Evaluating and Selecting a Bayesian MCMC Model

Casualty Actuarial Society Spring 2023 Meeting

Presenter: Michael R Larsen

Antitrust Notice

- **The Casualty Actuarial Society is committed to adhering strictly to the letter and spirit of the antitrust laws. Seminars conducted under the auspices of the CAS are designed solely to provide a forum for the expression of various points of view on topics described in the programs or agendas for such meetings.**

- **Under no circumstances shall CAS seminars be used as a means for competing companies or firms to reach any understanding – expressed or implied – that restricts competition or in any way impairs the ability of members to exercise independent business judgment regarding matters affecting competition.**

- **It is the responsibility of all seminar participants to be aware of antitrust regulations, to prevent any written or verbal discussions that appear to violate these laws, and to adhere in every respect to the CAS antitrust compliance policy.**

Goals of Presentation

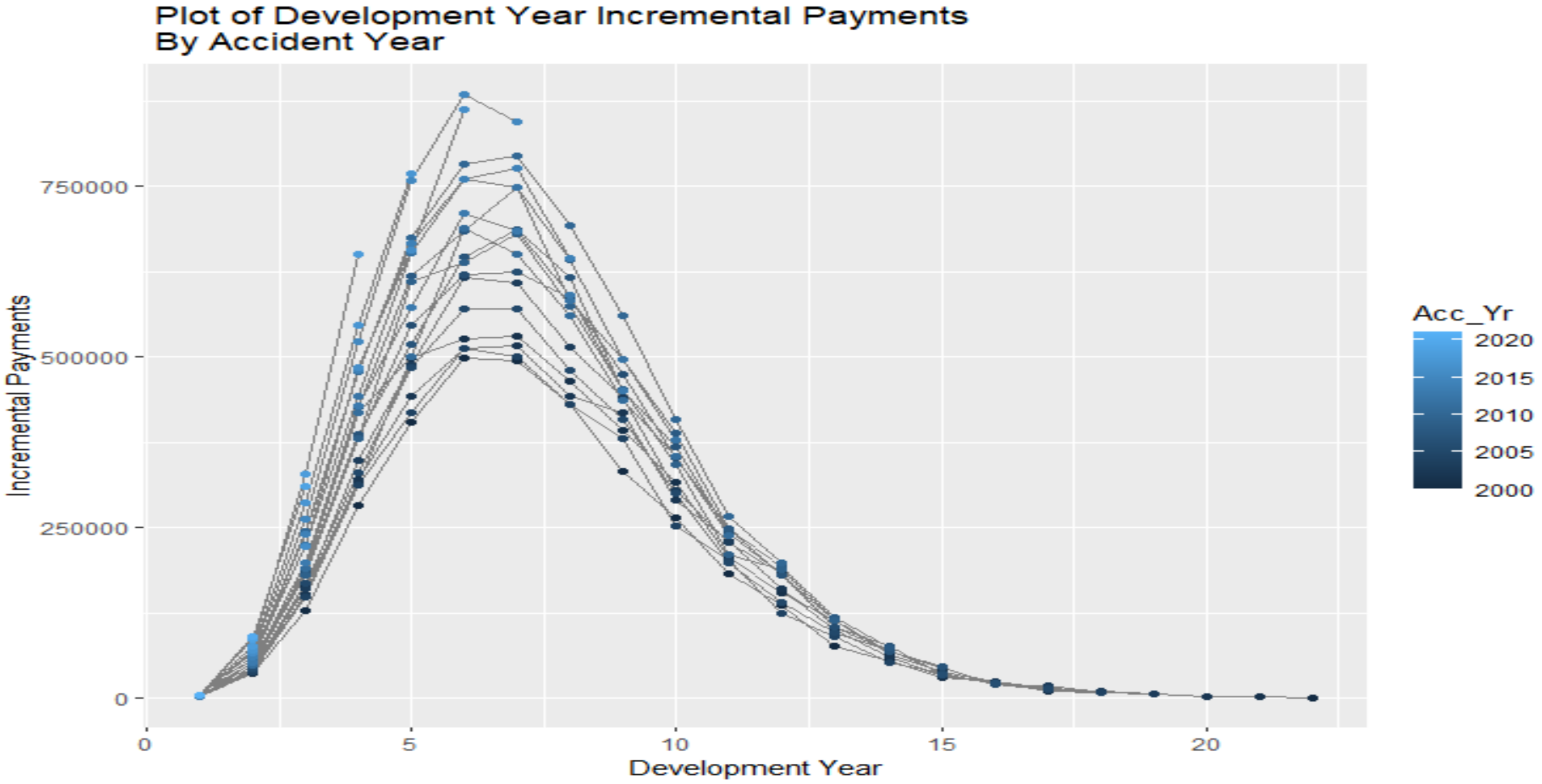
- Demonstrate steps in model selection
  - Exploratory Data Analysis
  - Identify potential model forms
  - Build alternative models
  - Check integrity of model estimates
  - Evaluate model fit
  - Compare predictive power
- Highlight software that makes this practical
  - Ggplot2
  - STAN
  - Brms
  - Bayesplot & ShinyStan:
  - Tidybayes & ggdist
  - Loo and loo_compare

Modeling Environment

Exploratory Data Analysis

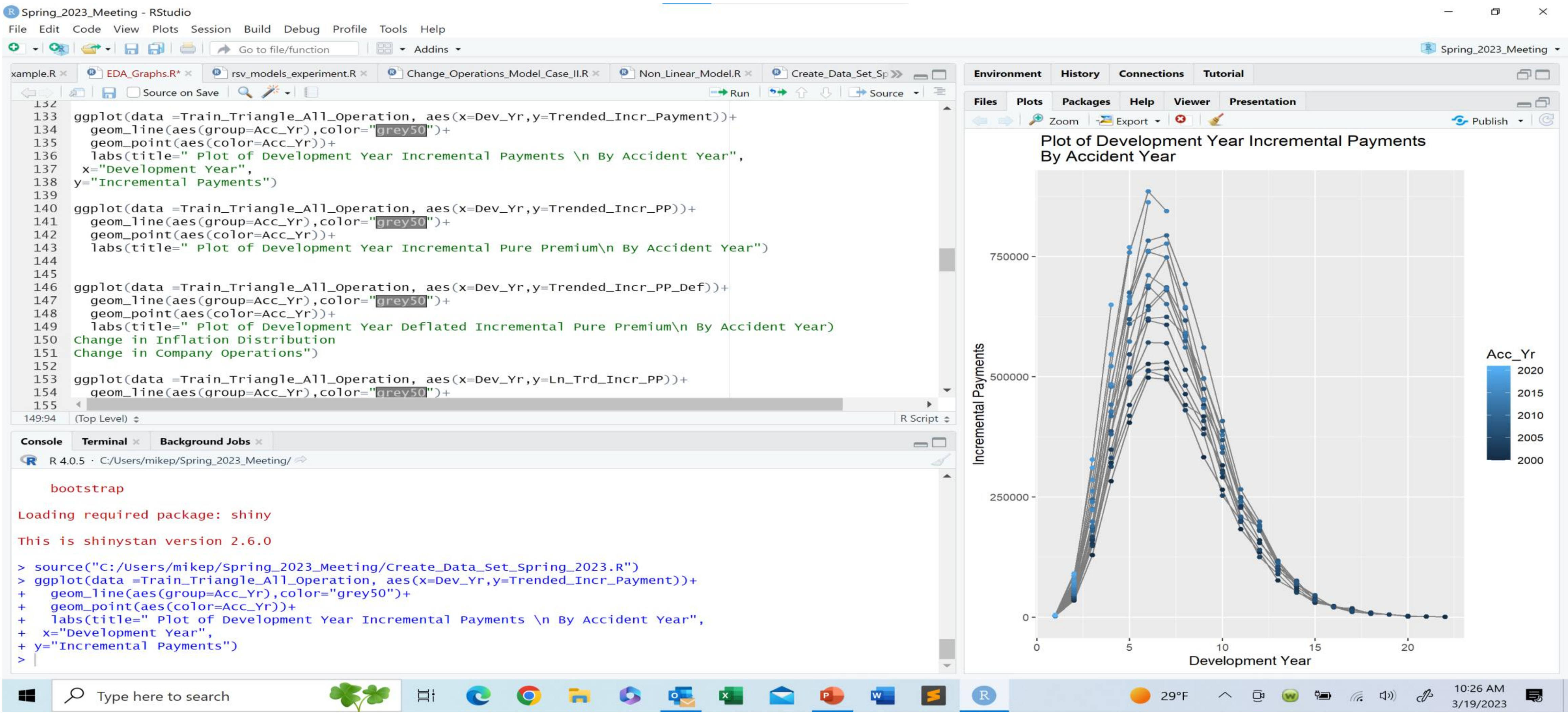
Exploratory Data Analysis

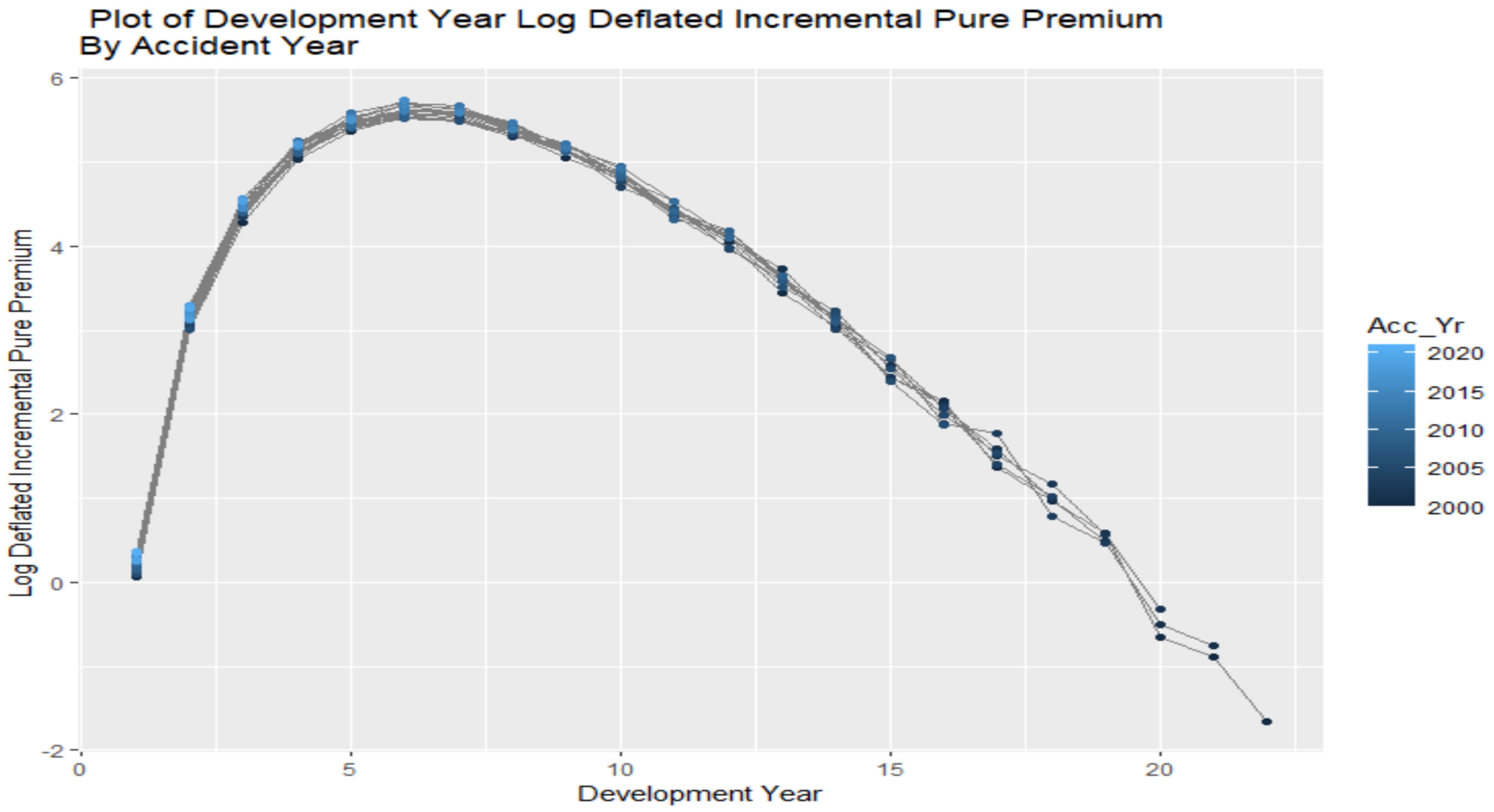
- Goals for this presentation
  - Display patterns as first step to writing formulas for models
  - Information on likely predictive variable distribution
- Additional steps for real life analysis
  - Search for anomalous behavior in claim handling by region or business unit
  - Look for changes in behavior in claims activity over time caused by change in underwriting practice
  - Check for claim recoding effects
- Data Source
  - Simulated data using Poisson for counts & Lognormal for severity
  - Underlying distributions displayed in appendix



Plot of Development Year Incremental Payments
By Accident Year
Incremental Payments
750000
500000
250000
0
0
5
10
15
20
Development Year
Acc_Yr
2020
2015
2010
2005
2000

