Reproducing Kuang and Nielsen Generalized log normal Chain-Ladder using the apc package

23 November 2019

Di Kuang Lloyds of London
Bent Nielsen Department of Economics, University of Oxford
& Nuffield College
& Institute for Economic Modelling
bent.nielsen@nuffield.ox.ac.uk
http://users.ox.ac.uk/~nuff0078

Contents

1 Introduction

The purpose of this vignette is to use the `apc` package version 1.3.5 to reproduce some the result in Kuang and Nielsen (2020): *Generalized Log-Normal Chain-Ladder*. This adopts the theory presented in Harnau and Nielsen (2018), from an over-dispersed Poisson model to a log-normal model. There is also a vignette available for that paper. The `apc` package builds on the identification analysis and the forecast theory in Kuang, Nielsen and Nielsen (2008a,b), the development of deviance analysis for general data arrays in Nielsen (2014). The package is discussed in Nielsen (2015).

2 Table 1.1: The data

The data set is a reserving triangle from the XL group. It represents US casualty, gross paid and reported loss and allocated loss adjustment expense in 1000 USD.

The data are available in the `apc` package. They can be called with the following command. Note that the output is wide and therefore truncated to fit the page width.

```
> library(apc)
> data <- data.loss.XL()
> data$response[,1:8]
```

```
     1997  1998  1999  2000  2001  2002  2003  2004
1997 2185 13908 44704 56445 67313 62830 72619 42511
1998 3004 17478 49564 55090 75119 66759 76212 62311
1999 5690 28971 55352 63830 71528 73549 72159 37275
2000 9035 29666 47086 41100 58533 80538 70521 40192
2001 7924 38961 41069 64760 64069 61135 62109 52702
2002 7285 25867 44375 58199 61245 48661 57238 29667
2003 3017 22966 62909 54143 72216 58050 29522 25245
2004 1752 25338 56419 75381 64677 58121 38339 21342
2005 1181 24571 66321 65515 62151 43727 29785 23981
2006 1706 13203 40759 57844 48205 50461 27801 21222
2007  623 14485 27715 52243 60190 45100 31092 22731
2008  338  6254 24473 32314 35698 25849 30407 15335
2009  255  3842 14086 26177 27713 15087 17085 12520
2010  258  7426 22459 28665 32847 28479 24096    NA
2011 1139 10300 19750 32722 41701 29904    NA    NA
2012  381  5671 34139 33735 33191    NA    NA    NA
2013  605 11242 24025 32777    NA    NA    NA    NA
2014 1091  9970 31410    NA    NA    NA    NA    NA
2015 1221  8374    NA    NA    NA    NA    NA    NA
2016 2458    NA    NA    NA    NA    NA    NA    NA
```

3 Table 4.1: Analysis of variance

The deviance table can be reproduced by the following commands. The first call has the APC model as reference. The second call has the AC model as reference. For an overview of the models, see Nielsen (2014). The output is wide, so only selected columns are shown.

```
> apc.fit.table(data,"log.normal.response")[,c(1,2,6,7)]

     -2logL df.residual F vs.APC prob(>F)
APC 170.003         153      NaN      NaN
AP  243.531         171    3.564    0.000
AC  179.873         171    0.409    0.984
PC  633.432         171   68.736    0.000
Ad  258.570         189    2.230    0.000
Pd  643.892         189   36.340    0.000
Cd  649.142         189   37.368    0.000
A   357.359         190    5.956    0.000
P   644.176         190   35.412    0.000
C   672.392         190   41.099    0.000
t   664.488         207   27.015    0.000
tA  681.993         208   29.072    0.000
tP  664.746         208   26.560    0.000
tC  686.181         208   29.713    0.000
1   690.399         209   29.830    0.000
```

```
> apc.fit.table(data,"log.normal.response","AC")[,c(1,2,6,7)]

    -2logL df.residual F vs.AC prob(>F)
AC 179.873         171     NaN      NaN
Ad 258.570         189   4.319        0
Cd 649.142         189  79.257        0
A  357.359         190  11.955        0
C  672.392         190  84.930        0
t  664.488         207  42.993        0
tA 681.993         208  45.869        0
tC 686.181         208  46.886        0
1  690.399         209  46.670        0
```

