
R in Action: 3 Case Studies

Case #1. Predictive modeling
of claim counts



Traditional count distributions can fit poorly

- Policyholder claim counts with a high frequency of zeroes are problematic for Poisson, Negative Binomial
- Alternative distributions can fit better
 - Tweedie (usually for aggregate claim amounts, can also be used for counts)
 - Meyers, *Actuarial Review* article
<http://www.casact.org/newsletter/index.cfm?fa=viewart&id=5756>
 - Smyth & Jørgensen, 2002 *Astin* article
<http://www.casact.org/library/astin/vol32no1/143.pdf>
 - Zero-inflated distributions
 - Flynn & Francis, *eForum*
http://www.casact.org/pubs/forum/09wforum/flynn_francis.pdf
- We will use R to simulate count data and compare the distributions' fits



Simulate counts for 10000 heterogeneous customers, fit Poissons, negative binomials

- Half male customers, half female with random and different claim propensities
- Tabulate frequency of counts
- Fit Poisson
 - density values at observed counts
 - overall average propensity
- Use package 'MASS' for negative binomial
 - find parameters
 - find density values

```
> num.claims.male<-rpois(5000,lambda=runif(5000,3,5))
> num.claims.female=rpois(5000,lambda=runif(5000,7,9))
> # concatenate into one vector for all customers
> num.claims.all=c(num.claims.male,num.claims.female)
```

```
> count.all = table(num.claims.all)
```

```
 0   1   2   3   4   5   6   7   8   9  10  11 ... 20 21
110 420 821 1107 1186 1238 1112 968 807 722 545 380 ... 1 1
```

```
> poisson.density.all =
      dpois(0:21,
            lambda=mean(num.claims.all))
```

Don't forget: ?fitdistr, \$dnbinom

```
> library(MASS)
> NB.all=fitdistr(num.claims.all,"negative binomial")
> NB.density.all=dnbinom(0:21,mu=NB.all$estimate[2],
                          size=NB.all$estimate[1])
```



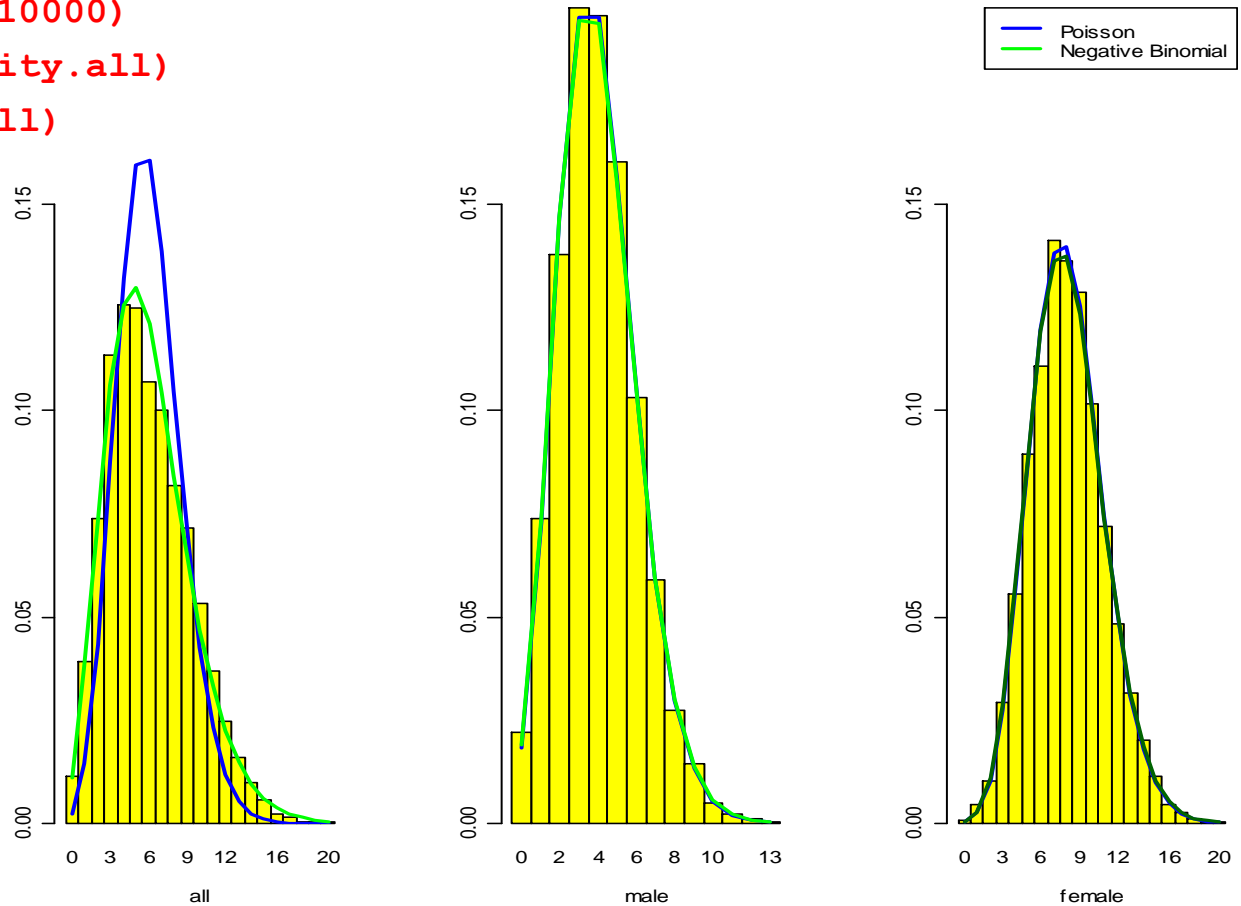
Simulated count densities, fitted distributions