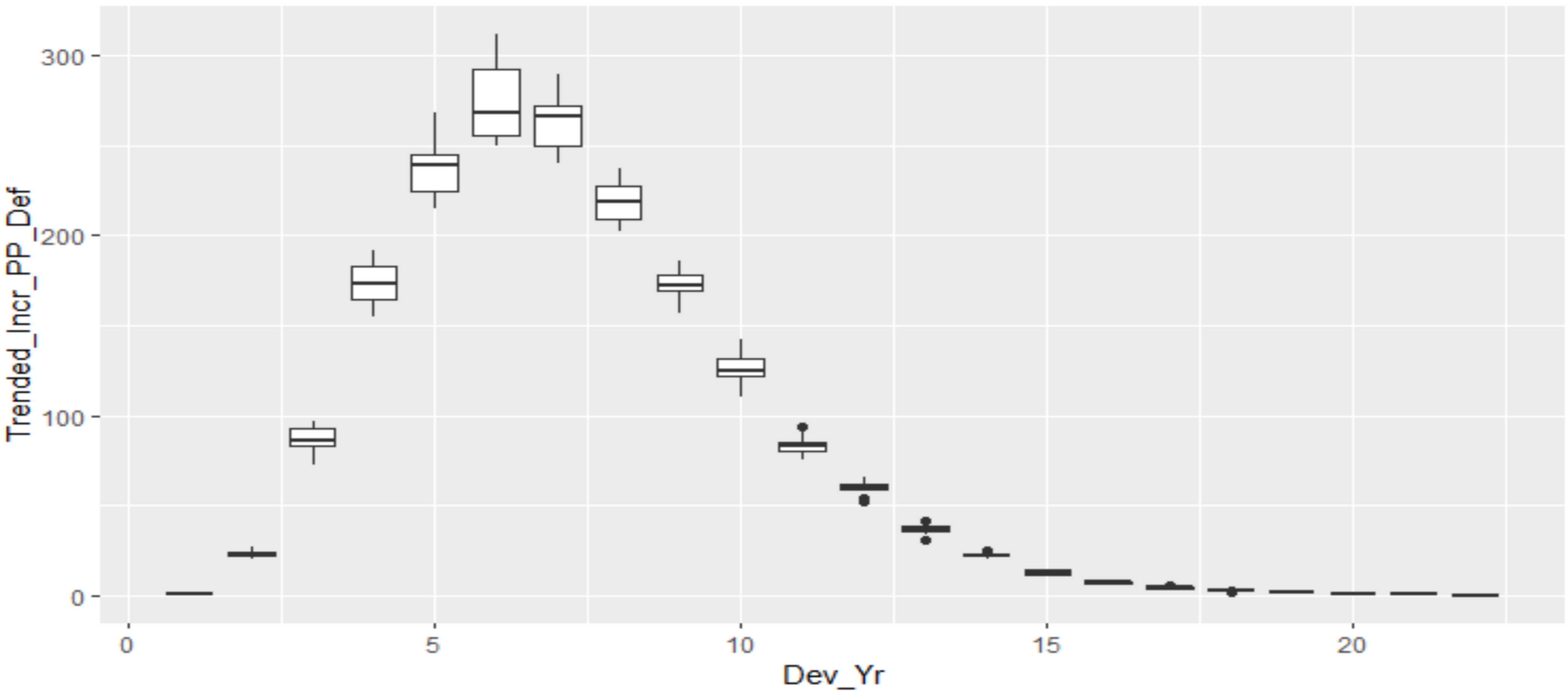
Ggplot2 Example

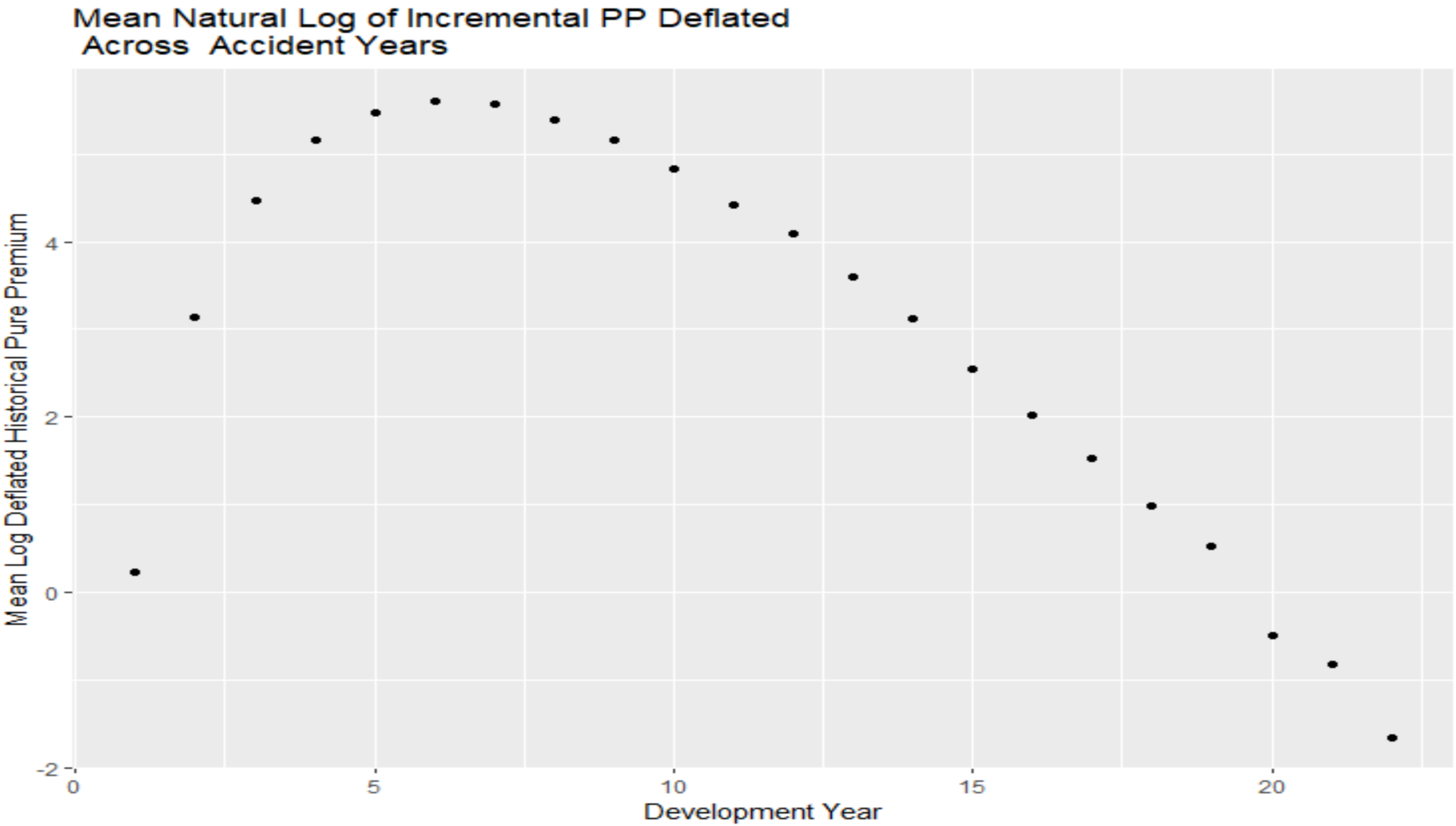




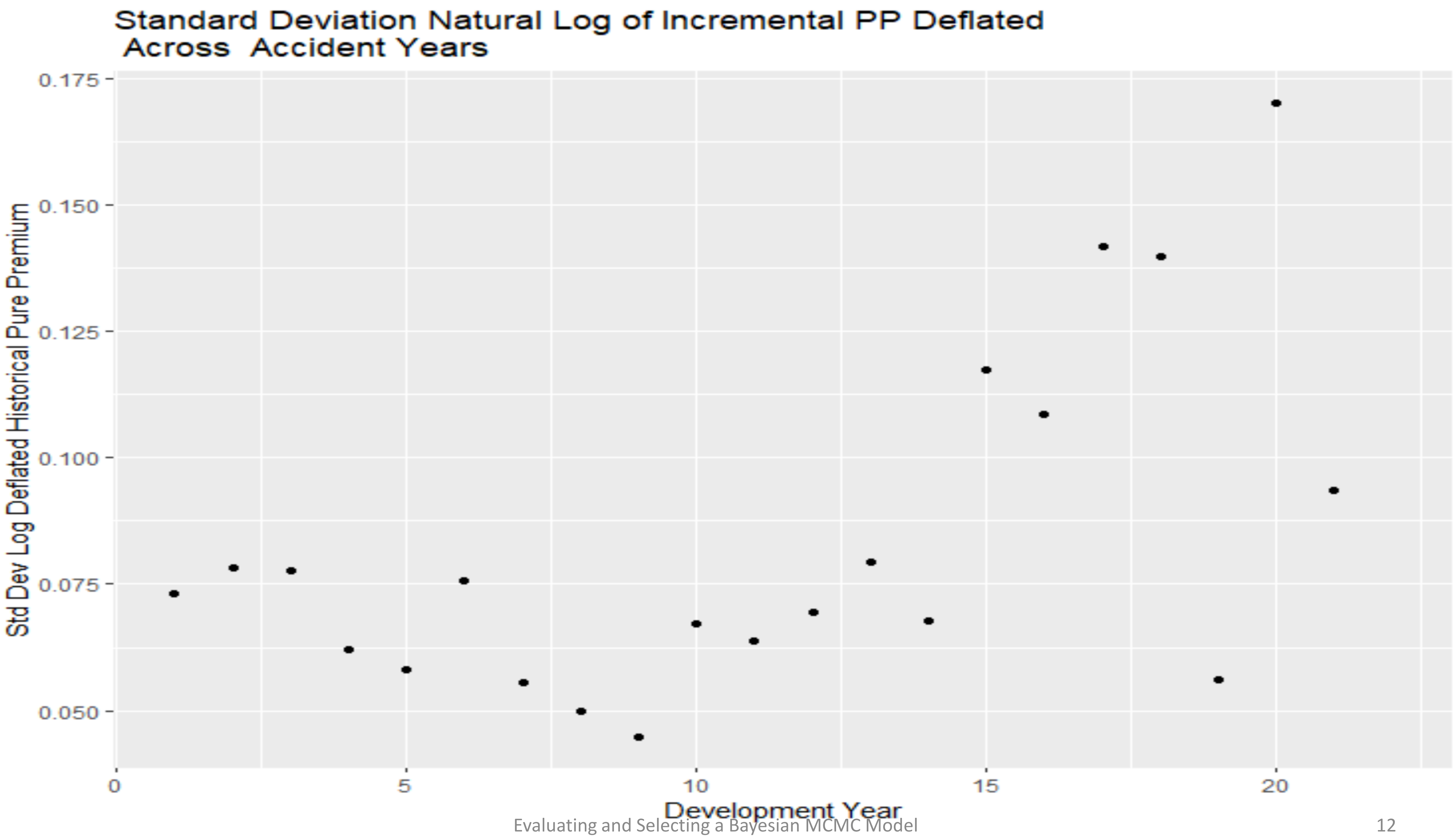
Plot of Development Year Log Deflated Incremental Pure Premium
By Accident Year
Log Deflated Incremental Pure Premium
Development Year
Acc_Yr
2020
2015
2010
2005
2000
6
4
2
0
-2
0
5
10
15
20

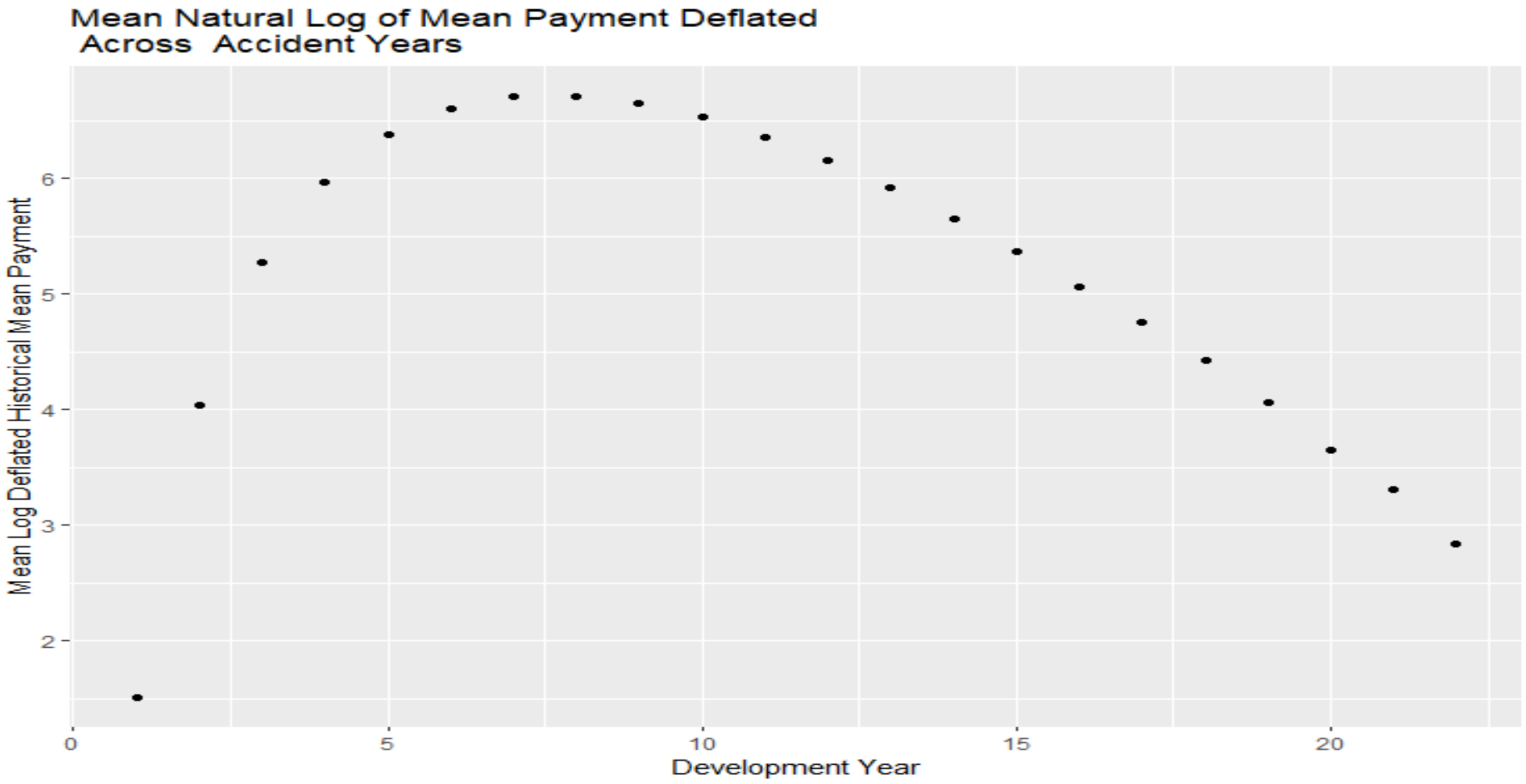
Box Plot of Deflated Incremental Pure Premium
Across Accident Year

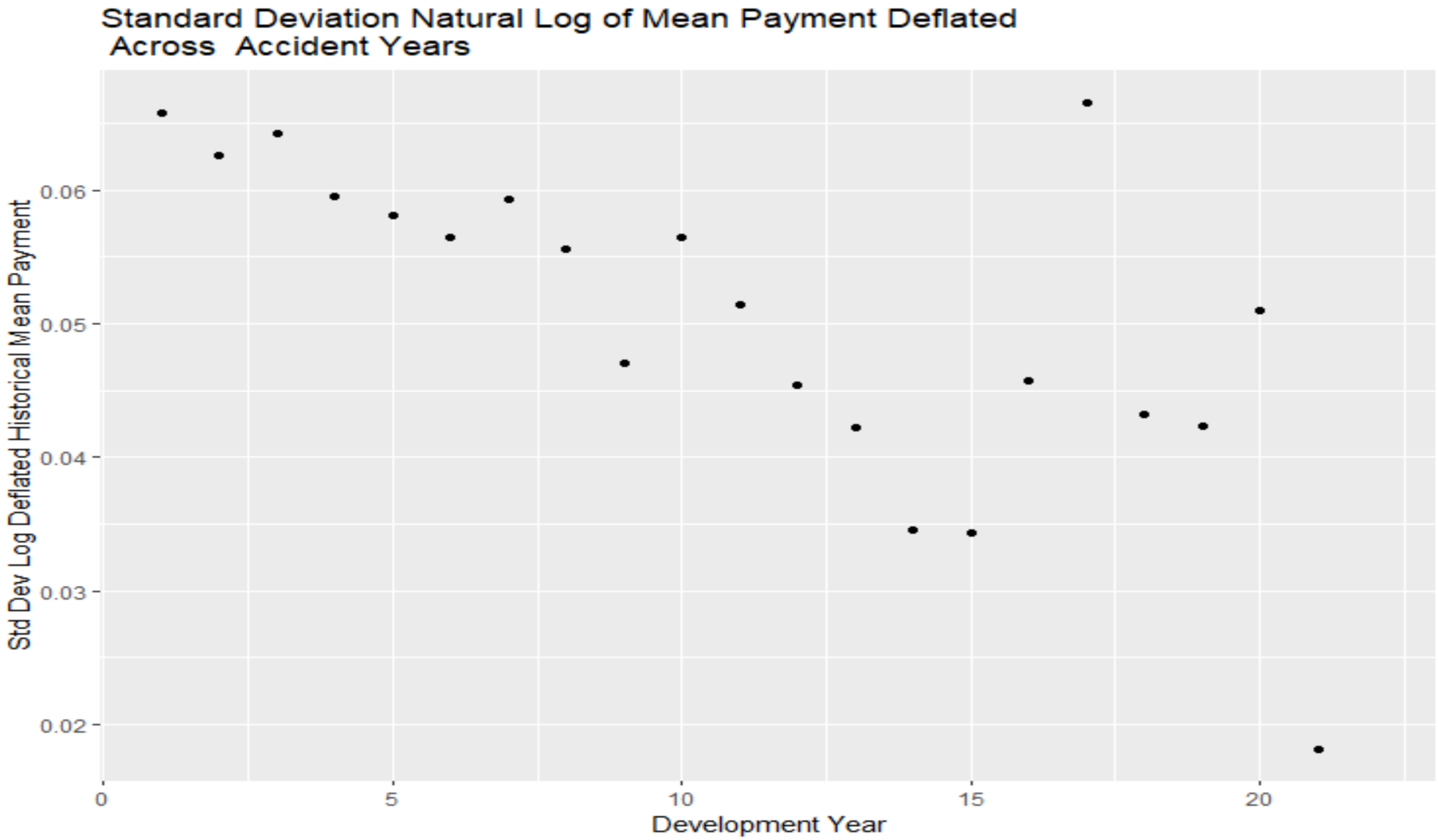




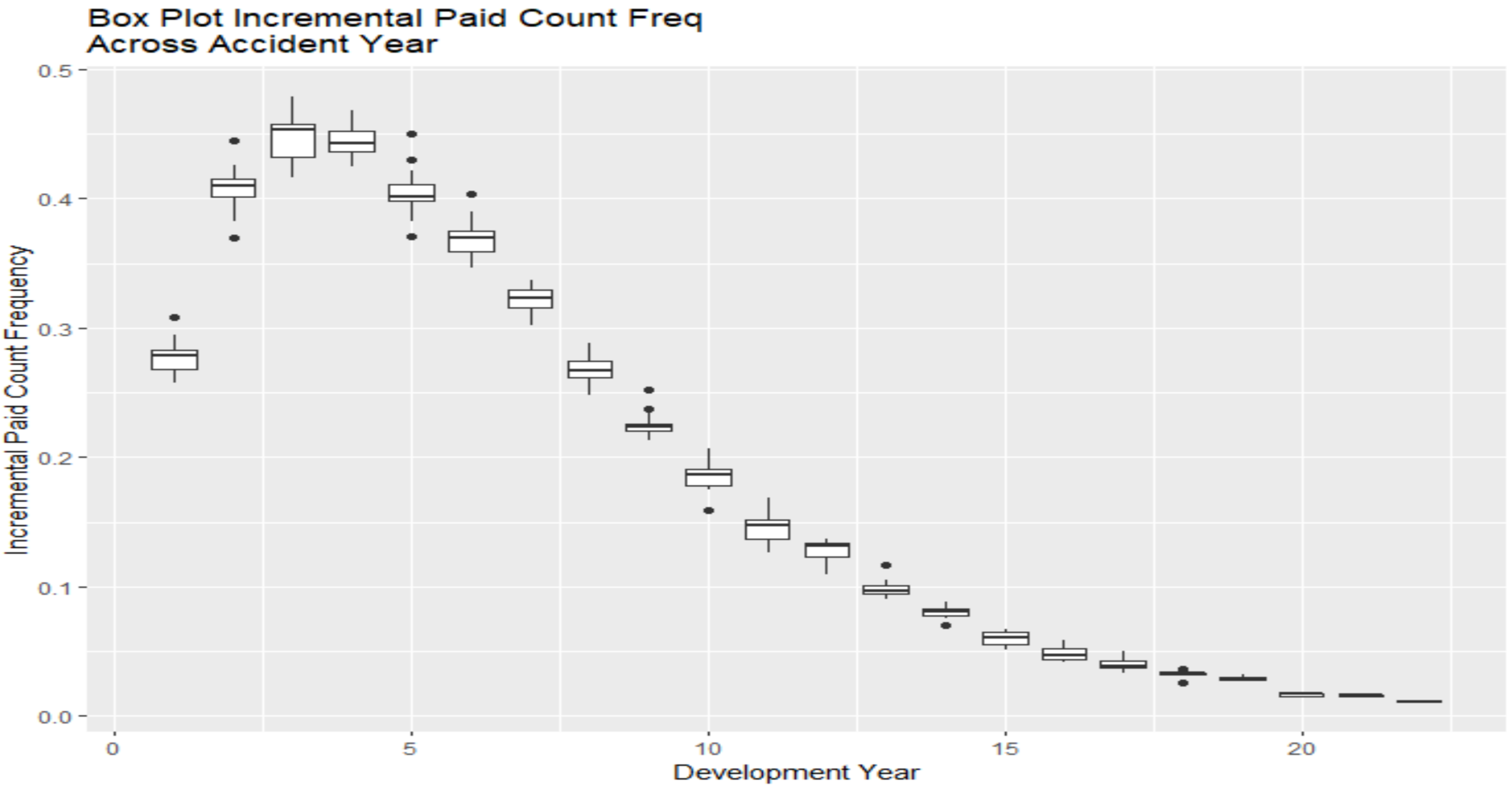
Mean Natural Log of Incremental PP Deflated
Across Accident Years
Mean Log Deflated Historical Pure Premium
4
2
0
-2
0
5
10
15
20
Development Year


Standard Deviation Natural Log of Incremental PP Deflated
Across Accident Years
Std Dev Log Deflated Historical Pure Premium
0.175
0.150
0.125
0.100
0.075
0.050
0
5
10
15
20
Development Year