Thus, Table 4.1 in the paper is constructed as follows.

```
> table.APC <- apc.fit.table(data,"log.normal.response")
> table.AC  <- apc.fit.table(data,"log.normal.response","AC")
> Table41 <- matrix(NA,nrow=3,ncol=6)
> Table41[1:3,1:2]  <- table.APC[c(1,3,5),1:2]
> Table41[2:3,3:4]  <- table.APC[c(3,5),6:7]
> Table41[3,5:6]    <- table.AC[2,6:7]
```

```
> rownames(Table41) <- c("apc","ac","ad")
> colnames(Table41) <- c("-2logL","df","F_sup,apc","p","F_sup,ac","p")
> Table41
```

```
     -2logL  df F_sup,apc     p F_sup,ac  p
apc 170.003 153        NA    NA       NA NA
ac  179.873 171     0.409 0.984       NA NA
ad  258.570 189     2.230 0.000    4.319  0
```

4 Table 4.2: Estimates

The table of estimates can be reproduced by the following commands. As a default the program gives the double differenced time effects. These are referred to as the canonical parameters. Note that in the apc package α is the age or development year β is the period or calendar year, γ is the cohort or policy year

```
> fit <- apc.fit.model(data,"log.normal.response","AC")
> fit$coefficients.canonical
```

```
                  Estimate Std. Error     t value     Pr(>|t|)
level          7.660055032  0.1377951 55.59016605 0.000000e+00
age slope      2.272100342  0.1335080 17.01846386 5.992216e-65
cohort slope   0.288755900  0.1335080  2.16283663 3.055375e-02
DD_age_1999   -1.339569792  0.2328429 -5.75310613 8.761844e-09
DD_age_2000   -0.696924194  0.2386087 -2.92078261 3.491534e-03
DD_age_2001   -0.146719747  0.2453026 -0.59811748 5.497615e-01
DD_age_2002   -0.264930913  0.2527431 -1.04822205 2.945363e-01
DD_age_2003    0.031598844  0.2609768  0.12107911 9.036284e-01
DD_age_2004   -0.283163142  0.2701148 -1.04830652 2.944974e-01
DD_age_2005    0.127081007  0.2803153  0.45335018 6.502966e-01
DD_age_2006   -0.099202405  0.2917899 -0.33997887 7.338724e-01
DD_age_2007    0.210073941  0.3048208  0.68917200 4.907150e-01
DD_age_2008   -0.052407612  0.3197906 -0.16388105 8.698248e-01
DD_age_2009   -0.018395937  0.3372315 -0.05454988 9.564971e-01
DD_age_2010   -0.294857921  0.3579095 -0.82383376 4.100340e-01
DD_age_2011    0.252083441  0.3829750  0.65822430 5.103940e-01
DD_age_2012    0.709064853  0.4142566  1.71165624 8.696004e-02
DD_age_2013   -1.301108834  0.4548907 -2.86026717 4.232842e-03
DD_age_2014    1.011946941  0.5108825  1.98078211 4.761571e-02
DD_age_2015   -0.499717114  0.5959631 -0.83850344 4.017480e-01
DD_age_2016    0.109628839  0.7552897  0.14514807 8.845940e-01
DD_cohort_1999 -0.125331664  0.2328429 -0.53826711 5.903927e-01
DD_cohort_2000 -0.428405722  0.2386087 -1.79543197 7.258490e-02
DD_cohort_2001  0.414810916  0.2453026  1.69101751 9.083346e-02
DD_cohort_2002 -0.524216258  0.2527431 -2.07410692 3.806938e-02
```

```
DD_cohort_2003  0.175650935  0.2609768  0.67305179 5.009143e-01
DD_cohort_2004  0.189928763  0.2701148  0.70314081 4.819680e-01
DD_cohort_2005  0.003469177  0.2803153  0.01237598 9.901256e-01
DD_cohort_2006 -0.126934898  0.2917899 -0.43502154 6.635468e-01
DD_cohort_2007  0.110410208  0.3048208  0.36221353 7.171925e-01
DD_cohort_2008 -0.450739627  0.3197906 -1.40948386 1.586921e-01
DD_cohort_2009  0.035029472  0.3372315  0.10387366 9.172696e-01
DD_cohort_2010  0.733084362  0.3579095  2.04823952 4.053654e-02
DD_cohort_2011  0.015034268  0.3829750  0.03925653 9.686859e-01
DD_cohort_2012 -0.579238239  0.4142566 -1.39825961 1.620351e-01
DD_cohort_2013  0.410823816  0.4548907  0.90312651 3.664588e-01
DD_cohort_2014  0.059646180  0.5108825  0.11675127 9.070572e-01
DD_cohort_2015 -0.294450287  0.5959631 -0.49407469 6.212534e-01
DD_cohort_2016  0.965669947  0.7552897  1.27854248 2.010582e-01
```