leftmost graph

We didn't see 'barplot' yesterday.
I omitted options that "pretty up" the graph.

```
> barplot(count.all/10000)
> lines(poisson.density.all)
> lines(NB.density.all)
```

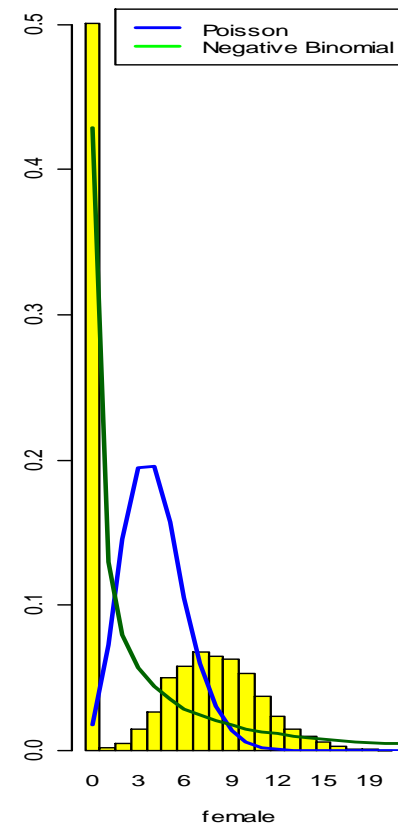
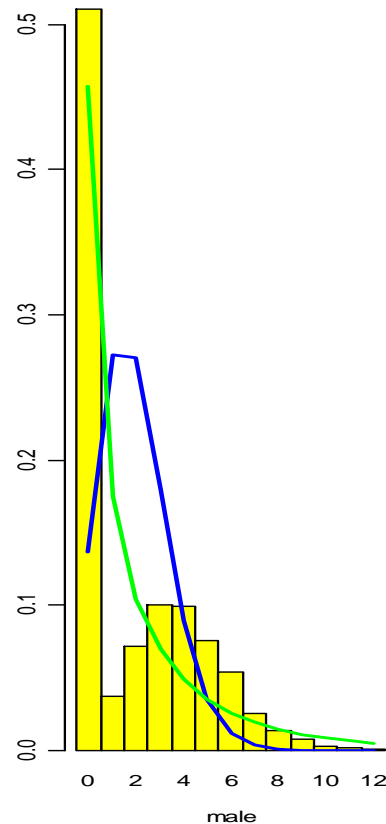
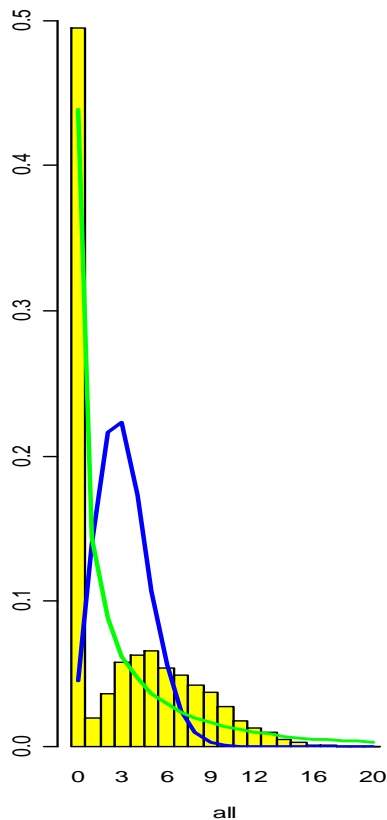
- NB looks better than Poisson for "all"
- Fits similar for subpopulations