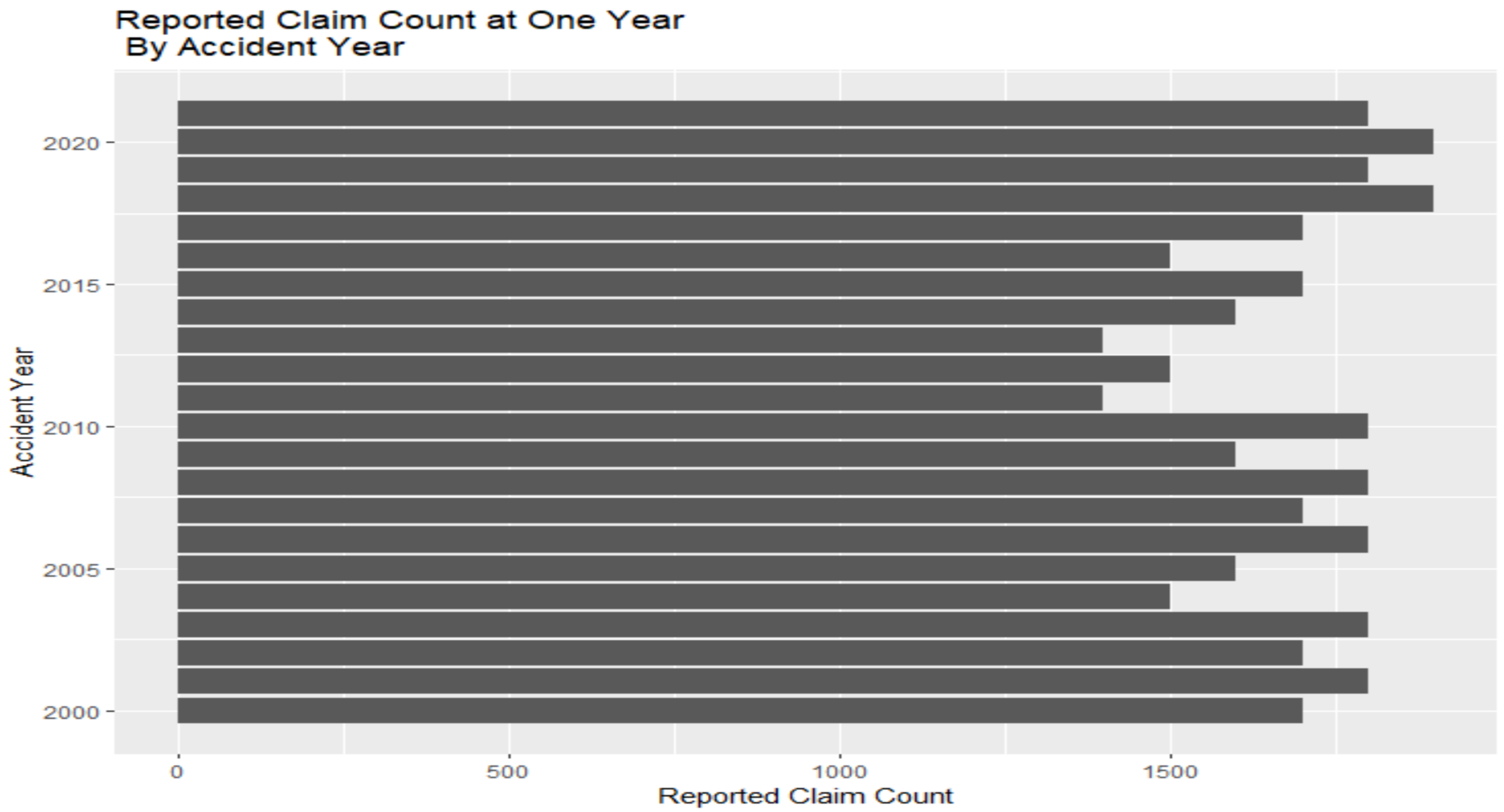

Mean Natural Log of Mean Payment Deflated
Across Accident Years
Mean Log Deflated Historical Mean Payment
2
3
4
5
6
0
5
10
15
20
Development Year



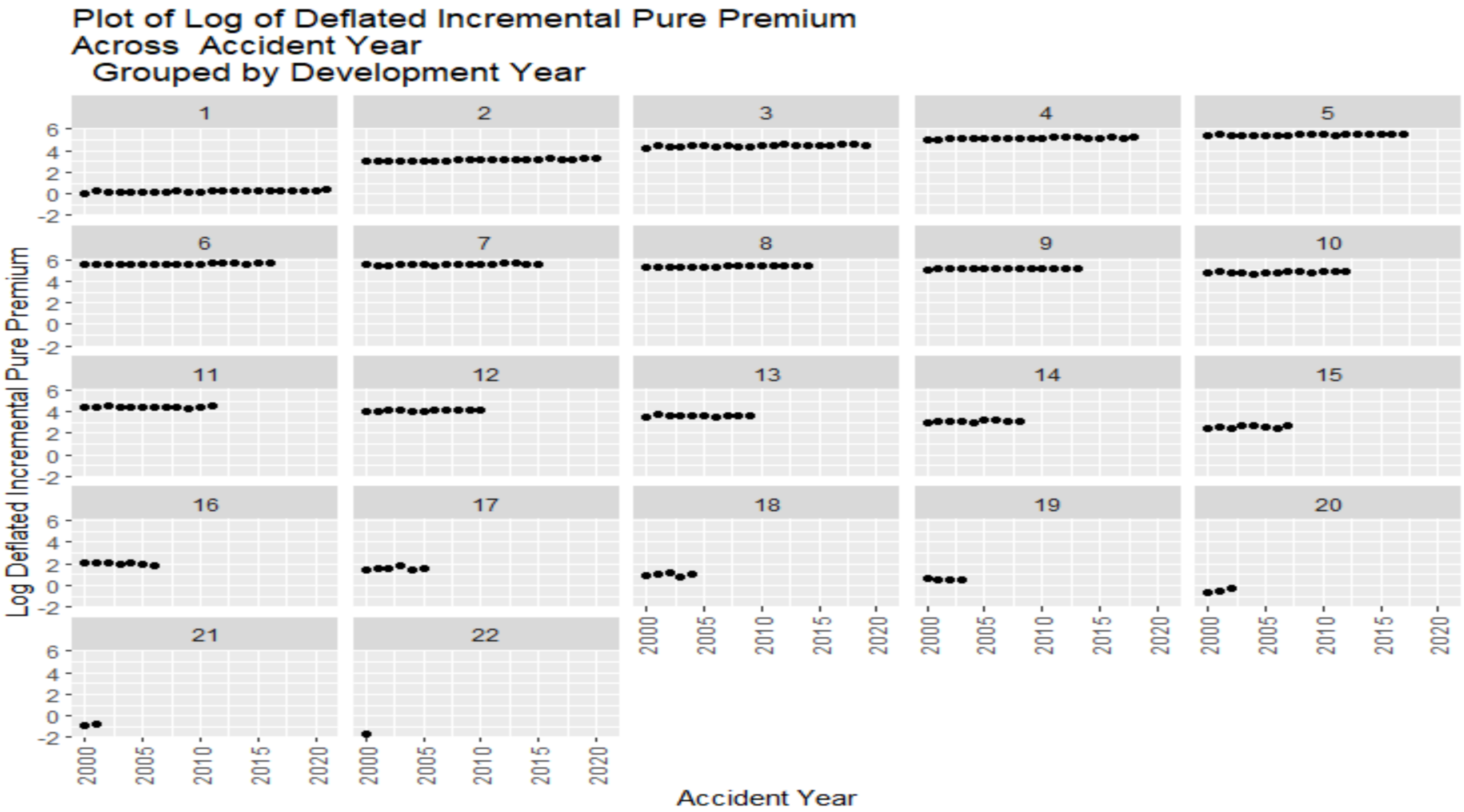
Standard Deviation Natural Log of Mean Payment Deflated
Across Accident Years
Std Dev Log Deflated Historical Mean Payment
0.06
0.05
0.04
0.03
0.02
0
5
10
15
20
Development Year



Box Plot Incremental Paid Count Freq
Across Accident Year
Incremental Paid Count Frequency
0.0
0.1
0.2
0.3
0.4
0.5
0
5
10
15
20
Development Year

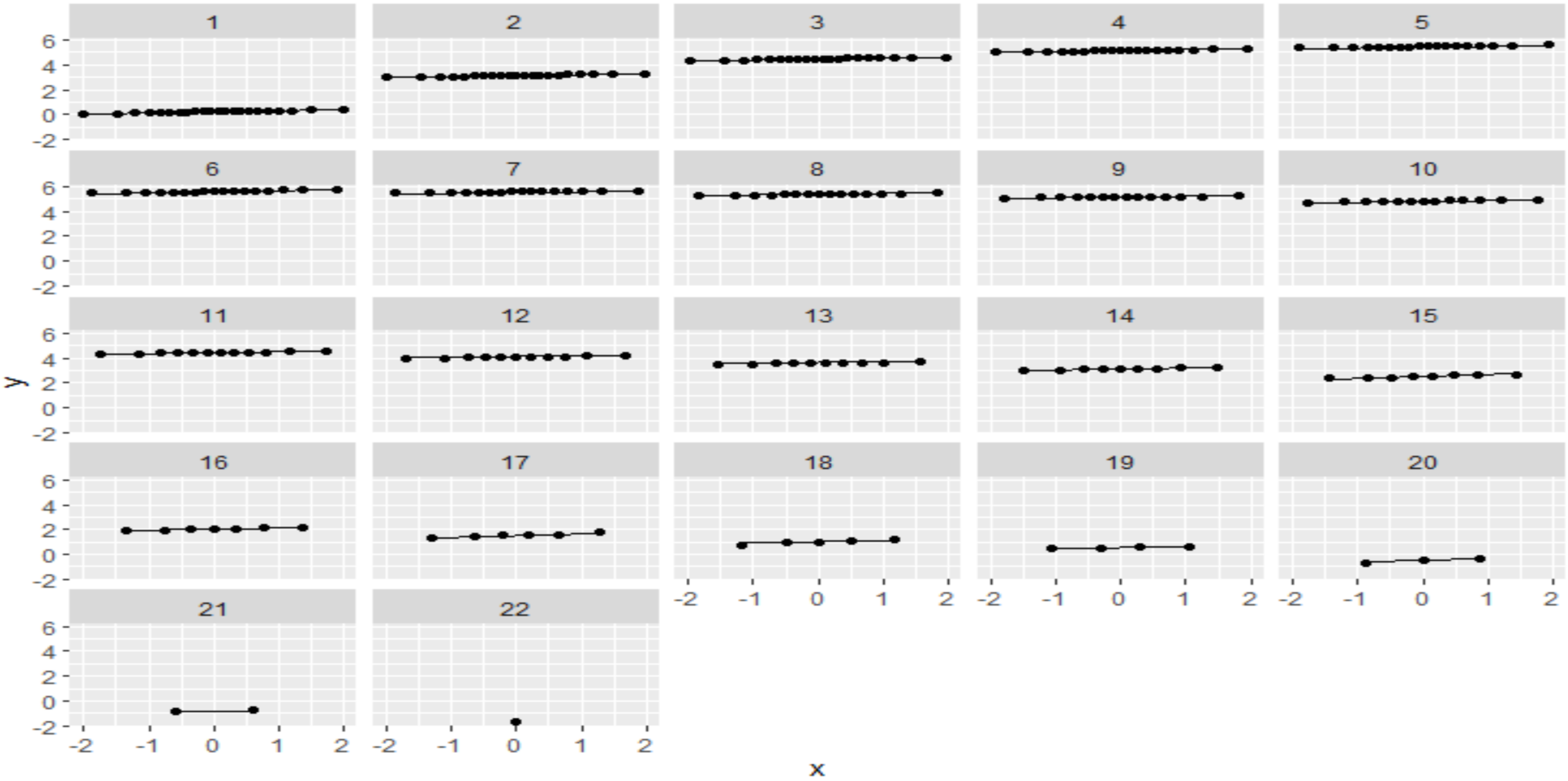


Reported Claim Count at One Year
By Accident Year
Accident Year
2020
2015
2010
2005
2000
0
500
1000
1500
Reported Claim Count



Plot of Log of Deflated Incremental Pure Premium
Across Accident Year
Grouped by Development Year
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
Log Deflated Incremental Pure Premium
6
4
2
0
-2
2000
2005
2010
2015
2020
Accident Year

QQ Plot for Log of Deflated Pure Premium Incremental Payments By Development Year



EDA Observations

- Lognormal distribution should work for dependent variable
- Mean shows some form of a parabolic development pattern
- Sigma generally decreases first 10 development years then starts to increase
- No sign of change in rate of loss cost increases
- No sign of change in claim handling or business mix

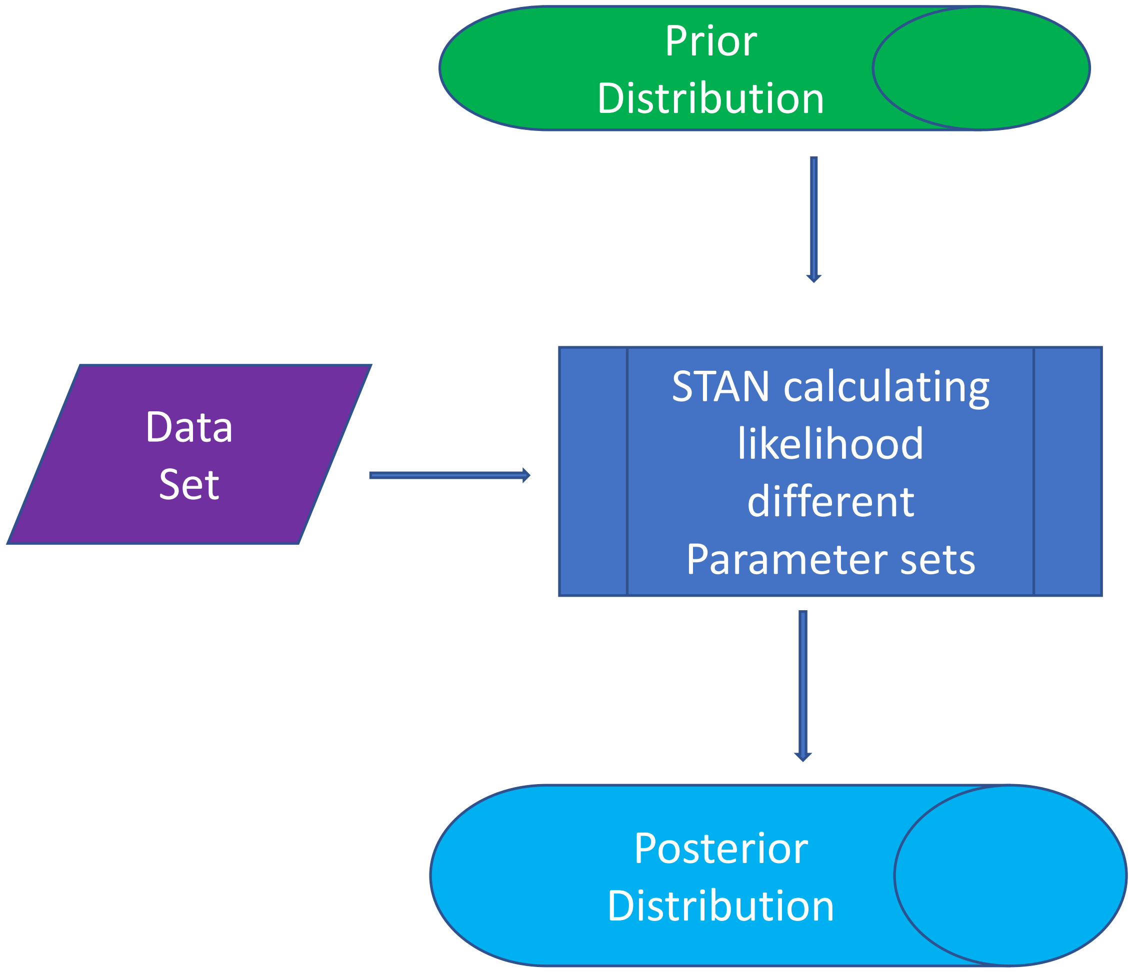
Alternative Models

Model forms

- Model incremental paid loss to isolate effect of inflation
- Normalize losses (incremental deflated pure premium)
  - Divide incremental paid losses by exposure (claim count at 12 months)
  - Divide incremental losses by accumulated CPI type index
- Incremental pure premium vs. incremental counts and average deflated severity
  - Modeling counts and amounts implies measuring correlation between counts & severity estimates which is a more complicated model
  - Built in log pointwise prediction comparison tools need a single, common predictive value
  - Appealing but impractical model form for this exercise
- Potential model forms:
  - Polynomial for development year & random effects (group) for accident year
  - Non-linear for development year & random effects (group) for accident year
  - Generalized Additive Model for development year & random effects (group) for accident year
  - Additional inflation effects as calendar year time elapsed

Effect of STAN on Modeling

- Execution time is much better
- Bayesian MCMC solves for parametric parameters iteratively for a wide range of model structures
- Start with a given set of parameters (prior distribution) and compare successive alternative sets effect on likelihood function substituting parameters that yield better likelihood results until convergence
- STAN uses an algorithm (Hamiltonian) that is much more efficient at selecting the next set of parameters to try than earlier Bayesian MCMC algorithms
- Diagnostics on algorithm execution success available



Polynomial 1 Prior Definitions

```
Polynomial_1_prior <- c(prior(normal(.6,.2),class=b, coef= Dev_Yr_6_Cap),
        prior(normal(-.2,.2),class=b, coef= Dev_Yr_6_Cap_Sqrd),
        prior(normal(-.2,.1),class=b, coef= Dev_Yr_6_Spline),
        prior(normal(-.1,.05),class=b, coef= Dev_Yr_6_Spline_Sqrd),
        prior(normal(1,.25),class=b, coef=Ln_Dev_Yr),
        prior(normal(-1,.25),class=b ,coef =Intercept),
        prior(normal(.02,.01),class=b, coef=Cal_Yr_Time),
        prior(normal(0.2,.1),class=b ,coef =Intercept, dpar=sigma),
        prior(normal(-.1,.05),class=b, coef=Dev_Yr_10_Cap,dpar=sigma),
        prior(student_t(3,.1,.05),class=b, coef=Dev_Yr_10_Spline,dpar=sigma))
```

Prior distributions reflect your knowledge of the subject in terms of plausible results.

The priors with a “dpar=“ are used to give instructions for model components besides the mean or mu for the lognormal distribution

A listing of variable definitions is given in the appendix.

Grammar for brms models

- Brm: calls the brms routine
- Components bolted on as needed via “+” signs
- bf( ): defines the formulas used to estimate the mean and other parameters
- Iter: tells the routine how many times to simulate the parameters to solve in MCMC routine
- Prior: identifies the prior distributions to be used in a model
- Seed: sets the simulation seed to ensure results can be replicated
- Control: instructions to STAN when default settings don’t work
- Data: name of data set

Polynomial Model 1 brms Instructions

```
Model_Polynomial_1 <- brm(bf(Trended_Incr_PP_Def ~ 0 + Intercept + Ln_Dev_Yr +
          Dev_Yr_6_Cap + Dev_Yr_6_Cap_Sqrd +
           Dev_Yr_6_Spline  + Dev_Yr_6_Spline_Sqrd +(1||Acc_Yr) +
           Cal_Yr_Time ,
      sigma ~  0+ Intercept + Dev_Yr_10_Cap + Dev_Yr_10_Spline ),
     iter = 4000,
     prior= Polynomial_1_prior,
     seed= 8603529,
    control = list(max_treedepth=15 ),
     data = Train_Triangle_All_Operation, family = lognormal())
```

Mu or the mean's model is defined after the first "~".

Sigma's model is defined after the second "~".

Other modeling instructions are bolted on as needed.