The present age-cohort model has no calendar (period) effect, so also the first differences parameters are identified. They are found by an additional identification command. Thus, Table 4.2 in the paper is constructed from the following commands.

```
> apc.identify(fit)$coefficients.dif[,1:2]

                  Estimate Std. Error
level          7.660055032  0.1377951
D_age_1998     2.272100342  0.1335080
D_age_1999     0.932530550  0.1362610
D_age_2000     0.235606356  0.1398301
D_age_2001     0.088886609  0.1438733
D_age_2002    -0.176044303  0.1483681
D_age_2003    -0.144445459  0.1533567
D_age_2004    -0.427608601  0.1589136
D_age_2005    -0.300527594  0.1651428
D_age_2006    -0.399729999  0.1721838
D_age_2007    -0.189656058  0.1802245
D_age_2008    -0.242063670  0.1895226
D_age_2009    -0.260459607  0.2004421
D_age_2010    -0.555317528  0.2135164
D_age_2011    -0.303234088  0.2295651
D_age_2012     0.405830766  0.2499291
D_age_2013    -0.895278068  0.2769988
D_age_2014     0.116668873  0.3156054
D_age_2015    -0.383048241  0.3777268
D_age_2016    -0.273419402  0.5083832
D_cohort_1998  0.288755900  0.1335080
D_cohort_1999  0.163424236  0.1362610
D_cohort_2000 -0.264981486  0.1398301
D_cohort_2001  0.149829430  0.1438733
```

```
D_cohort_2002 -0.374386828  0.1483681
D_cohort_2003 -0.198735893  0.1533567
D_cohort_2004 -0.008807130  0.1589136
D_cohort_2005 -0.005337953  0.1651428
D_cohort_2006 -0.132272851  0.1721838
D_cohort_2007 -0.021862643  0.1802245
D_cohort_2008 -0.472602270  0.1895226
D_cohort_2009 -0.437572798  0.2004421
D_cohort_2010  0.295511564  0.2135164
D_cohort_2011  0.310545832  0.2295651
D_cohort_2012 -0.268692406  0.2499291
D_cohort_2013  0.142131410  0.2769988
D_cohort_2014  0.201777590  0.3156054
D_cohort_2015 -0.092672697  0.3777268
D_cohort_2016  0.872997251  0.5083832
```

```
> fit$s2
```

```
[1] 0.1693316
```

```
> fit$RSS
```

```
[1] 28.9557
```

5 Table 4.3: Forecasts

The forecasts are produced as follows. First the log normal forecasts use the fit found above. We want the 99.5% quantile. The reserves are computed by policy year and for the entire lower triangle. Note that the `apc` also allows aggregation by development year and by calendar year as well as single cell forecasts. Aggregation by calendar year would give a cashflow. Further, there are three columns with standard errors: the overall standard error and breakdowns into process error and estimation error.

```
> forecast    <- apc.forecast.ac(fit,quantiles=0.995)
> forecast$response.forecast.coh
```

```
          forecast        se   se.proc     se.est    t-0.995
coh_1998  1871.073  1026.463  707.4405   743.7428   4544.891
coh_1999  5099.330  1874.681 1375.8435  1273.3744   9982.659
coh_2000  7171.317  2123.128 1622.5220  1369.3412  12701.822
coh_2001 11699.350  2984.949 2274.8292  1932.6338  19474.801
coh_2002 13717.388  3345.138 2654.4080  2035.6984  22431.090
coh_2003 14343.522  3188.410 2471.3130  2014.5886  22648.964
coh_2004 18377.001  3834.057 2910.9751  2495.2390  28364.281
coh_2005 25488.052  5241.618 3976.5389  3414.9225  39141.867
```

```
coh_2006  30524.942   6213.652  4662.3320   4107.5694   46710.794
coh_2007  40078.245   8115.990  5976.5789   5490.8835   61219.471
coh_2008  32680.319   6603.511  4727.4210   4610.6241   49881.712
coh_2009  28509.077   5895.265  4143.1332   4193.8760   43865.568
coh_2010  51760.526  11013.030  7540.3989   8026.7807   80448.208
coh_2011  98747.731  22063.641 14798.3216  16365.0210  156220.991
coh_2012 100330.677  23254.845 14704.7084  18015.5316  160906.889
coh_2013 149813.314  36629.836 21310.2885  29792.8931  245229.846
coh_2014 221549.649  58610.037 29815.3239  50459.7158  374222.093
coh_2015 229480.904  69931.745 29102.9866  63588.2473  411645.102
coh_2016 575343.178 235016.967 70362.1087 224236.8135 1187535.497
```

```
> forecast$response.forecast.all
```

```
    forecast       se  se.proc   se.est t-0.995
all  1656586 267445.9 88190.59 252487.1 2353252
```