When number of claims are “zero-inflated” Poisson, NB fits are poor, even for subpopulations

- About ½ of policyholders have no claims
- Those that do, claim count distribution is as above

```
> ZIclaims.all=num.claims.all*round(runif(10000))  
> ZIcount.all=table(ZIclaims.all)  
 0   1   2   3   4   5   6   7   8   9  10  11 ... 17 18  
5093 206 428 532 594 621 530 483 411 354 256 201 ...  5  2
```



Use special R packages to try Tweedie, Zero-Inflated Negative Binomial (ZINB) fits

- tweedie package can
 - find fitted parameters
 - calculate density, random values, quantiles, etc

```
> library(tweedie)
> tw.fit.all = tweedie.profile(ZIclaims.all~1,
                             fit.glm=TRUE,
                             do.smooth=FALSE)
> tw.density.all = dtweedie(0:18,
                             p=tw.fit.all$p.max,
                             mu=exp(tw.fit.all$glm.obj$coefficients),
                             phi=tw.fit.all$phi.max)
```

```
remember: ?tweedie.profile,
          ?dtweedie
```

tweedie density is a builtin function

- Political Science Computational Laboratory, Stanford University
 - zero-inflated models
 - GLM goodness-of-fit measures
 - other models and functions

```
> library(pscl)
> ZINB.fit.all = zeroinfl(ZIclaims.all~1|1,
                          dist="negbin", link="logit")
> p = 1/(1+exp(ZINB.fit.all$coef$zero))
> mu = exp(summary(ZINB.fit.all)$coef$count[1])
> size = exp(summary(ZINB.fit.all)$coef$count[2])
> ZINB.density.all = c(p, rep(0,18)) +
  (1-p) * dnbinom(0:18, mu=mu, size=size)
```

ZINB density must be constructed from p, NB density:

```
.5    0    0    0    0    0    ...    0    0
.01   .02  .04  .05  .06  .07  ...  .001  .001
```

