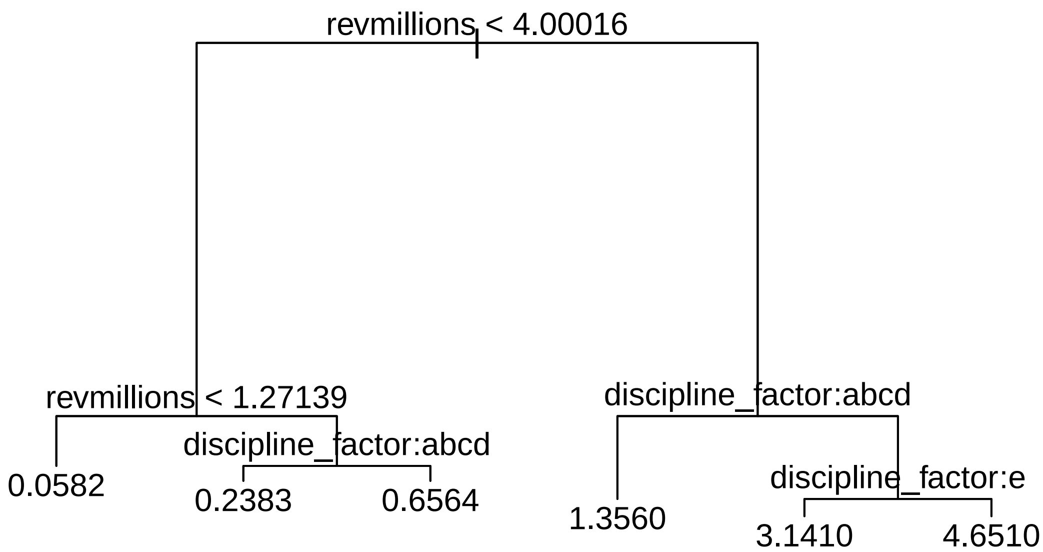
- Base state = Low group (lowest relativity)
- Base discipline = Landscape Architecture (lowest relativity)

```r
policies$revmillions <- policies$revenue / 1e6

dlevels <- c(
  "Landscape Architecture", "Surveyor", "Mechanical Engineering",
  "Civil Engineer", "Architect", "Structural Engineer")

slevels <- c("Low", "High", "Mid")

policies$state_group_factor <- factor(
  x = policies$state_group,
  levels = slevels
)

policies$discipline_factor <- factor(
  x = policies$discipline,
  levels = dlevels
)

policies$use_written_contracts_factor <- as.factor(policies$use_written_contracts)
```

Engineered Break

Not all revenue bands have the same model!

```r
tpol <- tree(
  claim_count ~ revmillions + discipline_factor +
    state_group_factor + year_started +
    employee_count + use_written_contracts_factor,
  policies
)

plot(tpol)
text(tpol)
```




```r
pol_low <- policies[policies$revenue < 4e6, ]
pol_high <- policies[policies$revenue >= 4e6, ]
nrow(pol_low)
```

```
## [1] 98099
```

```r
nrow(pol_high)
```

```
## [1] 1901
```