We compare with the standard chain ladder using the over-dispersed Poisson distribution forecasts of Harnau and Nielsen (2018). Here, the standard error has a third component, `tau.est`.

```
> CL.fit <- apc.fit.model(data,"od.poisson.response","AC")
> CL.forecast    <- apc.forecast.ac(CL.fit,quantiles=0.995)
> CL.forecast$response.forecast.coh
```

```
           forecast         se   se.proc     se.est    tau.est     t-0.995
coh_1998   1367.774   2472.419  1719.626   1776.238   26.88900    7808.143
coh_1999   4475.781   4120.885  3110.722   2701.363   87.98918   15210.216
coh_2000   6924.767   4745.192  3869.273   2743.546  136.13369   19285.449
coh_2001  10975.055   5928.969  4871.134   3373.155  215.75814   26419.342
coh_2002  14940.740   6519.633  5683.460   3180.821  293.71938   31923.638
coh_2003  18337.446   7135.410  6296.456   3337.478  360.49508   36924.373
coh_2004  24486.906   8225.669  7276.017   3806.503  481.38705   45913.833
coh_2005  31875.928   9355.281  8301.531   4267.690  626.64752   56245.364
coh_2006  35566.882   9836.955  8768.992   4402.451  699.20783   61191.025
coh_2007  48594.889  11673.490 10249.957   5504.188  955.32486   79002.994
coh_2008  42027.151  10902.427  9532.169   5226.666  826.20999   70426.726
coh_2009  37113.693  10490.973  8957.645   5411.911  729.61652   64441.479
coh_2010  66977.208  14927.051 12033.453   8733.793 1316.70209  105860.468
coh_2011 102982.095  20300.137 14921.333  13614.356 2024.52064  155861.631
coh_2012 136646.512  26549.216 17188.027  20055.329 2686.32798  205804.182
coh_2013 164317.838  35454.109 18848.167  29854.740 3230.31740  256671.737
coh_2014 218873.833  55148.569 21753.227  50494.034 4302.83140  362529.547
coh_2015 166119.642  82217.051 18951.224  79936.409 3265.73900  380285.655
coh_2016 337001.247 325178.113 26992.463 323988.148 6625.09322 1184053.039
```

```
> CL.forecast$response.forecast.all
```

```
    forecast       se   se.proc   se.est  tau.est t-0.995
all  1469605 350536.3 56367.29 344766.2 28890.91 2382712
```

We also look at the bootstrap forecast. This uses the bootstrap command in the `ChainLadder` package. That packages takes a cummulative triangle as input, which we can form in the `apc` package using the command `triangle.cummulative`. This command operates both on an `apc.data.list` and on a matrix. The output is a matrix.

```
> m.cum   <- triangle.cumulative(data)
```

We then call the bootstrap command in the `ChainLadder` package. In the paper we have 10^5 bootstrap repetitions. In the following code this is reduced to 10^3 repetitions.