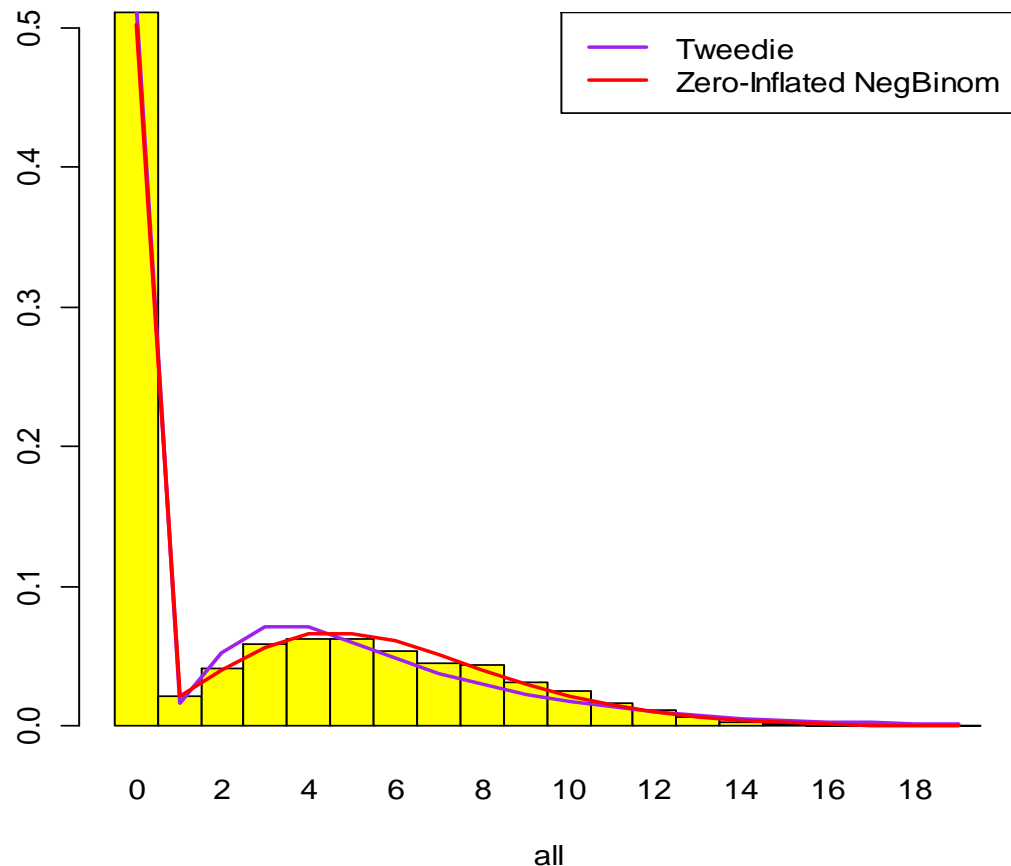
hypothetical



Tweedie, ZINB are much better than Poisson, NB for fitting zero-inflated counts

```
> barplot(ZIcount.all/10000)
> lines(tweedie.density.all)
> lines(ZINB.density.all)
```

- ZINB looks to be a slightly better fit than the Tweedie
- Runtime for fits
 - Tweedie: 37 min
 - ZINB: 6 sec



Authors and maintainers of R Packages are contemporaries of the technology

- tweedie authored by Peter Dunn
 - University of Wisconsin-Milwaukee
 - Department of Biological Sciences



- pscl authored by Simon Jackman
 - Stanford University
 - Department of Political Science



- zeroinfl authored by Achim Zeileis
 - University of Vienna
 - Department of Statistics & Mathematics



R in Action: 3 Case Studies

Case #3. Stochastic Reserving



We will look at some Schedule P commercial auto data

- This is the file ABCSchP.csv in Excel

	A	B	C	D	E	F	G	H	IncurLoss_D	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	rec	COM	LOB	AccidentYear	De	DevelopmentLag	Inc	Inc	IncurLoss_C	Inc	Inc	Inc	Inc	Inc	Inc	Inc	Inc	Inc	Cu	Cu	CumPaidLoss_C	Cu	Cu	
2	1	XXX	CML AUTO	2000	#		1	#	200589	#	0	0	0	#	#	#	0	0	#	#	31059	#	0	
3	2	XXX	CML AUTO	2000	#		2	#	167911	#	0	0	0	#	#	#	0	0	#	#	73059	#	0	
4	3	XXX	CML AUTO	2000	#		3	#	152533	#	0	0	0	#	#	#	0	0	#	#	94329	#	0	
5	4	XXX	CML AUTO	2000	#		4	#	164858	#	4	0	0	#	#	#	0	0	#	#	114317	#	4	
6	5	XXX	CML AUTO	2000	#		5	#	168227	#	4	0	0	#	#	#	0	0	#	#	138342	#	4	
7	6	XXX	CML AUTO	2000	#		6	#	170308	#	4	0	0	#	#	#	0	0	#	#	155079	#	4	
8	7	XXX	CML AUTO	2000	#		7	#	167270	#	4	0	0	#	#	#	0	0	#	#	163762	#	4	
9	8	XXX	CML AUTO	2000	#		8	#	160686	#	4	0	0	#	#	#	0	0	#	#	160426	#	4	
10	9	XXX	CML AUTO	2000	#		9	#	161323	#	4	0	0	#	#	#	0	0	#	#	161220	#	4	
11	10	XXX	CML AUTO	2000	#		10	#	161354	#	4	0	0	#	#	#	0	0	#	#	161312	#	4	
12	11	XXX	CML AUTO														#	0	0	#	#	30156	#	4
13	12	XXX	CML AUTO														#	0	0	#	#	67899	#	59
14	13	XXX	CML AUTO														#	0	0	#	#	101455	#	59
15	14	XXX	CML AUTO														#	0	0	#	#	122666	#	59
16	15	XXX	CML AUTO														#	0	0	#	#	134191	#	64
17	16	XXX	CML AUTO														#	0	0	#	#	139464	#	64
18	17	XXX	CML AUTO	2001	#		7	#	147190	#	64	0	0	#	#	#	0	0	#	#	141765	#	64	
19	18	XXX	CML AUTO	2001	#		8	#	146334	#	64	0	0	#	#	#	0	0	#	#	143171	#	64	
20	19	XXX	CML AUTO	2001	#		9	#	145389	#	64	0	0	#	#	#	0	0	#	#	144236	#	64	
21	20	XXX	CML AUTO	2002	#		1	#	134110	#	#	0	0	86	#	#	0	0	94	#	27162	#	12	
22	21	XXX	CML AUTO	2002	#		2	#	98791	#	85	0	0	98	#	#	0	0	#	#	44220	#	49	
23	22	XXX	CML AUTO	2002	#		3	#	105295	#	#	0	0	#	#	#	0	0	#	#	67259	#	27	
24	23	XXX	CML AUTO	2002	#		4	#	104409	#	78	0	0	#	#	#	0	0	#	#	78190	#	36	
25	24	XXX	CML AUTO	2002	#		5	#	98080	#	62	0	0	#	#	#	0	0	#	#	83635	#	62	
26	25	XXX	CML AUTO	2002	#		6	#	94581	#	62	0	0	#	#	#	0	0	#	#	88196	#	62	
27	26	XXX	CML AUTO	2002	#		7	#	94936	#	62	0	0	#	#	#	0	0	#	#	90206	#	62	