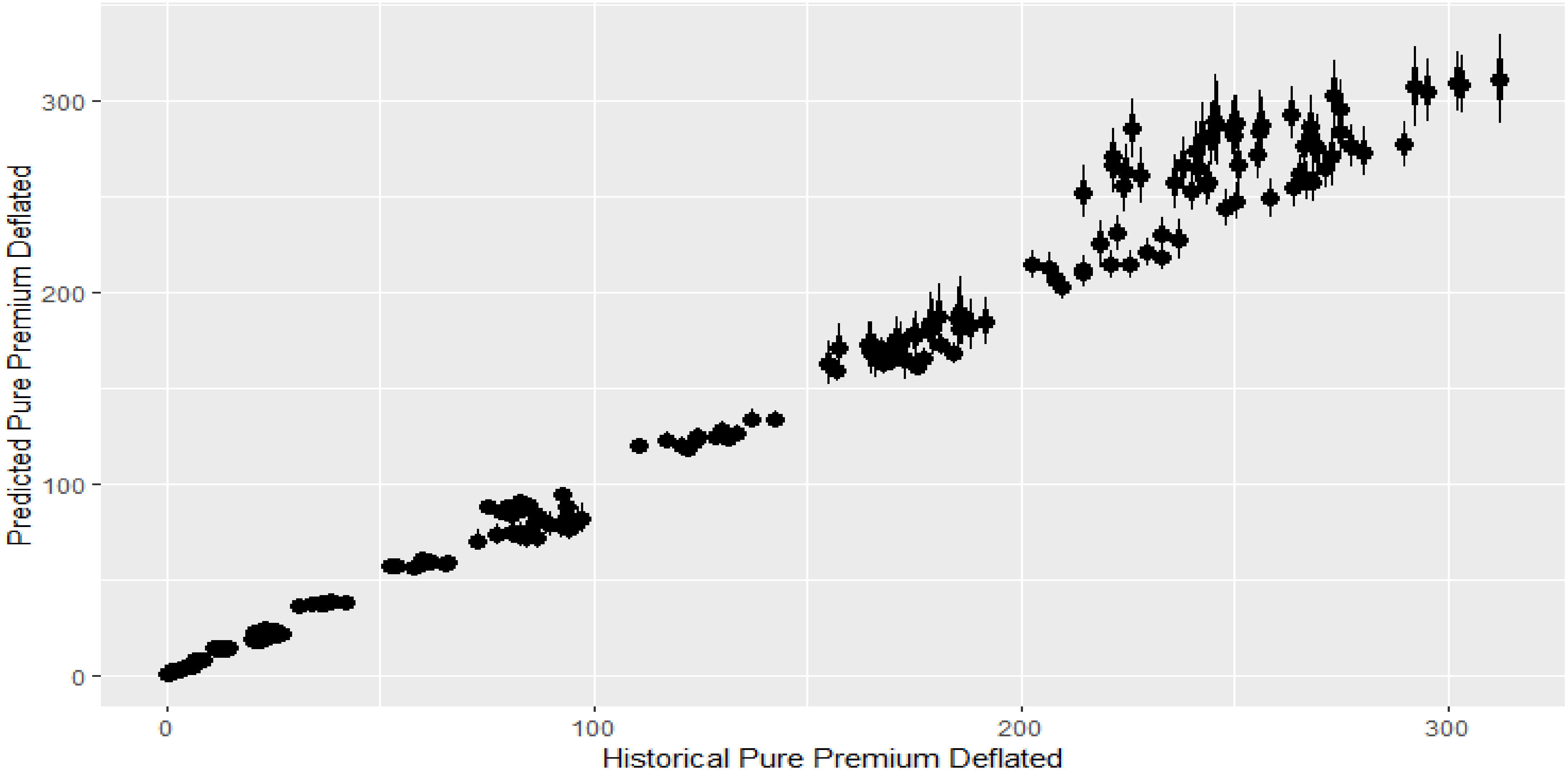
Polynomial Model 1 Results Summary

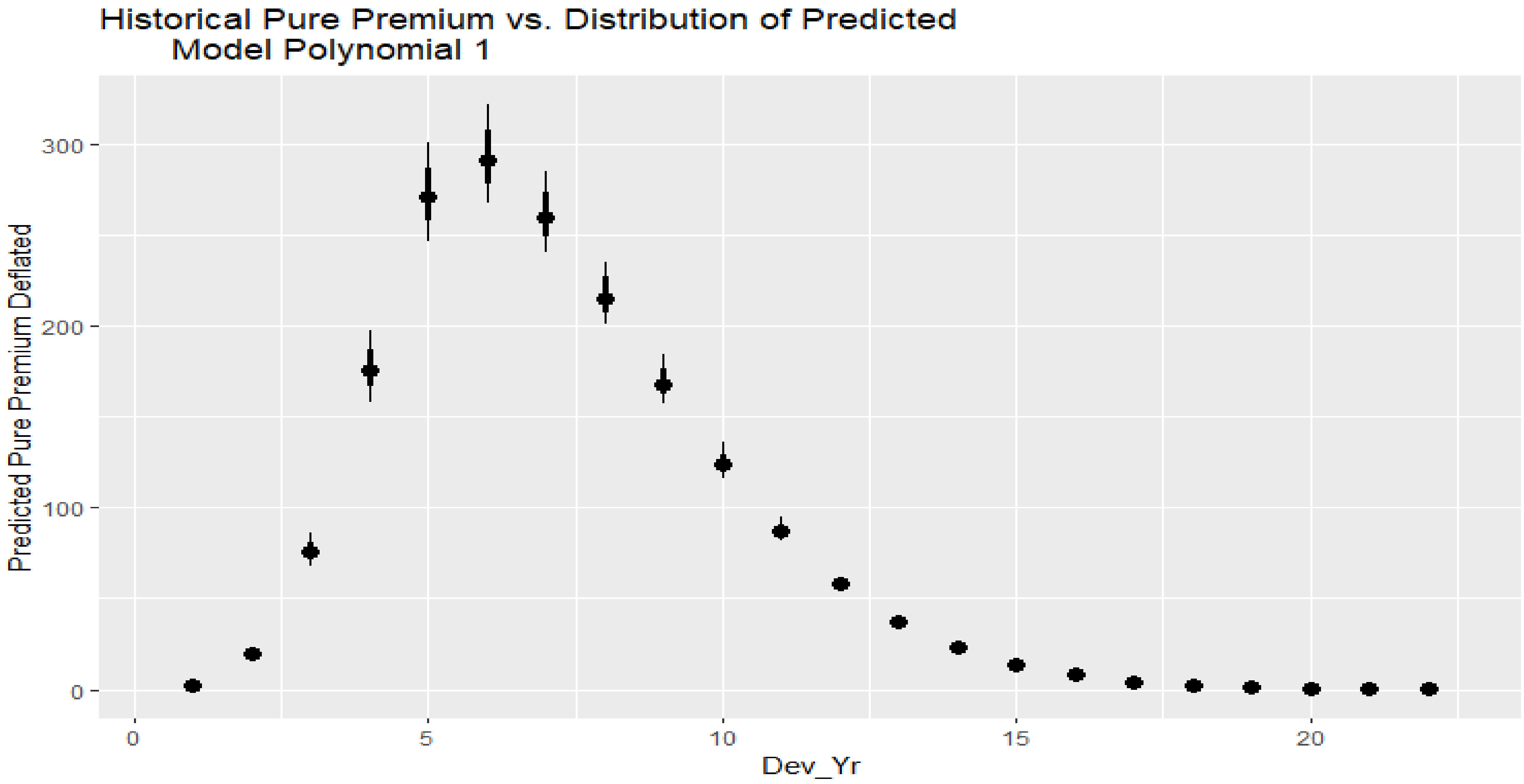
```
summary(Model_Polynomial_1)
 Family: lognormal
  Links: mu = identity; sigma = log
Formula: Trended_Incr_PP_Def ~ 0 + Intercept + Ln_Dev_Yr + Dev_Yr_6_Cap + Dev_Yr_6_Cap_Sqrd
        + Dev_Yr_6_Spline + Dev_Yr_6_Spline_Sqrd + (1 || Acc_Yr) + Cal_Yr_Time
         sigma ~ 0 + Intercept + Dev_Yr_10_Cap + Dev_Yr_10_Spline
   Data: Train_Triangle_All_Operation (Number of observations: 253)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Group-Level Effects:
~Acc_Yr (Number of levels: 22)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.03      0.01     0.01     0.05 1.00     2529     3532

Population-Level Effects:
                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept               -0.51      0.13    -0.77    -0.25 1.00     3371     4709
Ln_Dev_Yr                2.24      0.18     1.89     2.60 1.00     2466     4264
Dev_Yr_6_Cap             1.16      0.11     0.93     1.38 1.00     2153     3348
Dev_Yr_6_Cap_Sqrd       -0.14      0.01    -0.15    -0.12 1.00     2600     4169
Dev_Yr_6_Spline         -0.45      0.03    -0.50    -0.40 1.00     2296     3765
Dev_Yr_6_Spline_Sqrd    -0.01      0.00    -0.02    -0.01 1.00     3051     5357
Cal_Yr_Time              0.01      0.00     0.00     0.01 1.00     4143     5340
sigma_Intercept         -0.37      0.09    -0.53    -0.20 1.00     4437     5434
sigma_Dev_Yr_10_Cap     -0.30      0.01    -0.32    -0.27 1.00     3832     4614
sigma_Dev_Yr_10_Spline   0.22      0.02     0.17     0.27 1.00     5625     6240

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Bayesian MCMC performance summary terms

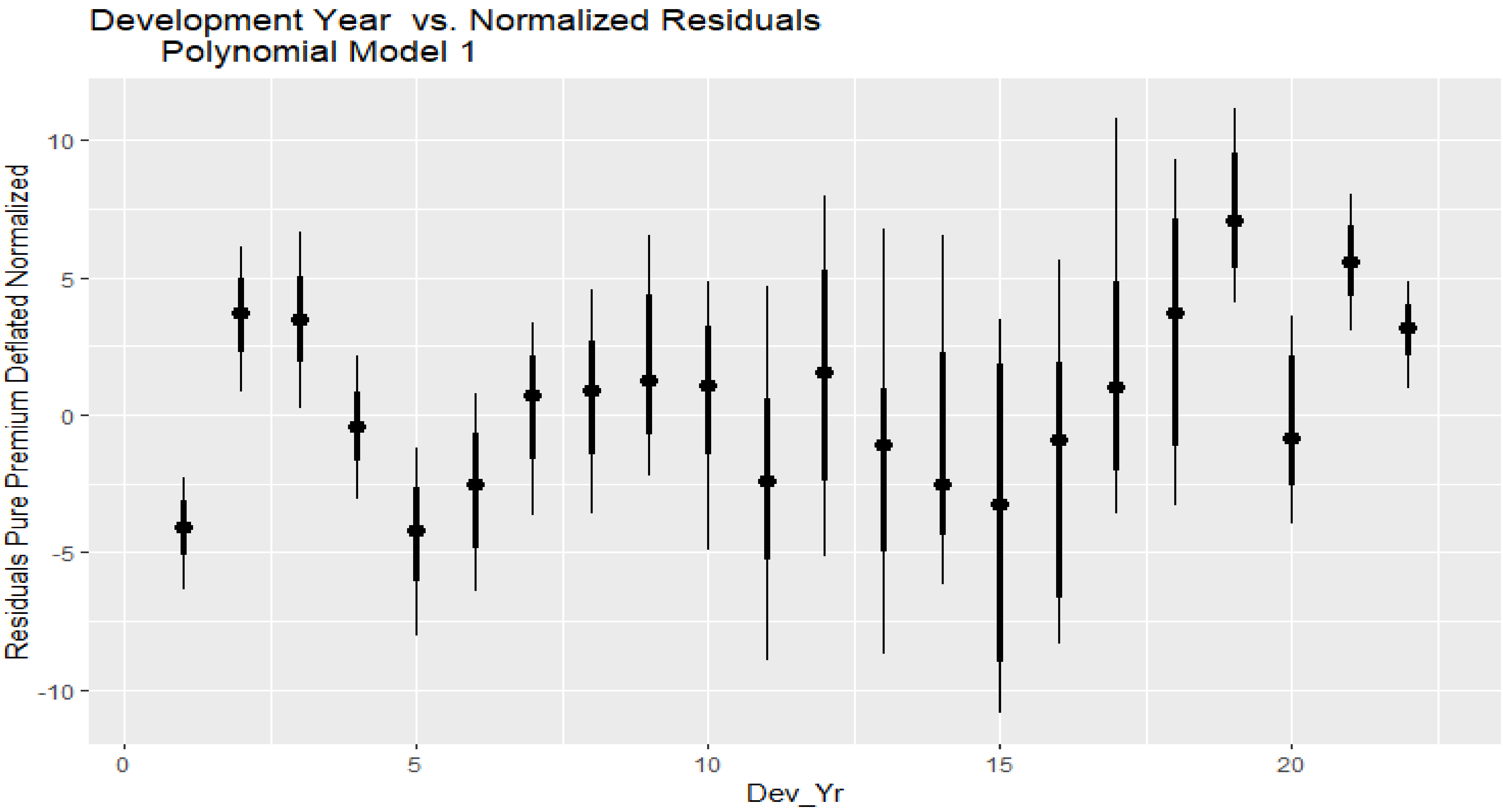
- Rhat: used to summarize parameter comparison between chains with 1.00 signifying the comparison is good.
- Bulk_ESS: Number of non-correlated iterations used to get measure of effective overall sample size for an estimate
- Tail_ESS: Number of non-correlated iterations used to get measure of effective sample size for an estimate in the tail of the distribution
- Group Level effects: displays standard deviation across groups used in least squares credibility weighting of mean result for a given group
- Population level effects: equivalent to GLM independent variables

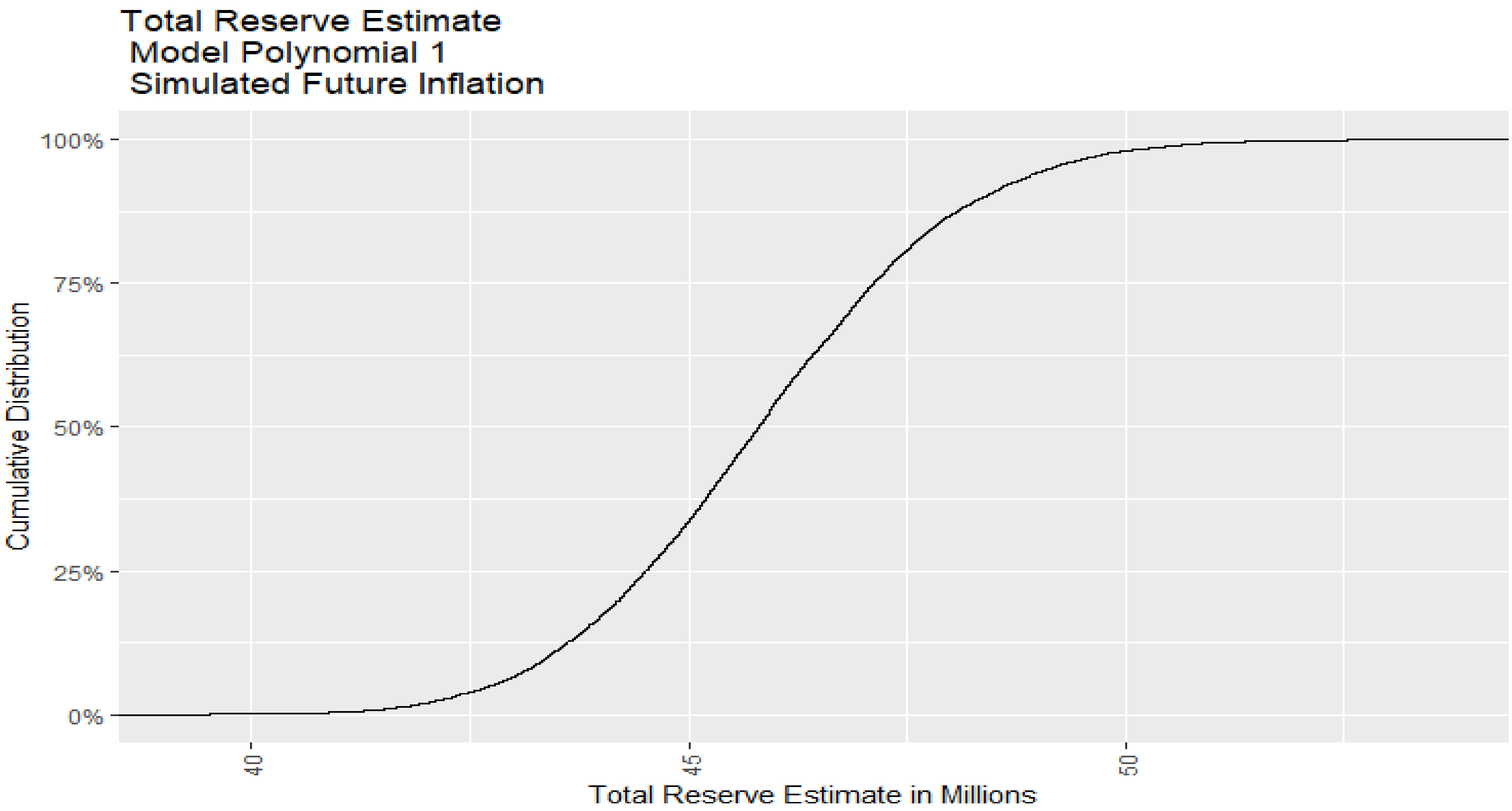
Historical Pure Premium vs. Distribution of Predicted Model Polynomial 1





Historical Pure Premium vs. Distribution of Predicted
Model Polynomial 1
Predicted Pure Premium Deflated
300
200
100
0
0
5
10
15
20
Dev_Yr


Development Year vs. Normalized Residuals
Polynomial Model 1
Residuals Pure Premium Deflated Normalized
10
5
0
-5
-10
0
5
10
15
20
Dev_Yr


Total Reserve Estimate
Model Polynomial 1
Simulated Future Inflation
100%
75%
50%
25%
0%
Cumulative Distribution
40
45
50
Total Reserve Estimate in Millions

Polynomial 2 Prior Definitions