Note, the bootstrap call does not appear to be fully stable and therefore it is commented out here and when building Table 4.3 below.

```
> library(ChainLadder)
> #BS <- BootChainLadder(m.cum, R = 10^3, process.distr=c("od.pois"))
> #summary(BS)
```

The information from the above methods are now combined into Table 4.3.

```
> Table_4_3   <- matrix(NA,nrow=20,ncol=9)
> rownames(Table_4_3) <- c(as.character(2:20),"total")
> col.3   <- c("Res","se/Res","99.5%/Res")
> colnames(Table_4_3) <- c(col.3,col.3,col.3)
> #   Table 4.3, part I, log normal Chain Ladder
> forecast.coh  <- forecast$response.forecast.coh
> forecast.all  <- forecast$response.forecast.all
> Table_4_3[1:19,1]   <- forecast.coh[,1]
> Table_4_3[1:19,2:3] <- forecast.coh[,c(2,5)]/forecast.coh[,1]
> Table_4_3[20,1]     <- forecast.all[,1]
> Table_4_3[20,2:3]   <- forecast.all[,c(2,5)]/forecast.all[,1]
> #   Table 4.3, part II, standard Chain Ladder
> CL.forecast.coh <- CL.forecast$response.forecast.coh
> CL.forecast.all <- CL.forecast$response.forecast.all
> Table_4_3[1:19,4]   <- CL.forecast.coh[,1]
> Table_4_3[1:19,5:6] <- CL.forecast.coh[,c(2,6)]/CL.forecast.coh[,1]
> Table_4_3[20,4]     <- CL.forecast.all[,1]
> Table_4_3[20,5:6]   <- CL.forecast.all[,c(2,6)]/CL.forecast.all[,1]
> #   Table 4.3, part III, bootstrap
> #sum.bs.B <- summary(BS)$ByOrigin
> #sum.bs.T <- summary(BS)$Totals
> #qua.bs.B <- quantile(BS, c(0.995))$ByOrigin
> #qua.bs.T <- quantile(BS, c(0.995))$Totals
> #Table_4_3[1:19,7]   <- sum.bs.B[2:20,3]
> #Table_4_3[1:19,8]   <- sum.bs.B[2:20,4]/sum.bs.B[2:20,3]
```

```
> #Table_4_3[1:19,9]    <- qua.bs.B[2:20,1]/sum.bs.B[2:20,3]
> #Table_4_3[20,7]      <- sum.bs.T[3,]
> #Table_4_3[20,8]      <- sum.bs.T[4,]/sum.bs.T[3,]
> #Table_4_3[20,9]      <- qua.bs.T[1,1]/sum.bs.T[3,]
> #Table_4_3
```

6 Table 4.4: Forecasts

In this table the analysis is redone when dropping the last calendar (period) year and when dropping the two last calendar years. `apc` can extract subsets of the data with the following commands.

```
> data.1       <- apc.data.list.subset(data,0,0,0,1,0,0)

WARNING apc.data.list.subset: cuts in arguments are:
[1] 0 0 0 1 0 0
have been modified to:
[1] 0 1 0 1 0 1
WARNING apc.data.list.subset: coordinates changed to "AC" & data.format changed to "t

> data.2       <- apc.data.list.subset(data,0,0,0,2,0,0)

WARNING apc.data.list.subset: cuts in arguments are:
[1] 0 0 0 2 0 0
have been modified to:
[1] 0 2 0 2 0 2
WARNING apc.data.list.subset: coordinates changed to "AC" & data.format changed to "t
```

Table 4.4 is built up from 9 subpanels, but the last row with the bootstrap panels. We start by defining the 6 top panels, labelled a,b,...,f.

```
> #   Define panels
> Table_4_4a <- matrix(NA,nrow=6,ncol=3)
> Table_4_4a[1:5,1] <- 16:20
> colnames(Table_4_4a) <- c("i","se/Res","99.5%/Res")
> Table_4_4b <- Table_4_4c <- Table_4_4d <- Table_4_4a
> Table_4_4b[1:5,1] <- 15:19
> Table_4_4c[1:5,1] <- 14:18
> Table_4_4e <- Table_4_4b
> Table_4_4f <- Table_4_4c
```