- Read ABC's data into an R 'data frame'

```
> ABCdf = read.csv("ABCSchP.csv")
```



Can use Markus Gesmann's ChainLadder package to run Mack's method

- load the package
- 'as.triangle' function reshapes columns from data frame into a "triangle" object
 - Just name desired columns
 - ?as.triangle for help
- Type object's name to reveal its contents

```
> library(ChainLadder)
```

```
> CAtri = as.triangle(ABCdf,  
                      origin="AccidentYear",  
                      dev="DevelopmentLag",  
                      value="CumPaidLoss_C")
```

```
> CAtri
```

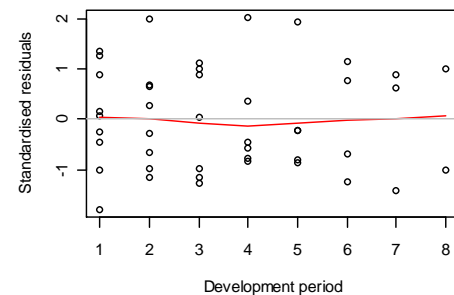
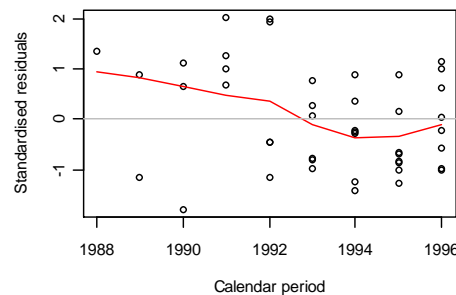
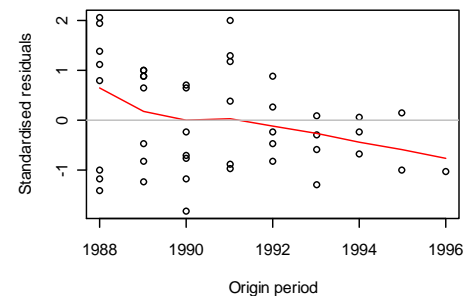
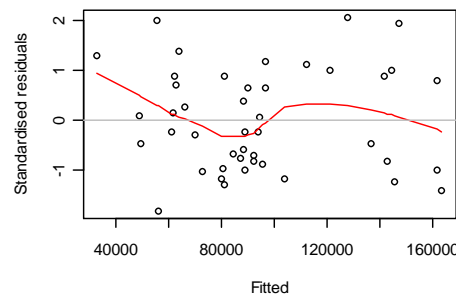
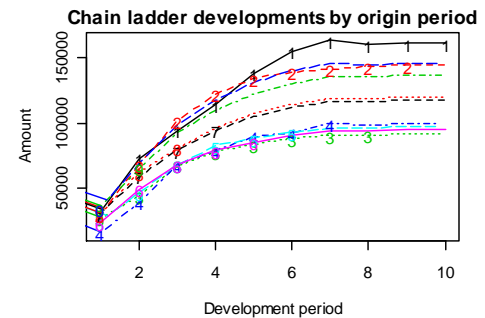
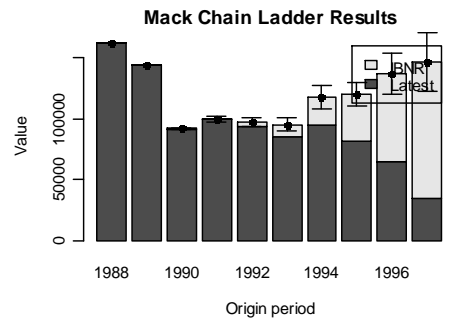
	1	2	3	4	5	6	7	8	9	10
2000	31059	73059	94329	114317	138342	155079	163762	160426	161220	161312
2001	30156	67899	101455	122666	134191	139464	141765	143171	144236	NA
2002	27162	44220	67259	78190	83635	88196	90206	91087	NA	NA
2003	15901	39065	67717	79028	89839	92536	99212	NA	NA	NA
2004	24083	46648	68050	82408	88002	92825	NA	NA	NA	NA
2005	23798	49339	68105	78981	85374	NA	NA	NA	NA	NA
2006	29725	59466	79491	94692	NA	NA	NA	NA	NA	NA
2007	29860	62377	81129	NA	NA	NA	NA	NA	NA	NA
2008	35283	65141	NA	NA	NA	NA	NA	NA	NA	NA
2009	34004	NA	NA	NA	NA	NA	NA	NA	NA	NA