Frequency models

I expect this to be the correct frequency model!

```r
fit_low <- glm(
  claim_count ~ offset(log(revmillions)) + state_group_factor + discipline_factor,
  data = pol_low,
  family = quasipoisson()
)

fit_high <- glm(
  claim_count ~ offset(log(revmillions)) + discipline_factor,
  data = pol_high,
  family = quasipoisson()
)

summary(fit_low)
```

```
##
## Call:
## glm(formula = claim_count ~ offset(log(revmillions)) + state_group_factor +
##     discipline_factor, family = quasipoisson(), data = pol_low)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9468  -0.4077  -0.2460  -0.1598   7.5442
##
## Coefficients:
##                                         Estimate Std. Error t value
## (Intercept)                             -3.02542    0.05285 -57.245
## state_group_factorHigh                   0.43496    0.02958  14.707
## state_group_factorMid                    0.23253    0.02948   7.888
## discipline_factorSurveyor                0.71104    0.06675  10.653
## discipline_factorMechanical Engineering  0.76628    0.07857   9.753
## discipline_factorCivil Engineer          1.08148    0.07130  15.168
## discipline_factorArchitect               1.39067    0.05428  25.618
## discipline_factorStructural Engineer     1.84700    0.05447  33.908
##                                         Pr(>|t|)
## (Intercept)                              < 2e-16 ***
## state_group_factorHigh                   < 2e-16 ***
## state_group_factorMid                   3.09e-15 ***
## discipline_factorSurveyor                < 2e-16 ***
## discipline_factorMechanical Engineering  < 2e-16 ***
## discipline_factorCivil Engineer          < 2e-16 ***
## discipline_factorArchitect               < 2e-16 ***
## discipline_factorStructural Engineer     < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.50075)
##
##     Null deviance: 42273  on 98098  degrees of freedom
## Residual deviance: 38905  on 98091  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 6
```

```r
summary(fit_high)
```

```
##
## Call:
## glm(formula = claim_count ~ offset(log(revmillions)) + discipline_factor,
##     family = quasipoisson(), data = pol_high)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2031  -1.2883  -0.2337   0.6681   4.2661
##
## Coefficients:
##                                         Estimate Std. Error t value
## (Intercept)                             -1.67941    0.07076 -23.732
## discipline_factorSurveyor                0.62295    0.09423   6.611
## discipline_factorMechanical Engineering  0.72987    0.11342   6.435
```

```
## discipline_factorCivil Engineer          0.91857    0.10536   8.718
## discipline_factorArchitect               1.32328    0.07619  17.368
## discipline_factorStructural Engineer     1.71479    0.07698  22.275
##                                        Pr(>|t|)
## (Intercept)                             < 2e-16 ***
## discipline_factorSurveyor              4.94e-11 ***
## discipline_factorMechanical Engineering 1.56e-10 ***
## discipline_factorCivil Engineer         < 2e-16 ***
## discipline_factorArchitect              < 2e-16 ***
## discipline_factorStructural Engineer    < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.602547)
##
##     Null deviance: 4561.3  on 1900  degrees of freedom
## Residual deviance: 3200.5  on 1895  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```
copy.model(fit_low)
```

```
##                                                                        variable
## (Intercept)                                                         (Intercept)
## state_group_factorHigh                                   state_group_factorHigh
## state_group_factorMid                                     state_group_factorMid
## discipline_factorSurveyor                             discipline_factorSurveyor
## discipline_factorMechanical Engineering discipline_factorMechanical Engineering
## discipline_factorCivil Engineer                 discipline_factorCivil Engineer
## discipline_factorArchitect                           discipline_factorArchitect
## discipline_factorStructural Engineer       discipline_factorStructural Engineer
##                                        coefficient
## (Intercept)                             -3.0254152
## state_group_factorHigh                   0.4349623
## state_group_factorMid                    0.2325301
## discipline_factorSurveyor                0.7110352
## discipline_factorMechanical Engineering   0.7662762
## discipline_factorCivil Engineer          1.0814783
## discipline_factorArchitect               1.3906705
## discipline_factorStructural Engineer     1.8470023
```

```
copy.model(fit_high)
```

```
##                                                                        variable
## (Intercept)                                                         (Intercept)
## discipline_factorSurveyor                             discipline_factorSurveyor
## discipline_factorMechanical Engineering discipline_factorMechanical Engineering
## discipline_factorCivil Engineer                 discipline_factorCivil Engineer
## discipline_factorArchitect                           discipline_factorArchitect
## discipline_factorStructural Engineer       discipline_factorStructural Engineer
##                                        coefficient
## (Intercept)                             -1.6794087
## discipline_factorSurveyor                0.6229469
## discipline_factorMechanical Engineering   0.7298676
```

```
## discipline_factorCivil Engineer           0.9185694
## discipline_factorArchitect                1.3232756
## discipline_factorStructural Engineer      1.7147894
```