We then fill in the forecast information similar to Table 4.3.

```
> #   Panel a. log normal Chain Ladder, no cut
> for.coh <- forecast$response.forecast.coh
> for.all <- forecast$response.forecast.all
```

```
> Table_4_4a[1:5,2:3] <- for.coh[15:19,c(2,5)]/for.coh[15:19,1]
> Table_4_4a[ 6,2:3] <- for.all[     ,c(2,5)]/for.all[     ,1]
> #   Panel b. log normal Chain Ladder, cut 1 calendar year
> fit.1 <- apc.fit.model(data.1,"log.normal.response","AC")
> forecast.1  <-  apc.forecast.ac(fit.1,quantiles=c(0.995))
> for.1.coh <- forecast.1$response.forecast.coh
> for.1.all <- forecast.1$response.forecast.all
> Table_4_4b[1:5,2:3] <- for.1.coh[14:18,c(2,5)]/for.1.coh[14:18,1]
> Table_4_4b[ 6,2:3] <- for.1.all[     ,c(2,5)]/for.1.all[     ,1]
> #   Panel c. log normal Chain Ladder, cut 2 calendar years
> fit.2 <- apc.fit.model(data.2,"log.normal.response","AC")
> forecast.2  <-  apc.forecast.ac(fit.2,quantiles=c(0.995))
> for.2.coh <- forecast.2$response.forecast.coh
> for.2.all <- forecast.2$response.forecast.all
> Table_4_4c[1:5,2:3] <- for.2.coh[13:17,c(2,5)]/for.2.coh[13:17,1]
> Table_4_4c[ 6,2:3] <- for.2.all[     ,c(2,5)]/for.2.all[     ,1]
> #   Panel d. Standard Chain Ladder, no cut
> CL.for.coh <- CL.forecast$response.forecast.coh
> CL.for.all <- CL.forecast$response.forecast.all
> Table_4_4d[1:5,2:3] <- CL.for.coh[15:19,c(2,6)]/CL.for.coh[15:19,1]
> Table_4_4d[ 6,2:3] <- CL.for.all[     ,c(2,6)]/CL.for.all[     ,1]
> #   Panel e. Standard Chain Ladder, cut 1 calendar year
> CL.fit.1        <- apc.fit.model(data.1,"od.poisson.response","AC")
> CL.forecast.1   <- apc.forecast.ac(CL.fit.1,quantiles=c(0.995))
> CL.for.coh <- CL.forecast.1$response.forecast.coh
> CL.for.all <- CL.forecast.1$response.forecast.all
> Table_4_4e[1:5,2:3] <- CL.for.coh[14:18,c(2,6)]/CL.for.coh[14:18,1]
> Table_4_4e[ 6,2:3] <- CL.for.all[     ,c(2,6)]/CL.for.all[     ,1]
> #   Panel f. Standard Chain Ladder, cut 2 calendar years
> CL.fit.2        <- apc.fit.model(data.2,"od.poisson.response","AC")
> CL.forecast.2   <- apc.forecast.ac(CL.fit.2,quantiles=c(0.995))
> CL.for.coh <- CL.forecast.2$response.forecast.coh
> CL.for.all <- CL.forecast.2$response.forecast.all
> Table_4_4f[1:5,2:3] <- CL.for.coh[13:17,c(2,6)]/CL.for.coh[13:17,1]
> Table_4_4f[ 6,2:3] <- CL.for.all[     ,c(2,6)]/CL.for.all[     ,1]
> #   Combine table
> rbind(cbind(Table_4_4a,Table_4_4b,Table_4_4c),
+       cbind(Table_4_4d,Table_4_4e,Table_4_4f))
```

```
      i    se/Res 99.5%/Res  i    se/Res 99.5%/Res  i    se/Res 99.5%/Res
 [1,] 16 0.2317820  1.603766 15 0.2330485  1.607871 14 0.2325165  1.607441
 [2,] 17 0.2445032  1.636903 16 0.2454914  1.640326 15 0.2453367  1.640933
 [3,] 18 0.2645458  1.689112 17 0.2653584  1.692146 16 0.2653280  1.693160
 [4,] 19 0.3047388  1.793810 18 0.3054005  1.796590 17 0.3054789  1.798052
 [5,] 20 0.4084814  2.064047 19 0.4090368  2.066909 18 0.4091658  2.068930
```

```
 [6,] NA 0.1614440  1.420543 NA 0.1252687  1.326744 NA 0.1181010  1.308534
 [7,] 16 0.1942912  1.506106 15 0.2028927  1.529214 14 0.2237359  1.584502
 [8,] 17 0.2157654  1.562044 16 0.2158180  1.562928 15 0.2387938  1.623840
 [9,] 18 0.2519651  1.656340 17 0.2831099  1.738448 16 0.2766660  1.722780
[10,] 19 0.4949267  2.289228 18 0.4784675  2.248008 17 0.4775386  2.247552
[11,] 20 0.9649166  3.513497 19 1.4000506  4.651814 18 1.5254827  4.985267
[12,] NA 0.2385241  1.621328 NA 0.2070006  1.539929 NA 0.2020588  1.527871
```