With data in triangle form, run Mack Method

- 'MackChainLadder' carries out calculations
- est.sigma="Mack" implements Mack's heuristic at the tip of the triangle
- There are many more options available
 - ?MackChainLadder for help on the function
- 'plot' function displays the CAcl object's error-banded AY ultimates, development curves, and residuals

```
> CAcl = MackChainLadder(CAtri, est.sigma="Mack")  
> plot(CAcl)
```



Type object's name to see object content that package author thinks most important

> CAcl

	Latest	Dev.To.Date	Ultimate	IBNR	Mack.S.E	CV (IBNR)
1988	161,312	1.000	161,312	0.0	0	NaN
1989	144,236	0.999	144,318	82.3	37	0.450
1990	91,087	0.993	91,697	610.0	237	0.389
1991	99,212	0.996	99,612	399.7	2,362	5.909
1992	92,825	0.956	97,056	4,231.2	3,777	0.893
1993	85,374	0.899	94,964	9,590.2	5,611	0.585
1994	94,692	0.806	117,423	22,730.8	9,121	0.401
1995	81,129	0.678	119,730	38,600.6	9,557	0.248
1996	65,141	0.477	136,466	71,324.8	16,638	0.233
1997	34,004	0.232	146,267	112,263.0	24,186	0.215

will use later

ByOrigin section

	Totals
Latest:	949,012.00
Ultimate:	1,208,844.71
IBNR:	259,832.71
Mack S.E.:	37,964.72
CV (IBNR):	0.15

Totals section

summary function
best way to access
chain ladder
object's values

> CA.mean = summary(CAcl)\$Totals["IBNR",1]

[1] 259832.7

> CA.se = summary(CAcl)\$Totals["Mack S.E.",1]