```
Polynomial_2_prior <- c(prior(normal(.6,.2),class=b, coef= Dev_Yr_6_Cap),
                        prior(normal(-.2,.2),class=b, coef= Dev_Yr_6_Cap_Sqrd),
                        prior(normal(-.2,.2),class=b, coef= Dev_Yr_6_Spline),
                        prior(normal(-.2,.2),class=b, coef= Dev_Yr_6_Spline_Sqrd),
                        prior(normal(1,.25),class=b, coef=Dev_Yr_2_Factor2),
                        prior(normal(2,.25),class=b, coef=Dev_Yr_2_Factor3),
                        prior(normal(-1,.25),class=b ,coef =Intercept),
                        prior(normal(.02,.01),class=b, coef=Cal_Yr_Time),
                        prior(normal(-.01,.005),class=b, coef=Dev_Yr_10_Cap,dpar=sigma),
                        prior(student_t(2,.02,.01),class=b, coef=Dev_Yr_10_Spline,dpar=sigma))
```

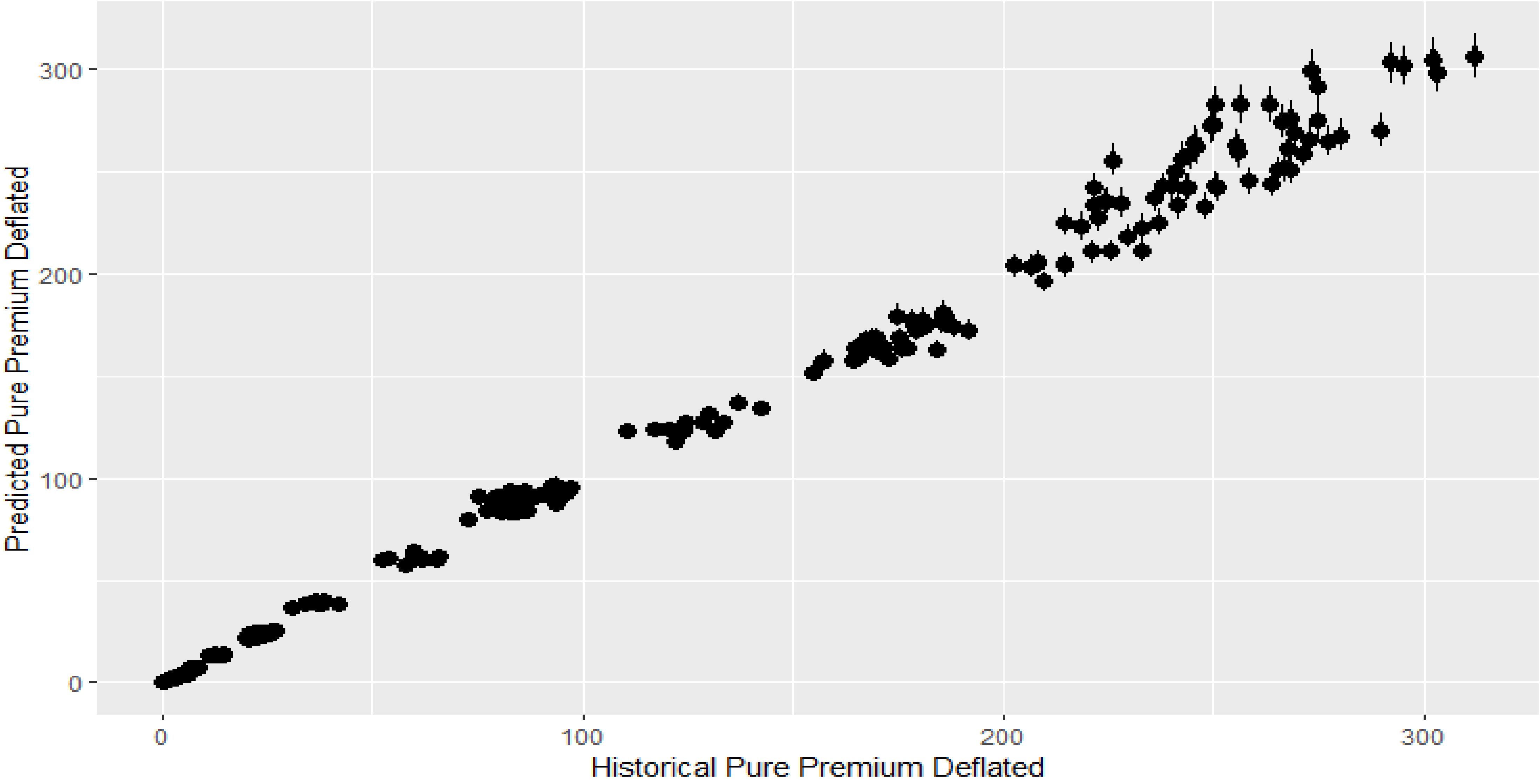
Polynomial Model 2 brms Instructions

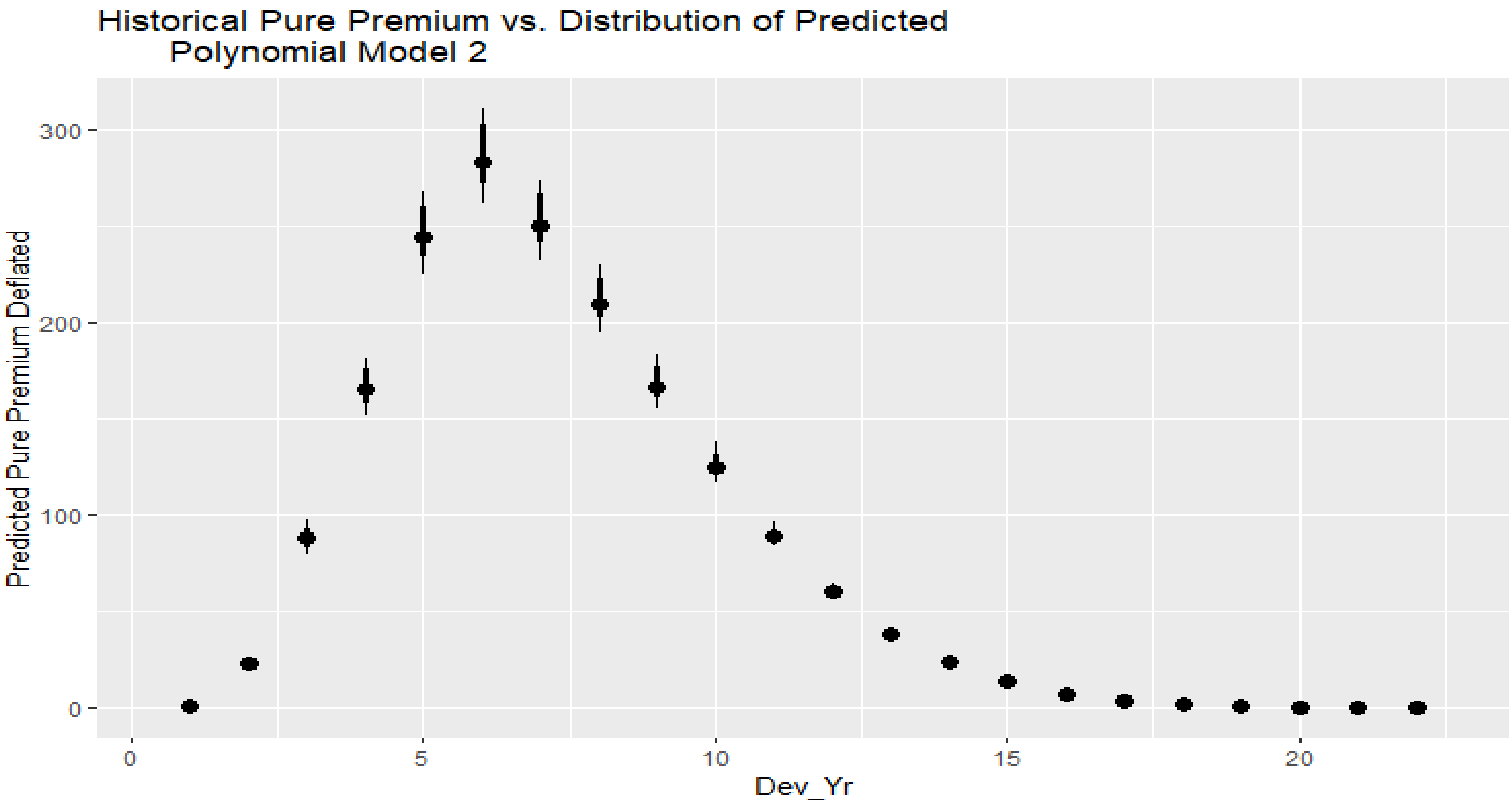
```
Model_Polynomial_2 <- brm(bf(Trended_Incr_PP_Def ~ 0 + Intercept + Dev_Yr_2_Factor +
                              Dev_Yr_6_Cap + Dev_Yr_6_Cap_Sqrd +Dev_Yr_6_Spline
                            + Dev_Yr_6_Spline_Sqrd+Cal_Yr_Time +(1||Acc_Yr),
                            sigma ~   Dev_Yr_10_Cap + Dev_Yr_10_Spline ),
                          iter = 4000,
                          prior= Polynomial_2_prior,
                          seed= 8603529,
                          data = Train_Triangle_All_Operation, family = lognormal())
```

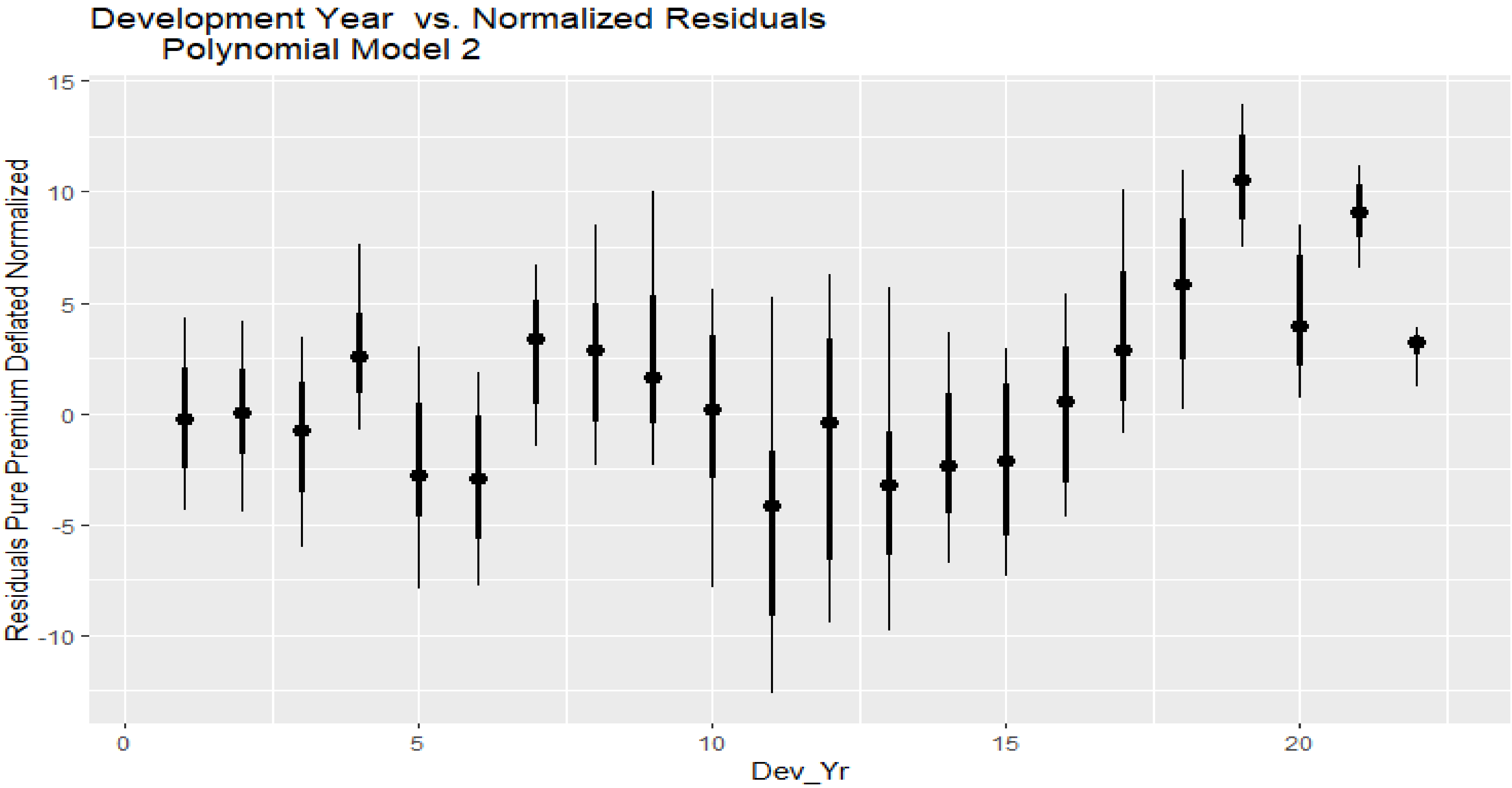
Polynomial Model 2 Results Summary

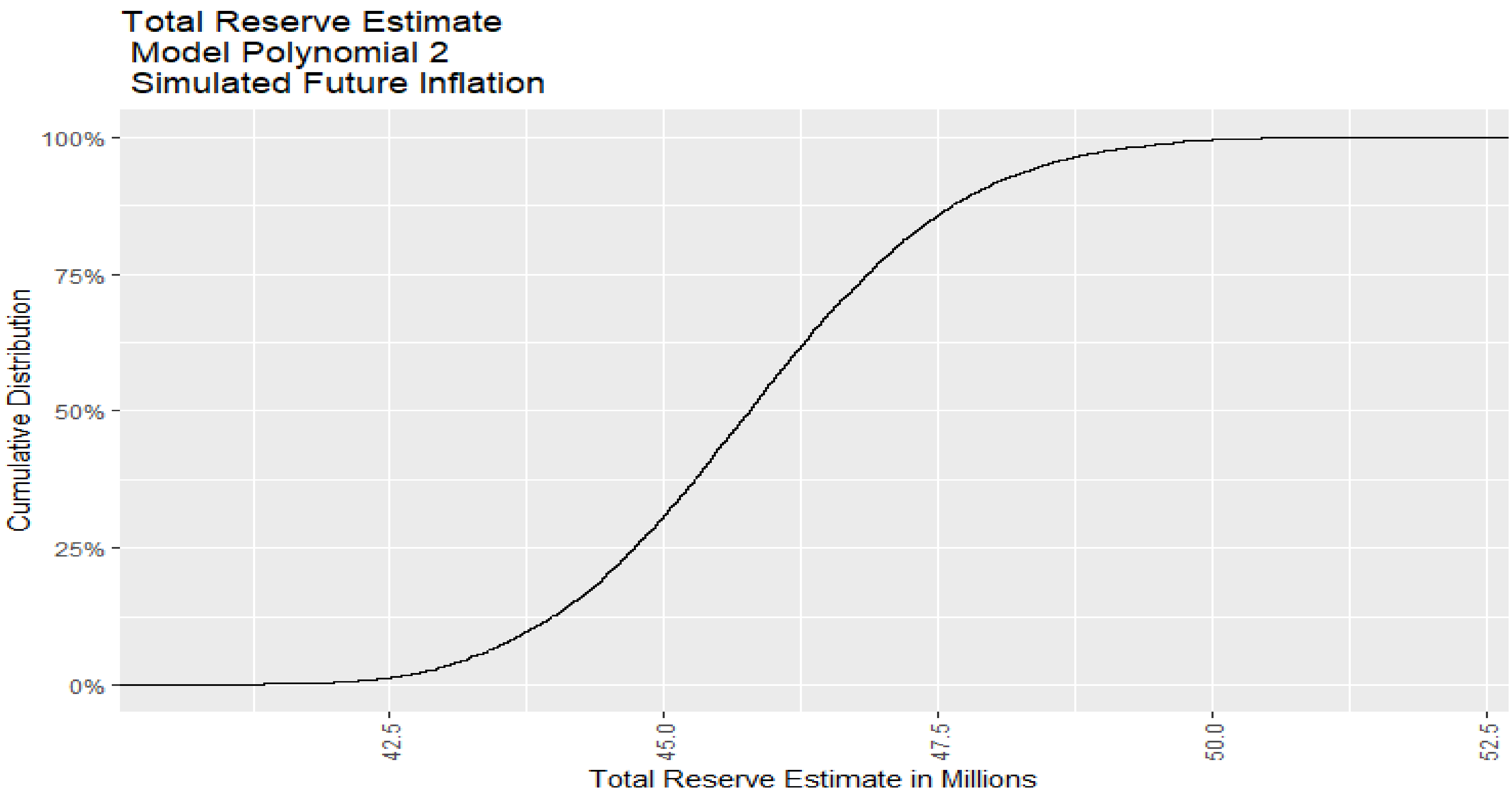
```
summary(Model_Polynomial_2)
 Family: lognormal
  Links: mu = identity; sigma = log
Formula: Trended_Incr_PP_Def ~ 0 + Intercept + Dev_Yr_2_Factor + Dev_Yr_6_Cap
        + Dev_Yr_6_Cap_Sqrd + Dev_Yr_6_Spline + Dev_Yr_6_Spline_Sqrd + Cal_Yr_Time + (1 || Acc_Yr)
         sigma ~ Dev_Yr_10_Cap + Dev_Yr_10_Spline
   Data: Train_Triangle_All_Operation (Number of observations: 253)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Group-Level Effects:
~Acc_Yr (Number of levels: 22)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.02      0.01     0.01     0.03 1.00     2671     2719

Population-Level Effects:
                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma_Intercept           -2.88      0.07    -3.00    -2.74 1.00     5753     5598
Intercept                 -1.22      0.05    -1.31    -1.12 1.00     2148     3641
Dev_Yr_2_Factor2           1.79      0.04     1.72     1.86 1.00     2288     3638
Dev_Yr_2_Factor3           2.26      0.06     2.14     2.39 1.00     2093     3217
Dev_Yr_6_Cap               1.47      0.05     1.37     1.57 1.00     1973     2856
Dev_Yr_6_Cap_Sqrd         -0.12      0.01    -0.13    -0.11 1.00     1974     2861
Dev_Yr_6_Spline           -0.10      0.01    -0.12    -0.09 1.00     4320     5297
Dev_Yr_6_Spline_Sqrd      -0.03      0.00    -0.03    -0.02 1.00     4383     5083
Cal_Yr_Time                0.01      0.00     0.01     0.01 1.00     5867     5854
sigma_Dev_Yr_10_Cap       -0.02      0.00    -0.03    -0.01 1.00    10244     6103
sigma_Dev_Yr_10_Spline     0.25      0.02     0.20     0.29 1.00     4987     5873

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Historical Pure Premium Deflated vs. Distribution of Predicted Polynomial Model 2




Historical Pure Premium vs. Distribution of Predicted
Polynomial Model 2
Predicted Pure Premium Deflated
300
200
100
0
0
5
10
15
20
Dev_Yr


Development Year vs. Normalized Residuals
Polynomial Model 2
Residuals Pure Premium Deflated Normalized
Dev_Yr
15
10
5
0
-5
-10
0
5
10
15
20


Total Reserve Estimate
Model Polynomial 2
Simulated Future Inflation
100%
75%
50%
25%
0%
Cumulative Distribution
42.5
45.0
47.5
50.0
52.5
Total Reserve Estimate in Millions

Comments on Multivariate Model Slides

- Excluded from model comparison
- Slides included to show some features of building a reserve model if you have to split it into counts and severity to get a decent fit
- Prior definitions
  - Split Poisson vs. Lognormal using “resp=“
  - Other components work as before
- Split the model formula using separate objects for counts and severity
  - Keeps the length of code for a given task readable
  - Bolt the model objects together using “+”