7 Table 4.5: Bartlett tests

Here we have a variety of specification tests. They are not coded in apc as yet, so they require a bit of work. We start with Bartlett's test and define a function that can take 2 or 3 subsets.

```
> Bartlett <- function(data,model.family,model.design,s.1,s.2,s.3=NULL)
+ #   data is an apc.data.list
+ #   s are subset indices, that is sets of six numbers
+ {   fit.model <- function(data,model.family,model.design,s)
+     {   data.s <- apc.data.list.subset(data,s[1],s[2],s[3],s[4],s[5],s[6],
+                             suppress.warning=TRUE)
+         fit <- apc.fit.model(data.s,model.family,model.design)
+         dev <- fit$deviance
+         if(model.family=="log.normal.response")
+             dev <- fit$RSS
+         return(list(dev=dev, df=fit$df.residual))
+     }
+     fit <- fit.model(data,model.family,model.design,s.1)
+     dev <- fit$dev
+      df <- fit$df
+     fit <- fit.model(data,model.family,model.design,s.2)
+     dev <- c(dev,fit$dev)
+      df <- c( df,fit$df )
+     m <- 2
+     if(!is.null(s.3))
+     {   fit <- fit.model(data,model.family,model.design,s.3)
+         dev <- c(dev,fit$dev)
+          df <- c( df,fit$df )
+         m <- 3
+     }
+     dev.<- sum(dev)
+      df.<- sum(df)
+     LR  <- df.*log(dev./df.)-sum(df*log(dev/df))
+     C   <- 1+(1/3/(m-1))*(sum(1/df)-1/df.)
+     t   <- LR/C
+     p   <- pchisq(LR/C,m-1,0,FALSE)
```

```
+     return(list(t=t,p=p))
+ }
```

The next function is for testing the mean of subsets using F tests when the variance is common.

```
> Ftest <- function(data,model.family,model.design,s.1,s.2,s.3=NULL)
+ #   data is an apc.data.list
+ #   s are subset indices, that is sets of six numbers
+ {   append <- function(data,model.design,s,v=NULL,d=NULL)
+     {   data.s <- apc.data.list.subset(data,s[1],s[2],s[3],s[4],s[5],s[6],
+                             suppress.warning=TRUE)
+         index  <- apc.get.index(data.s)
+         v1 <- index$response[index$index.data]
+         d1 <- apc.get.design(index,model.design)$design
+         if(is.null(v))  v <- v1
+         else            v <- c(v,v1)
+         if(is.null(d))  d <- d1
+         else
+         {   d0 <- matrix(0,nrow(d),ncol(d1))
+             d10 <- matrix(0,nrow(d1),ncol(d))
+             d <- rbind(cbind(d,d0),cbind(d10,d1))
+         }
+         return(list(v=v,d=d))
+     }
+     a <- append(data,model.design,s.1)
+     v <- a$v; d <- a$d
+     a <- append(data,model.design,s.2,v,d)
+     v <- a$v; d <- a$d
+     if(!is.null(s.3))
+     {   a <- append(data,model.design,s.3,v,d)
+         v <- a$v; d <- a$d  }
+     fit.R <- apc.fit.model(data,model.family,model.design)
+     if(model.family=="log.normal.response")
+     {   fit.R$deviance <- fit.R$RSS
+         fit.U <- glm.fit(d,log(v),family=gaussian(link = "identity"))
+     }
+     if(model.family=="od.poisson.response")
+         fit.U <- glm.fit(d,v,family=quasipoisson(link = "log"))
+     dev.R <- fit.R$deviance
+     dev.U <- fit.U$deviance
+      df.R <- fit.R$df.residual
+      df.U <- fit.U$df.residual
+     F <- (dev.R-dev.U)/(df.R-df.U)/(dev.U/df.U)
+     p <- pf(F,df.R-df.U,df.U,lower.tail=FALSE)
+ #    #   reproducing typo
```

```
+ #    F <- (dev.R/df.R)/(dev.U/df.U)
+ #    p <- pf(F,df.R,df.U,lower.tail=FALSE)
+     return(list(F=F,p=p))
+ }
```