[1] 37964.72

For gory details of
any object, use **str**

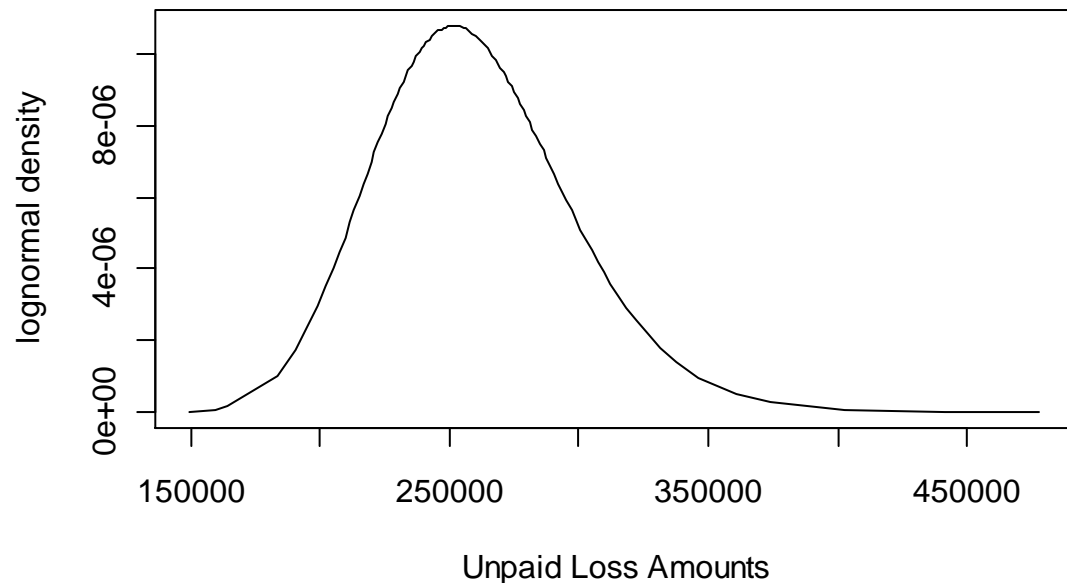


Per Mack's recommendation, assume outstanding loss \sim lognormal

- 'lnormParms' function solves for lognormal's μ , σ parameters
 - see appendix
- p are the percentiles where we want density plotted
 - qlnorm = quantiles of the lognormal, ie., the x-axis values
- dlnorm = density values of the lognormal

```
> CA.parms = lnormParms(CA.mean, CA.se)
> p = c(.0001, .0005, .001,
        seq(.01, .99, by=.01),
        .995, .999, .9999, .99999)
> x = qlnorm(p, CA.parms$mu, CA.parms$sigma)
> plot(x, dlnorm(x, CA.parms$mu, CA.parms$sigma))
```

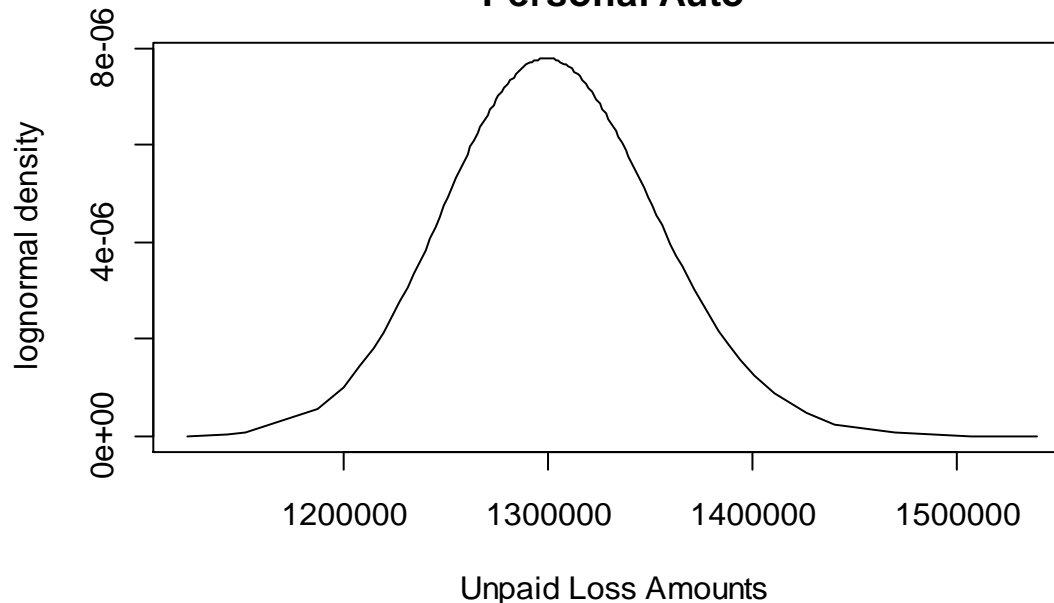
**Estimated Distribution Using Mack Method on Paid Data
Commercial Auto**



Run the same steps on Personal Auto

```
> PAtri = as.triangle(ABCdf, origin="AccidentYear",  
                    dev="DevelopmentLag", value="CumPaidLoss_B")  
> PAcl = MackChainLadder(PAtri, est.sigma="Mack")  
> PA.parms = lnormParms(summary(PAcl)$Totals["IBNR",1],  
                       summary(PAcl)$Totals["Mack S.E.",1])  
> x = qlnorm(p, PA.parms$mu, PA.parms$sigma)  
> plot(x, dlnorm(x, PA.parms$mu, PA.parms$sigma))
```

**Estimated Distribution Using Mack Method on Paid Data
Personal Auto**



Brief guess at correlation between commercial and personal auto

- Between first column of the paid loss triangles

```
> cor(CAtri[,1],PAtri[,1])
```

```
[1] 0.5562267
```