Multivariate Model Prior Definitions

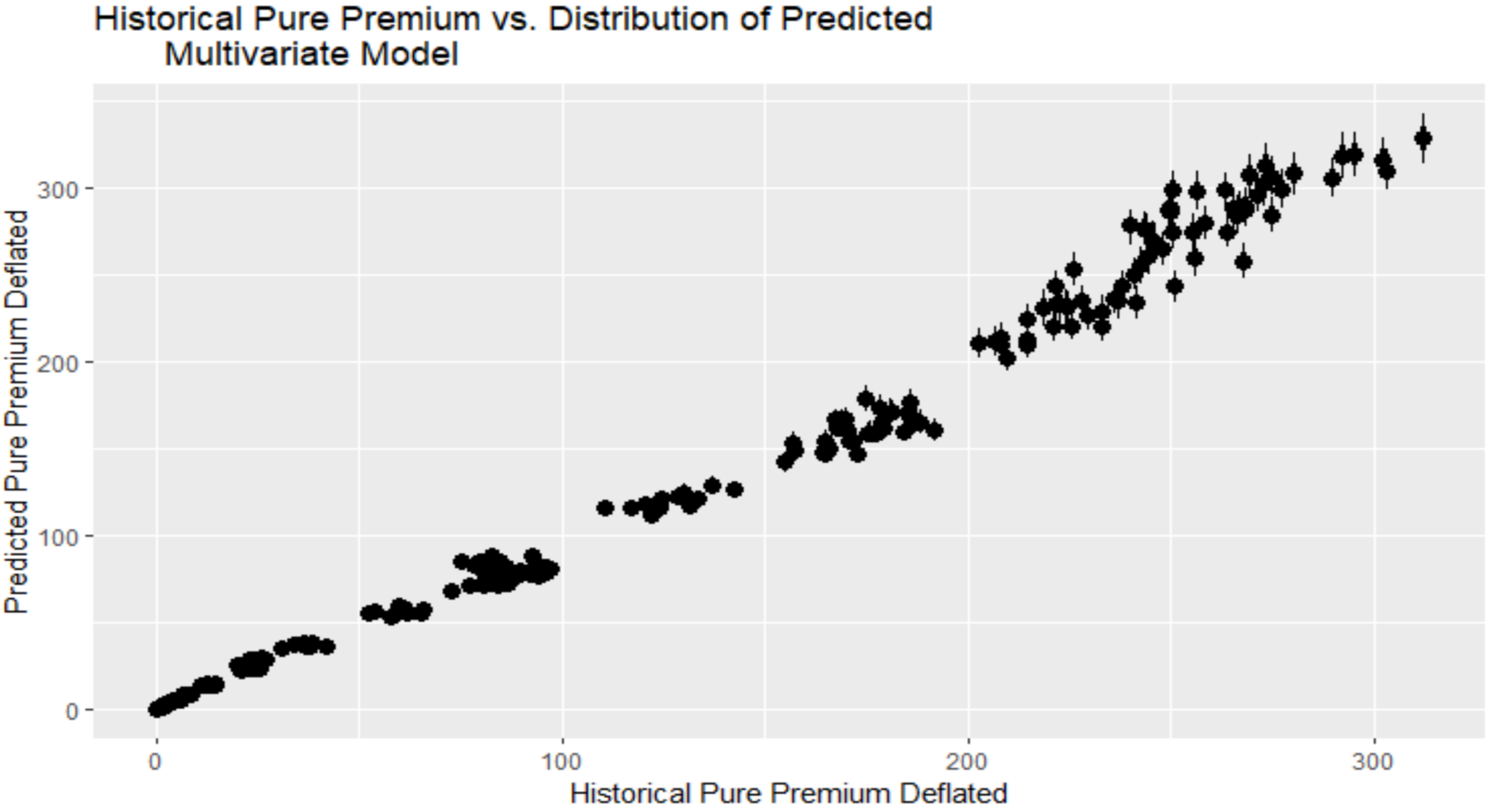
```
multivariate_prior <- c(prior(normal(.6,.2),class=b, coef= Dev_Yr_8_Cap, resp=TrendedMeanPaymentDef),
                       prior(normal(-.2,.2),class=b, coef= Dev_Yr_8_Cap_Sqrd,resp=TrendedMeanPaymentDef),
                       prior(normal(-.2,.2),class=b, coef= Dev_Yr_8_Spline,resp=TrendedMeanPaymentDef),
                       prior(normal(-.2,.2),class=b, coef= Dev_Yr_8_Spline_Sqrd,resp=TrendedMeanPaymentDef),
                       prior(normal(1,.25),class=b, coef=Dev_Yr_1_Factor2,resp=TrendedMeanPaymentDef),
                       prior(normal(1,.25),class=b ,coef =Intercept,resp=TrendedMeanPaymentDef),
                       prior(normal(.02,.01),class=b, coef=Cal_Yr_Time,resp=TrendedMeanPaymentDef),
                       prior(normal(-2,.5),class=b ,coef =Intercept,dpar=sigma,resp=TrendedMeanPaymentDef),
                       prior(normal(-.01,.005),class=b, coef=Dev_Yr_15_Cap,dpar=sigma,resp=TrendedMeanPaymentDef),
                       prior(normal(-1,.5),class=b, coef=Intercept,resp=PaidCnt),
                       prior(normal(.2,.1),class=b, coef=Dev_Yr_4_Cap,resp=PaidCnt),
                       prior(normal(1,.1),class=b, coef=Ln_Dev_Yr,resp=PaidCnt),
                       prior(normal(-.2,.1),class=b, coef=Dev_Yr_4_Spline,resp=PaidCnt))
```

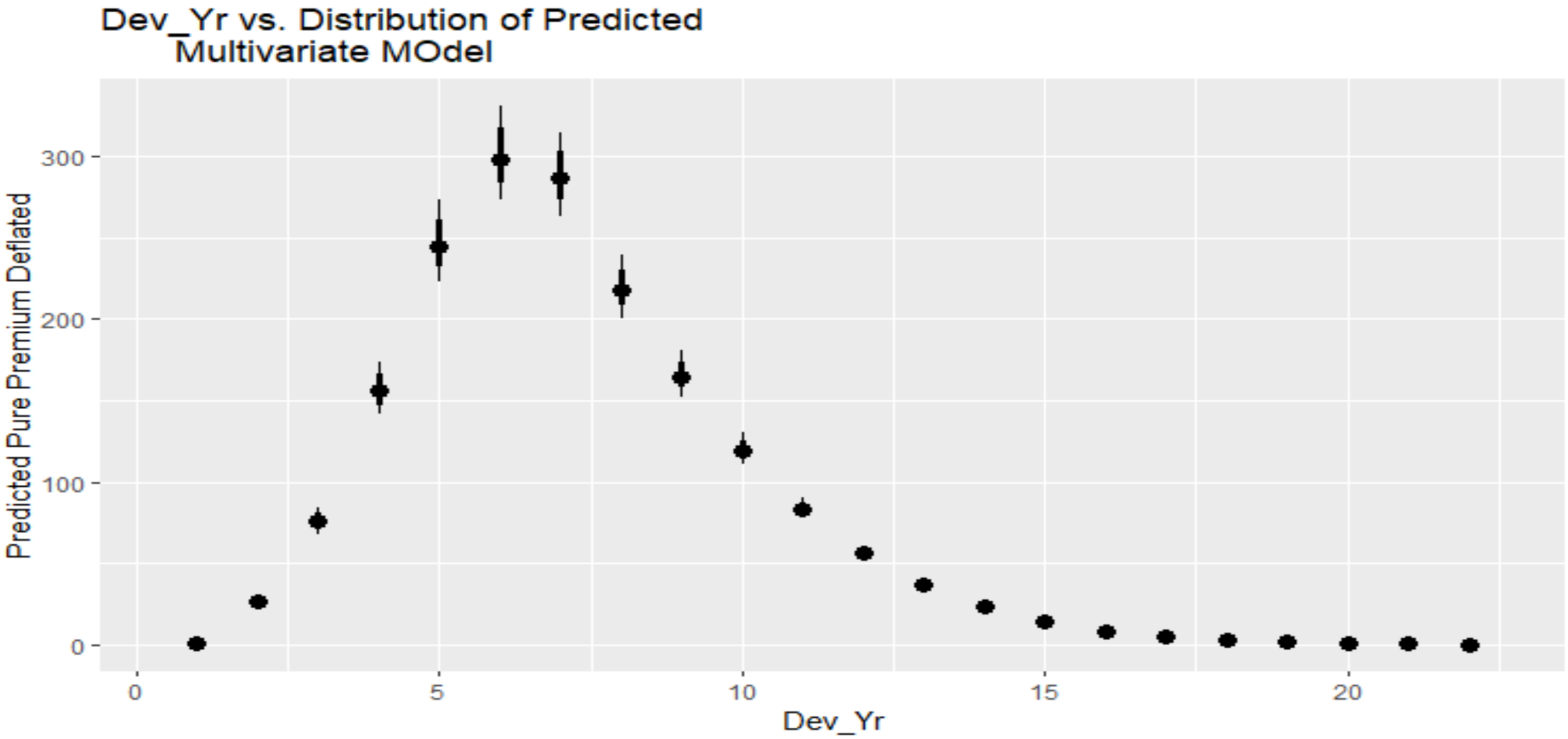
Multivariate Model brms Instructions

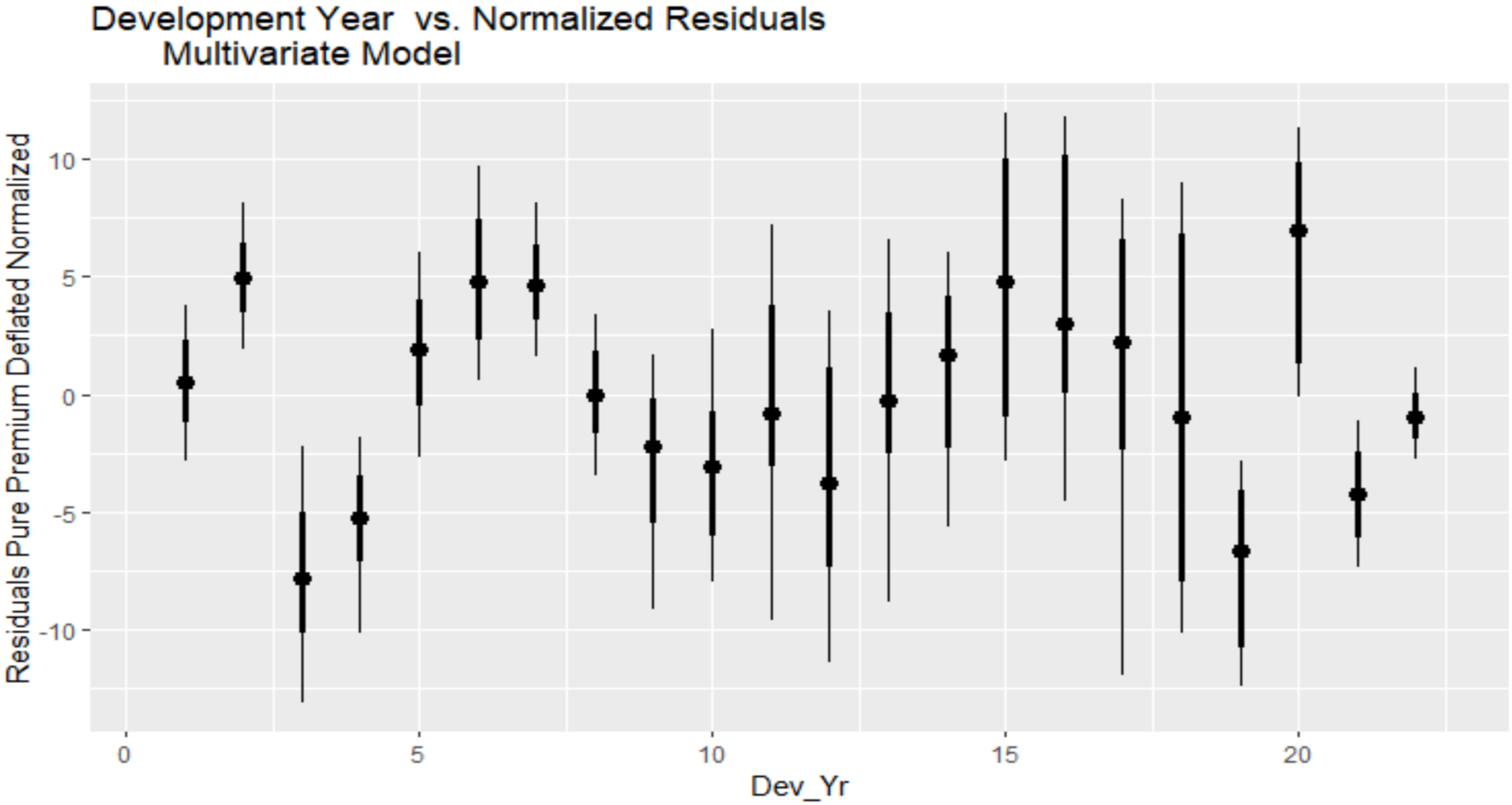
```
form_count <- bf(Paid_Cnt| rate(Rptd_Cnt) ~ 0 + Intercept
                  + Dev_Yr_4_Cap
                  + Ln_Dev_Yr
                  + Dev_Yr_4_Spline

                  + (1||Acc_Yr),   family = poisson())


form_sev <-bf(Trended_Mean_Payment_Def~ 0 + Intercept
               + Dev_Yr_8_Cap
               + Dev_Yr_8_Cap_Sqrd
               + Dev_Yr_8_Spline
               + Dev_Yr_8_Spline_Sqrd
               + Dev_Yr_1_Factor
               + Cal_Yr_Time
               + (1||Acc_Yr),
               sigma ~  0 + Intercept +Dev_Yr_15_Cap ,
               family =lognormal() )

Model_Multivariate <- brm(form_sev + form_count ,
                          prior=multivariate_prior,
                          data = Train_Triangle_All_Operati
```