We now define Table 4.5 and input the results from Bartlett's test.

```
> dim.names <- list(c("a","b","c"),c("LR/C","p","F","p","LR/C","p","F","p"))
> Table_4_5 <- matrix(NA,nrow=3,ncol=8,dimnames=dim.names)
> s1 <- c(0,0,0,0,0,14)
> s2 <- c(0,0,0,0,6,0)
> Bt <- Bartlett(data,"log.normal.response","AC",s1,s2)
> Ft <-    Ftest(data,"log.normal.response","AC",s1,s2)
> Table_4_5[1,1:4] <- c(Bt$t,Bt$p,Ft$F,Ft$p)
> Bt <- Bartlett(data,"od.poisson.response","AC",s1,s2)
> Ft <-    Ftest(data,"od.poisson.response","AC",s1,s2)
> Table_4_5[1,5:8] <- c(Bt$t,Bt$p,Ft$F,Ft$p)
> s1 <- c(0,0,0,10,0,0)
> s2 <- c(0,0,10,0,0,10)
> s3 <- c(0,0,0,0,10,0)
> Bt <- Bartlett(data,"log.normal.response","AC",s1,s2,s3)
> Ft <-    Ftest(data,"log.normal.response","AC",s1,s2,s3)
> Table_4_5[2,1:4] <- c(Bt$t,Bt$p,Ft$F,Ft$p)
> Bt <- Bartlett(data,"od.poisson.response","AC",s1,s2,s3)
> Ft <-    Ftest(data,"od.poisson.response","AC",s1,s2,s3)
> Table_4_5[2,5:8] <- c(Bt$t,Bt$p,Ft$F,Ft$p)
> s1 <- c(0,0,0 ,6,0,0)
> s2 <- c(0,0,14,0,0,0)
> Bt <- Bartlett(data,"log.normal.response","AC",s1,s2)
> Ft <-    Ftest(data,"log.normal.response","AC",s1,s2)
> Table_4_5[3,1:4] <- c(Bt$t,Bt$p,Ft$F,Ft$p)
> Bt <- Bartlett(data,"od.poisson.response","AC",s1,s2)
> Ft <-    Ftest(data,"od.poisson.response","AC",s1,s2)
> Table_4_5[3,5:8] <- c(Bt$t,Bt$p,Ft$F,Ft$p)
> Table_4_5
```

```
      LR/C          p        F            p     LR/C            p        F
a 6.287150 0.01216164 5.504489 3.481042e-08 11.67530 0.0006333518 6.627022
b 4.703779 0.09518913 4.484162 1.685108e-09 11.63477 0.0029753818 6.033364
c 1.116055 0.29076951 3.080728 7.938771e-06 15.07004 0.0001035946 2.504775
             p
a 5.142369e-10
b 2.717596e-13
c 2.616378e-04
```

References

Harnau, J. and Nielsen, B. (2018) Asymptotic theory for over-dispersed age-period-cohort and extended chain-ladder models. *Journal of the American Statistical Association* 113, 1722-1732 *Download*: Earlier version: https://www.nuffield.ox.ac.uk/economics/Papers/2017/HarnauNielsen2017apcDP.pdf

Kuang, D. and Nielsen, B. (2020) Generalized Log-Normal Chain-Ladder. *Scandinavian Actuarial Journal* 2020, 553-576. *Download*: Open access: https://www.tandfonline.com/doi/full/10.1080/03461238.2019.1696885. Earlier version: https://www.nuffield.ox.ac.uk/economics/Papers/2018/2018W02_KuangNielsen2018GLNCL.pdf.

Kuang, D., Nielsen, B. and Nielsen, J.P. (2008a) Identification of the age-period-cohort model and the extended chain ladder model. *Biometrika* 95, 979-986. *Download*: Earlier version: http://www.nuffield.ox.ac.uk/economics/papers/2007/w5/KuangNielsenNielsen07.pdf.

Kuang, D., Nielsen, B. and Nielsen, J.P. (2008b) Forecasting with the age-period-cohort model and the extended chain-ladder model. *Biometrika* 95, 987-991. *Download*: Earlier version: http://www.nuffield.ox.ac.uk/economics/papers/2008/w9/KuangNielsenNielsen_Forecast.pdf.

Nielsen, B. (2014) Deviance analysis of age-period-cohort models. *Download*: http://www.nuffield.ox.ac.uk/economics/papers/2014/apc_deviance.pdf.

Nielsen, B. (2015) apc: An R package for age-period-cohort analysis. *R Journal* 7, 52-64. *Download*: https://journal.r-project.org/archive/2015-2/nielsen.pdf.