Looks high

- Between first column of the incurred loss triangles

```
> cor(CAIncdtri[,1],PAIncdtri[,1])
```

```
[1] -0.477677
```

Looks low

code not shown above

- Between estimated ultimates

```
> cor(summary(CAcl)$ByOrigin[, "Ultimate"],  
       summary(PAcl)$ByOrigin[, "Ultimate"])
```

```
[1] 0.2646838
```

Looks high

code to get premium

- Between ultimate loss ratios

```
> CANep=ABCdf[ABCdf$DevelopmentLag==1, "EarnedPremNet_C"]
```

```
> PANep=ABCdf[ABCdf$DevelopmentLag==1, "EarnedPremNet_B"]
```

```
> cor(summary(CAcl)$ByOrigin[, "Ultimate"]/CANep,  
       summary(PAcl)$ByOrigin[, "Ultimate"]/PANep)
```

```
[1] 0.1018575
```

Suppose we select 10% correlation



t-copula for estimating aggregate distribution

```
> library(copula)
> bivar = mvdc(tCopula(param=0.10, dim=2, df=2),
  margins=c("lnorm", "lnorm"),
  paramMargins=list(list(meanlog=CA.parms$mu, sdlog=CA.parms$sigma),
    list(meanlog=PA.parms$mu, sdlog=PA.parms$sigma)))
```

10% correlation → 2 lines

'mvdc' function creates a "multivariate distribution object", here based on a t copula

```
> samp.bivar = rmvdc(bivar, 100000)
> cor(samp.bivar)
```

100,000 random samples of 2 dependent r.v's simulating each line's O/S loss

```
      [,1]      [,2]
[1,] 1.0000000 0.1025357
[2,] 0.1025357 1.0000000
```

```
> hist(rowSums(samp.bivar))
> VAR=quantile(rowSums(samp.bivar), .995)
```

'rowSums' gives 100000 samples of agg O/S for portfolio

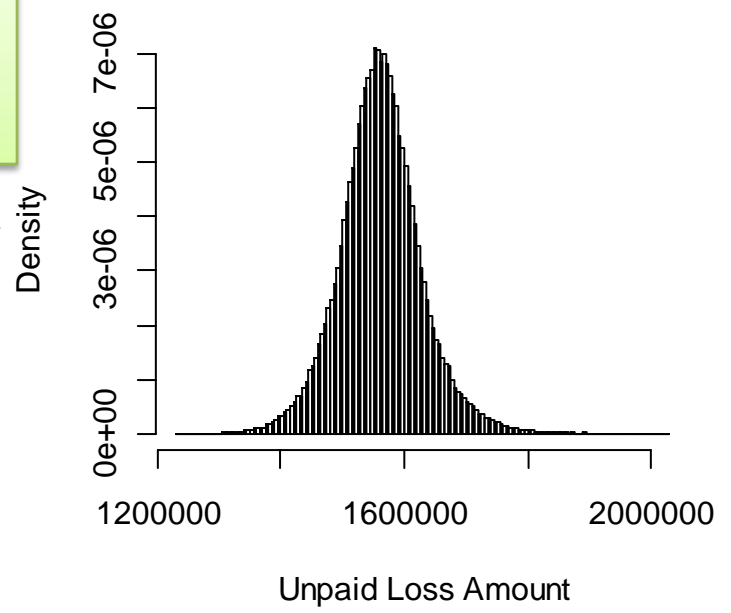
```
 99.5%
1774586
```

```
> quantile(samp.bivar[,1], .995)
+ quantile(samp.bivar[,2], .995)
- VAR
```

```
 99.5%
39360.60
```

Diversification benefit @ 99.5% VAR ≈ \$39mm

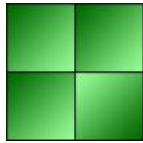
Portfolio Distribution



Appendix

```
InnormParms <- function(mean,std) {  
  if (any(mean<=0) | any(std<=0))  
    stop("Negative mean or std")  
  sigma2 <- log(1+(std/mean)^2)  
  mu      <- log(mean)-.5*sigma2  
  list(mu=mu, sigma=sqrt(sigma2))  
}
```





Trinostics LLC is in the business of collaboration and education in the design and construction of transparently valuable actuarial models

Daniel Murphy, FCAS, MAAA
dmurphy@trinostics.com