Multivariate Model Results Summary

```
 Family: MV(lognormal, poisson)
  Links: mu = identity; sigma = log
         mu = log
Formula: Trended_Mean_Payment_Def ~ 0 + Intercept + Dev_Yr_8_Cap + Dev_Yr_8_Cap_Sqrd + Dev_Yr_8_Spline
         + Dev_Yr_8_Spline_Sqrd + Dev_Yr_1_Factor + Cal_Yr_Time + (1 || Acc_Yr)
         sigma ~ 0 + Intercept + Dev_Yr_15_Cap
         Paid_Cnt | rate(Rptd_Cnt) ~ 0 + Intercept + Dev_Yr_4_Cap + Ln_Dev_Yr + Dev_Yr_4_Spline + (1 || Acc_Y
   Data: Train_Triangle_All_Operation (Number of observations: 253)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
~Acc_Yr (Number of levels: 22)
                                    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(TrendedMeanPaymentDef_Intercept)     0.01      0.01     0.00     0.03 1.00     1533     1789
sd(PaidCnt_Intercept)                   0.02      0.01     0.01     0.03 1.00     1556     2187

Population-Level Effects:
                                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
TrendedMeanPaymentDef_Intercept                0.04      0.03    -0.01     0.10 1.00     4628     3231
TrendedMeanPaymentDef_Dev_Yr_8_Cap             1.48      0.02     1.44     1.52 1.00     3322     3061
TrendedMeanPaymentDef_Dev_Yr_8_Cap_Sqrd       -0.11      0.00    -0.11    -0.10 1.00     3347     3171
TrendedMeanPaymentDef_Dev_Yr_8_Spline         -0.09      0.01    -0.11    -0.08 1.00     6696     3244
TrendedMeanPaymentDef_Dev_Yr_8_Spline_Sqrd    -0.01      0.00    -0.01    -0.01 1.00     6412     3328
TrendedMeanPaymentDef_Dev_Yr_1_Factor2         1.48      0.04     1.41     1.55 1.00     4355     3272
TrendedMeanPaymentDef_Cal_Yr_Time              0.01      0.00     0.01     0.01 1.00     5502     3292
sigma_TrendedMeanPaymentDef_Intercept         -2.37      0.05    -2.47    -2.26 1.00     5246     2927
sigma_TrendedMeanPaymentDef_Dev_Yr_15_Cap     -0.02      0.00    -0.03    -0.01 1.00     5743     3437
PaidCnt_Intercept                             -0.99      0.02    -1.02    -0.96 1.00     2647     2966
PaidCnt_Dev_Yr_4_Cap                          -0.29      0.02    -0.32    -0.26 1.00     2438     2788
PaidCnt_Ln_Dev_Yr                              0.97      0.03     0.91     1.03 1.00     2509     2677
PaidCnt_Dev_Yr_4_Spline                       -0.29      0.00    -0.30    -0.29 1.00     2649     2834

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```


Historical Pure Premium vs. Distribution of Predicted
Multivariate Model
Predicted Pure Premium Deflated
0
100
200
300
Historical Pure Premium Deflated
0
100
200
300


Dev_Yr vs. Distribution of Predicted
Multivariate MOdel
Predicted Pure Premium Deflated
300
200
100
0
0
5
10
15
20
Dev_Yr



Development Year vs. Normalized Residuals
Multivariate Model
Residuals Pure Premium Deflated Normalized
10
5
0
-5
-10
0
5
10
15
20
Dev_Yr

Nonlinear Model Prior Definitions

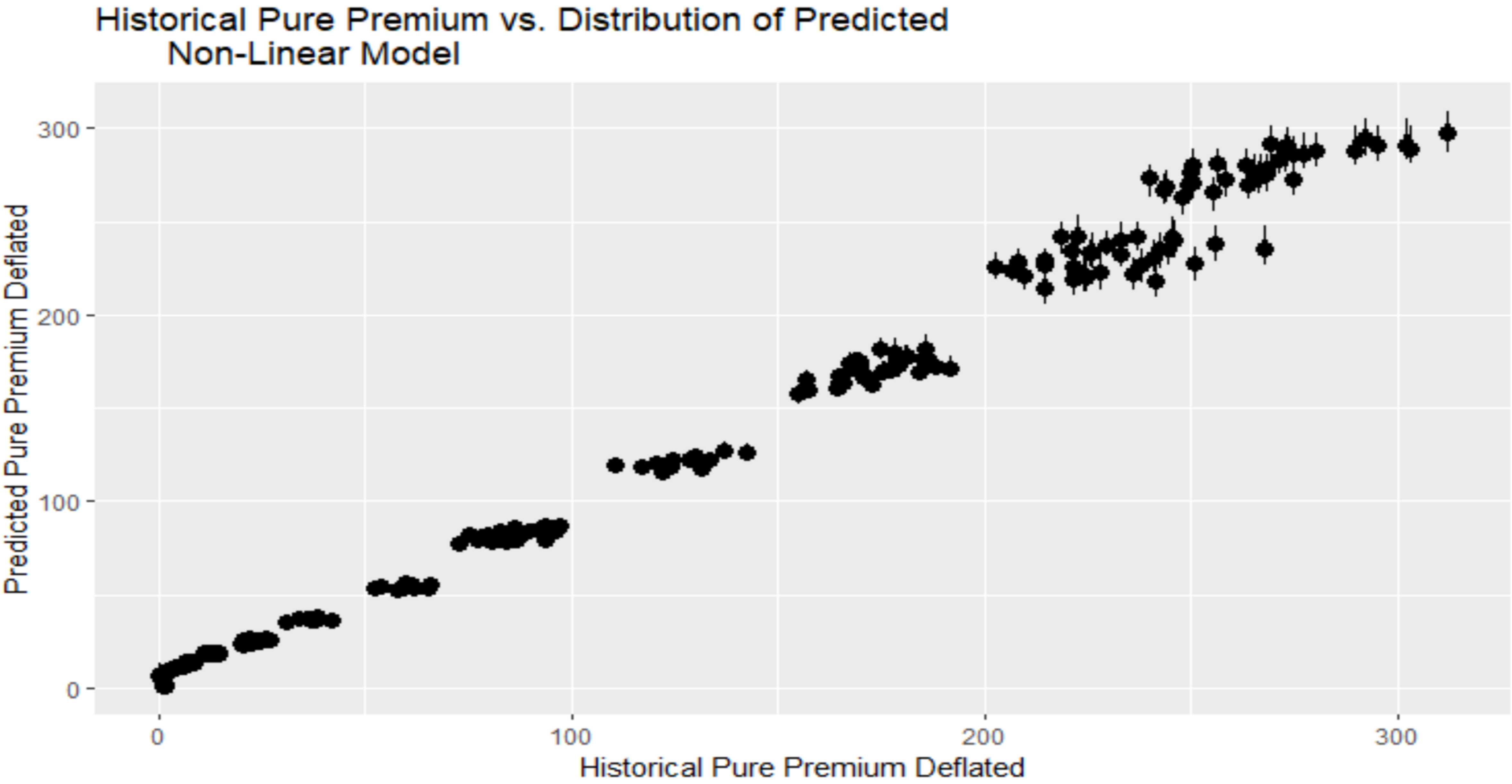
```
Non_linear_prior_4 <- c(prior(normal(-.1,.05),class=b, nlpar=b1,ub=0),
                        prior(normal(1.5,.25),class=b, nlpar=b2,),
                        prior(normal(-.1,.05),class=b, nlpar=b3,ub=0),
                        prior(normal(-.1,.05),class=b, nlpar=b4,ub=0),
                        prior(normal(1,.5),class=b, nlpar=b5,ub=2),
                        prior(normal(.02,.02),class=b, nlpar=infl,lb=0),
                        prior(normal(-1,.1),class=b, nlpar=dev1,ub =-.4),
                        prior(normal(-2,.1),class=b ,coef =Intercept, dpar=sigma),
                        prior(normal(-.01,.005),class=b, coef=Dev_Yr_10_Cap,dpar=sigma),
                        prior(student_t(3,.05,.01),class=b, coef=Dev_Yr_10_Spline_Ln,dpar=sigma))
```

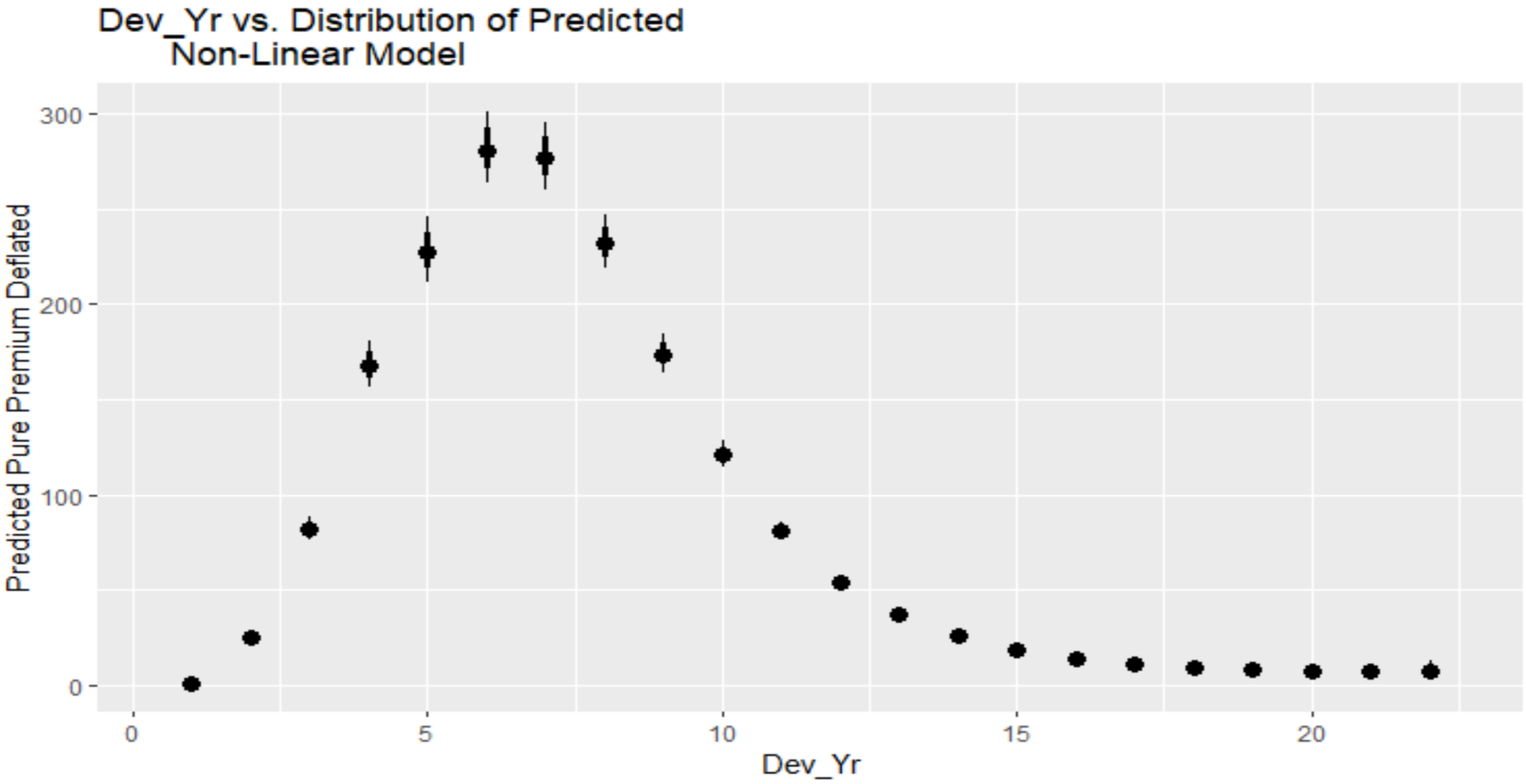
Nonlinear Model brms Instructions

```
Model_NL_4 <- brm(
  bf(Trended_Incr_PP_Def ~
       ( exp(dev1 + b1 *Dev_Yr_5_Cap
             + b2*Ln_Dev_Yr
             + b3 * Dev_Yr_5_Spline
             + b4 * Dev_Yr_5_Spline_Sqrd
             + b5 * Dev_Yr_GT_1
       )
       +infl*Cal_Yr_Time),

     b1 +b2 +b3 + +b4+ b5 +infl ~ 1,dev1 ~ 1 +(1||Acc_Yr),
     sigma ~  0 + Intercept +Dev_Yr_10_Spline_Ln +Dev_Yr_10_Cap,
     nl = TRUE),
  data = Train_Triangle_All_Operation, family = lognormal(),
  prior= Non_linear_prior_4,
  seed= 8603529,
  iter= 4000,
  control = list(adapt_delta = 0.99,
                 max_treedepth =12))
```

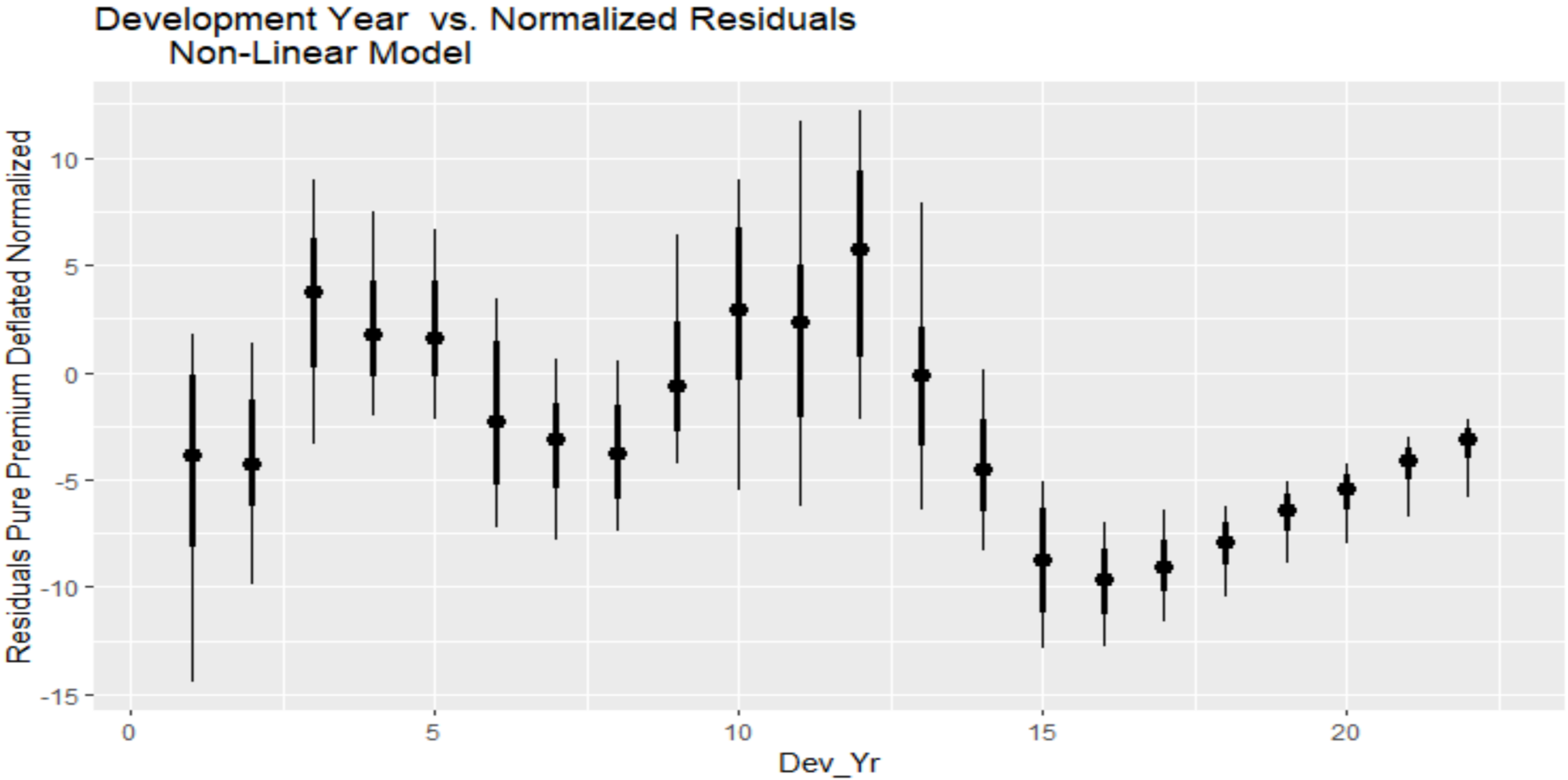
Nonlinear Model Results Summary

```
 Family: lognormal
  Links: mu = identity; sigma = log
Formula: Trended_Incr_PP_Def ~ (exp(dev1 + b1 * Dev_Yr_5_Cap + b2 * Ln_Dev_Yr + b3 * Dev_Yr_5_Spline
         + b4 * Dev_Yr_5_Spline_Sqrd + b5 * Dev_Yr_GT_1) + infl * Cal_Yr_Time)
         b1 ~ 1
         b2 ~ 1
         b3 ~ 1
         b4 ~ 1
         b5 ~ 1
         infl ~ 1
         dev1 ~ 1 + (1 || Acc_Yr)
         sigma ~ 0 + Intercept + Dev_Yr_10_Spline_Ln + Dev_Yr_10_Cap
   Data: Train_Triangle_All_Operation (Number of observations: 253)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Group-Level Effects:
~Acc_Yr (Number of levels: 22)
                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(dev1_Intercept)     0.00      0.00     0.00     0.01 1.00     2556     3119

Population-Level Effects:
                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
b1_Intercept                 -0.27      0.01    -0.29    -0.24 1.00     3385     4533
b2_Intercept                  1.45      0.03     1.38     1.52 1.00     3280     4305
b3_Intercept                 -0.23      0.00    -0.24    -0.22 1.00     3224     4263
b4_Intercept                 -0.00      0.00    -0.00    -0.00 1.00     6008     3723
b5_Intercept                  1.93      0.05     1.83     2.00 1.00     4674     3548
infl_Intercept                0.01      0.00     0.00     0.01 1.00     4563     3458
dev1_Intercept               -1.26      0.05    -1.33    -1.15 1.00     4566     3762
sigma_Intercept              -2.50      0.06    -2.60    -2.38 1.00     7136     5925
sigma_Dev_Yr_10_Spline_Ln     1.19      0.06     1.08     1.31 1.00     7106     5937
sigma_Dev_Yr_10_Cap          -0.02      0.00    -0.03    -0.01 1.00     9890     6757

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

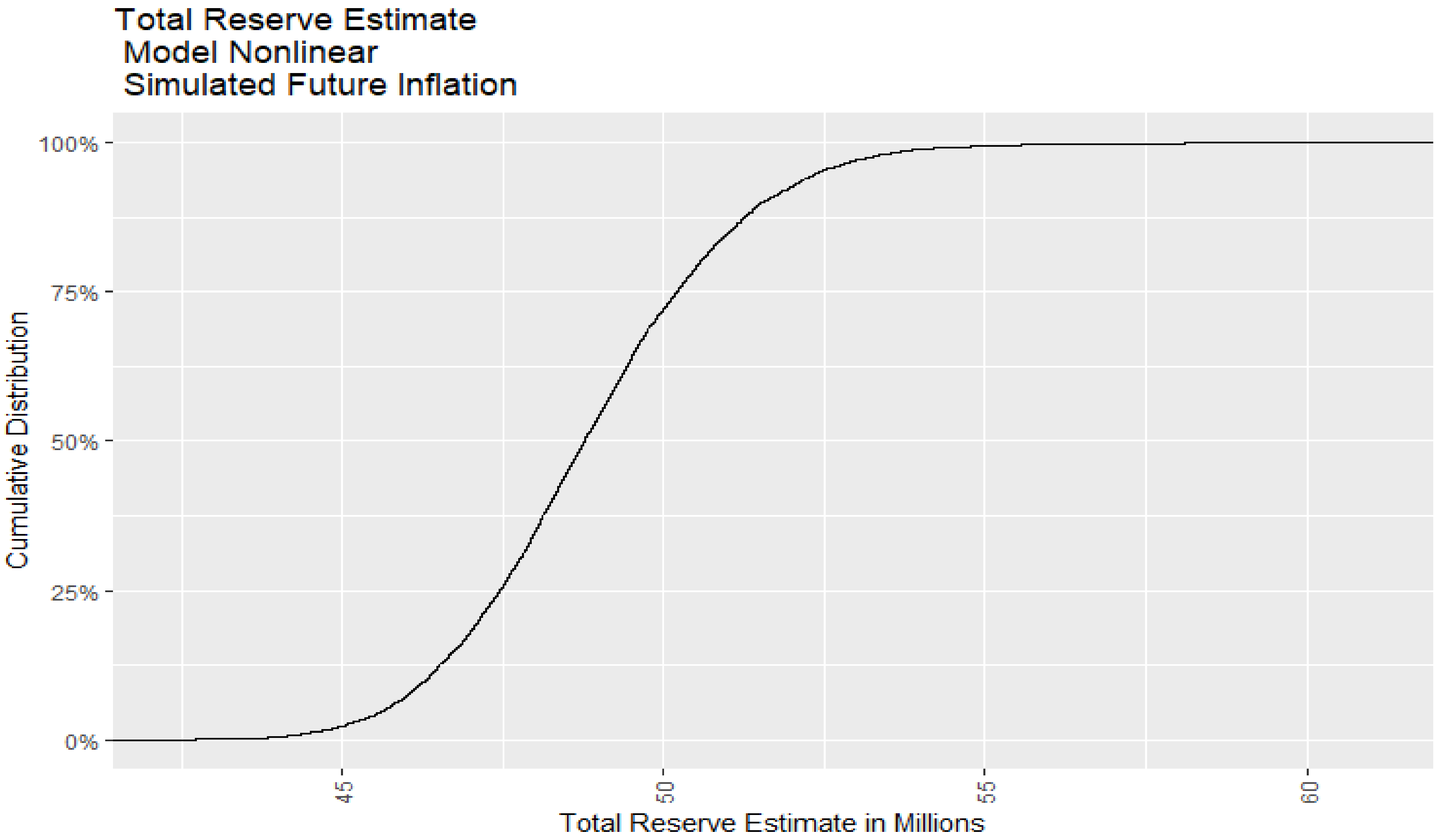

Historical Pure Premium vs. Distribution of Predicted
Non-Linear Model
Predicted Pure Premium Deflated
Historical Pure Premium Deflated
0
100
200
300
0
100
200
300



Dev_Yr vs. Distribution of Predicted
Non-Linear Model
Predicted Pure Premium Deflated
300
200
100
0
0
5
10
15
20
Dev_Yr



Development Year vs. Normalized Residuals
Non-Linear Model
Residuals Pure Premium Deflated Normalized
10
5
0
-5
-10
-15
0
5
10
15
20
Dev_Yr


Total Reserve Estimate
Model Nonlinear
Simulated Future Inflation
100%
75%
50%
25%
0%
Cumulative Distribution
45
50
55
60
Total Reserve Estimate in Millions

Checking Prior Distribution for Plausibility

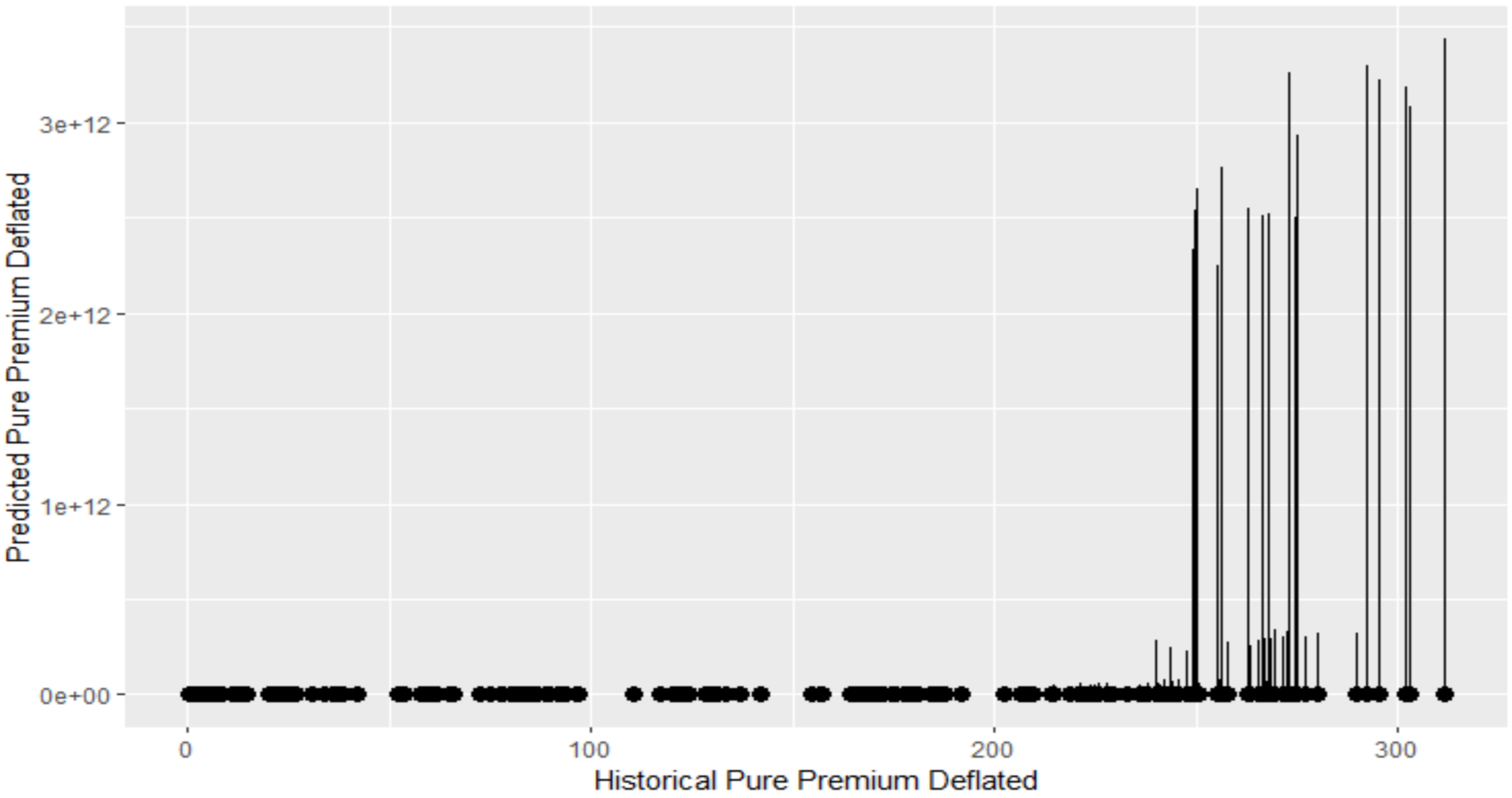
- Generally, a good practice to simulate prior distributions
  - Essential for some model forms and small data sets
  - Sometimes, just checking mean parameters in EXCEL is enough
  - In larger data sets, prior distribution effects may not matter
- Brms offers option to simulate population level prior distributions
- Example follows to show how software today reduces time burden to create Bayesian MCMC models

Nonlinear Model brms Instructions (Use Only Prior Distributions)

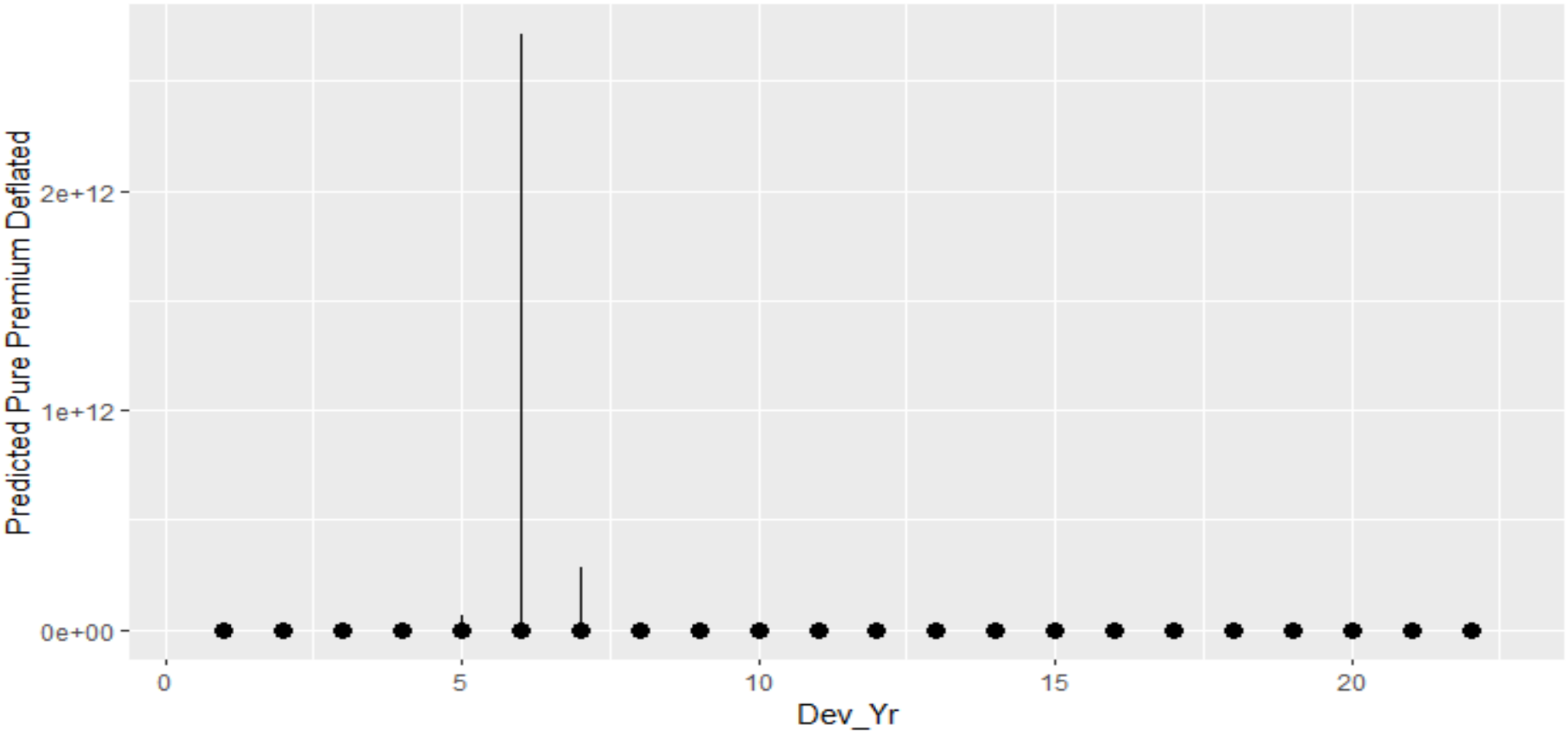
```
Model_NL_4_prior <- brm(
  bf(Trended_Incr_PP_Def ~
      ( exp(dev1 + b1 *Dev_Yr_5_Cap
           + b2*Ln_Dev_Yr
           + b3 * Dev_Yr_5_Spline
           + b4 * Dev_Yr_5_Spline_Sqrd
           + b5 * Dev_Yr_GT_1
      )
      +infl*Cal_Yr_Time),

    b1 +b2 +b3 + +b4+ b5 +infl ~ 1,dev1 ~ 1 ,
    sigma ~  0 + Intercept +Dev_Yr_10_Spline_Ln +Dev_Yr_10_Cap,
    nl = TRUE),
  sample_prior = "only",
  data = Train_Triangle_All_Operation, family = lognormal(),
  save_pars = save_pars(all=TRUE),
  prior= Non_linear_prior_4,
  seed= 8603529,
  iter= 4000,
  control = list(adapt_delta = 0.99,
              max_treedepth =12))
```

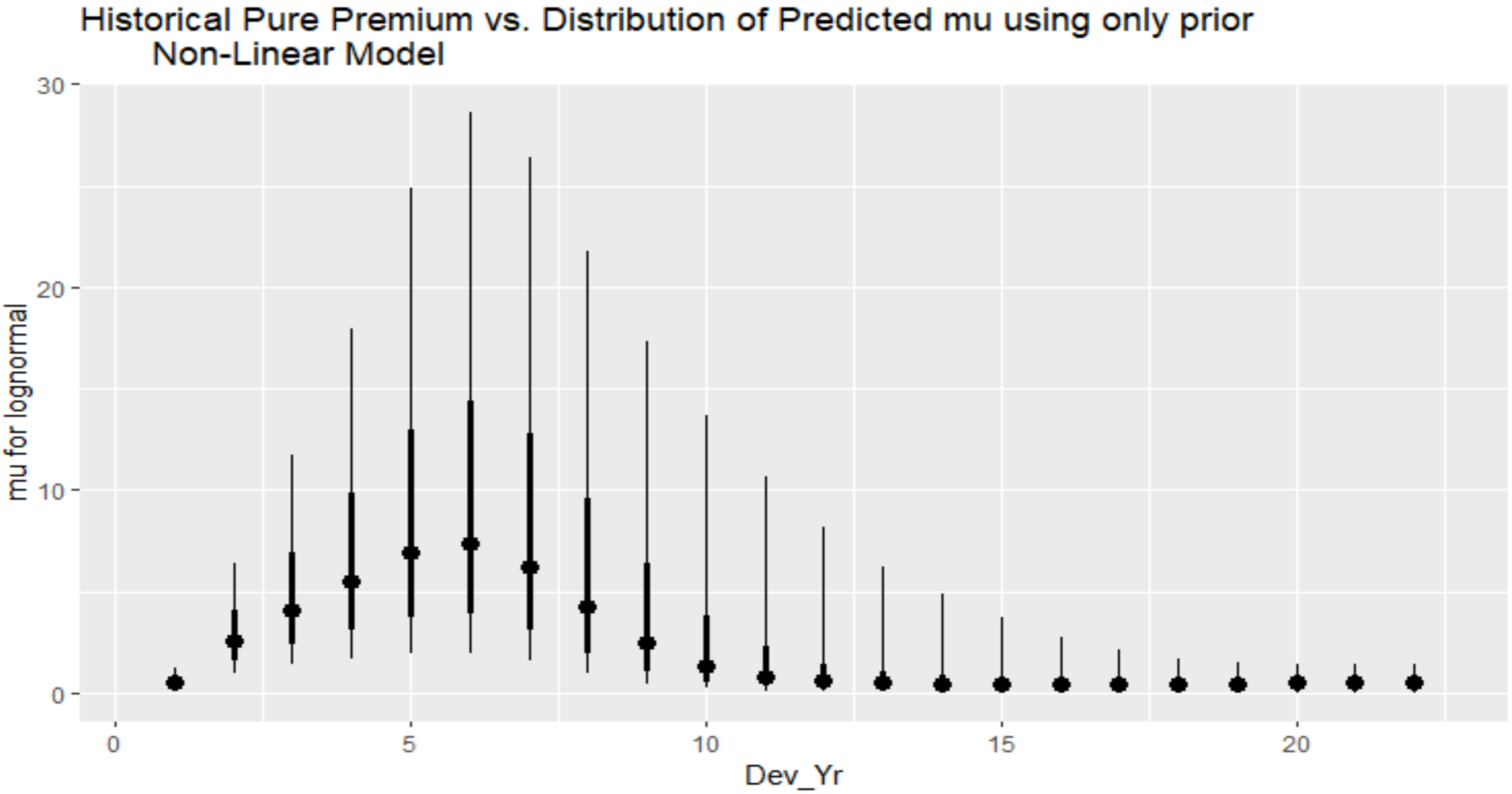
Include statement "sample_prior ="only"
Drop random effects (group) component of variable dev1

Historical Pure Premium vs. Distribution of Predicted Using Only Prior Non-Linear Model

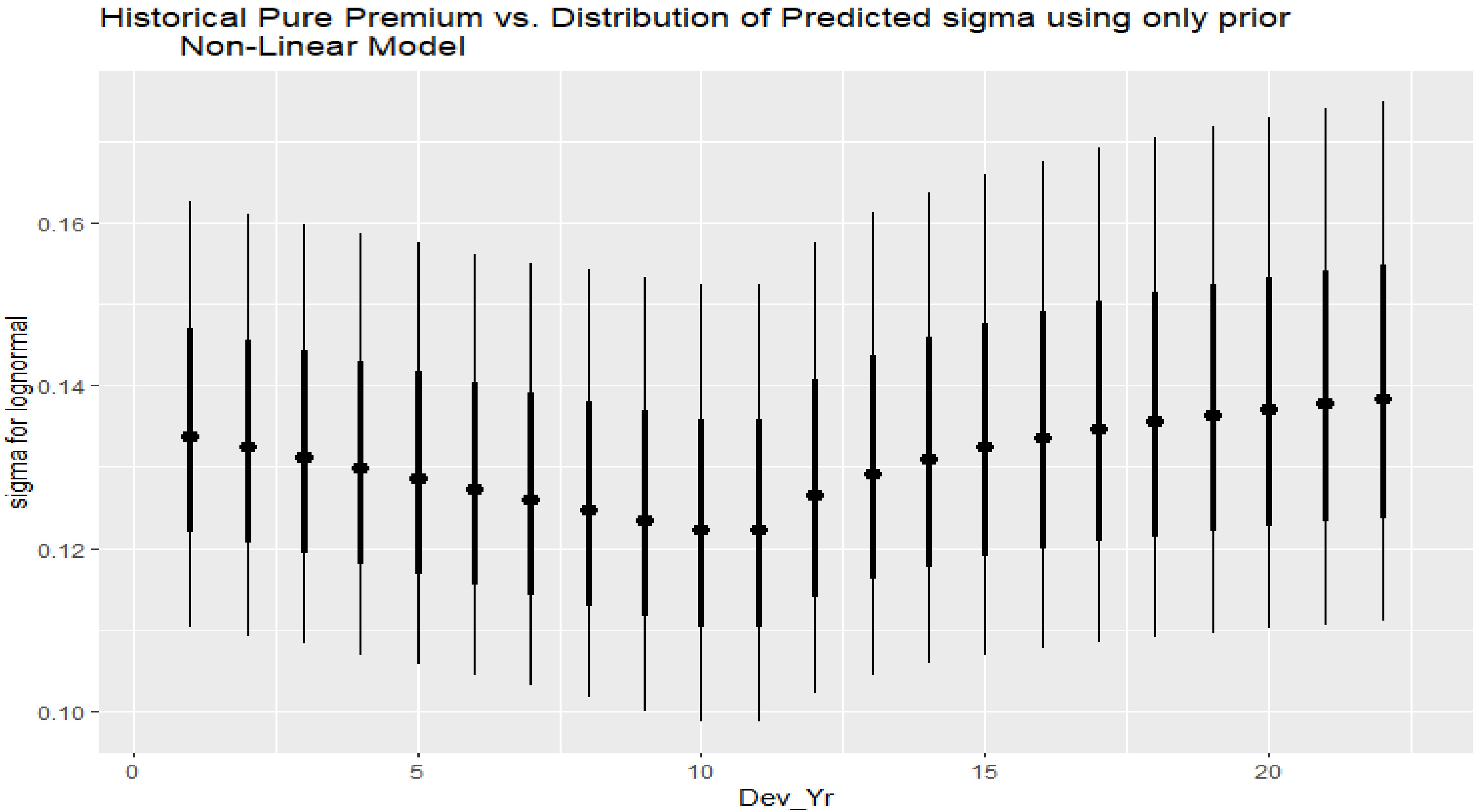


Dev_Yr vs. Distribution of Predicted Using Only Prior
Non-Linear Model





Historical Pure Premium vs. Distribution of Predicted mu using only prior
Non-Linear Model
30
20
10
0
mu for lognormal
0
5
10
15
20
Dev_Yr



Historical Pure Premium vs. Distribution of Predicted sigma using only prior
Non-Linear Model
sigma for lognormal
0.16
0.14
0.12
0.10
0
5
10
15
20
Dev_Yr

Comments on Generalized Additive Model Example

- Excluded from list of models to be compared given unresolved divergent error message
- Included in presentation:
  - Illustrates STAN/brms does give useful feedback on MCMC integrity
  - Provides illustration of bayesplot and ShinyStan diagnostics
  - Demonstrates why one has to pay attention to warnings
- Bayesplot is a package one can invoke within Rstudio with a range of diagnostics
- ShinyStan is a package one can launch which produces a set of prepackaged diagnostics

Generalized Additive Model Prior Definitions

```
GAM_prior <- c(prior(normal(.02,.01),class=b, coef=Cal_Yr_Time),
               prior(normal(0,.25),class=b, coef=Intercept),
               prior(normal(-2,.5),class=b ,coef =Intercept, dpar=sigma),
              prior(normal(-.02,.01),class=b, coef=Dev_Yr_10_Cap,dpar=sigma),
              prior(student_t(3,.02,.01),class=b, coef=Dev_Yr_10_Spline,dpar=sigma))
```

```
prior_summary(Model_GAM)
                   prior class                 coef   group resp  dpar nlpar lb ub       source
                  (flat)     b                                                          default
        normal(0.02, 0.01)     b          Cal_Yr_Time                                       user
           normal(0, 0.25)     b            Intercept                                       user
                  (flat)     b            sDev_Yr_1                                 (vectorized)
                  (flat)     b                                 sigma                    default
       normal(-0.02, 0.01)     b        Dev_Yr_10_Cap          sigma                       user
   student_t(3, 0.02, 0.01)     b     Dev_Yr_10_Spline          sigma                       user
           normal(-2, 0.5)     b            Intercept          sigma                       user
     student_t(3, 0, 2.5)    sd                                              0          default
     student_t(3, 0, 2.5)    sd                      Acc_Yr                  0     (vectorized)
     student_t(3, 0, 2.5)    sd            Intercept Acc_Yr                  0     (vectorized)
     student_t(3, 0, 2.5)   sds                                              0          default
     student_t(3, 0, 2.5)   sds s(Dev_Yr, k = 3, m = 2)                      0     (vectorized)
```

Complete set of prior specifications including defaults

Generalized Additive Model brms Instructions

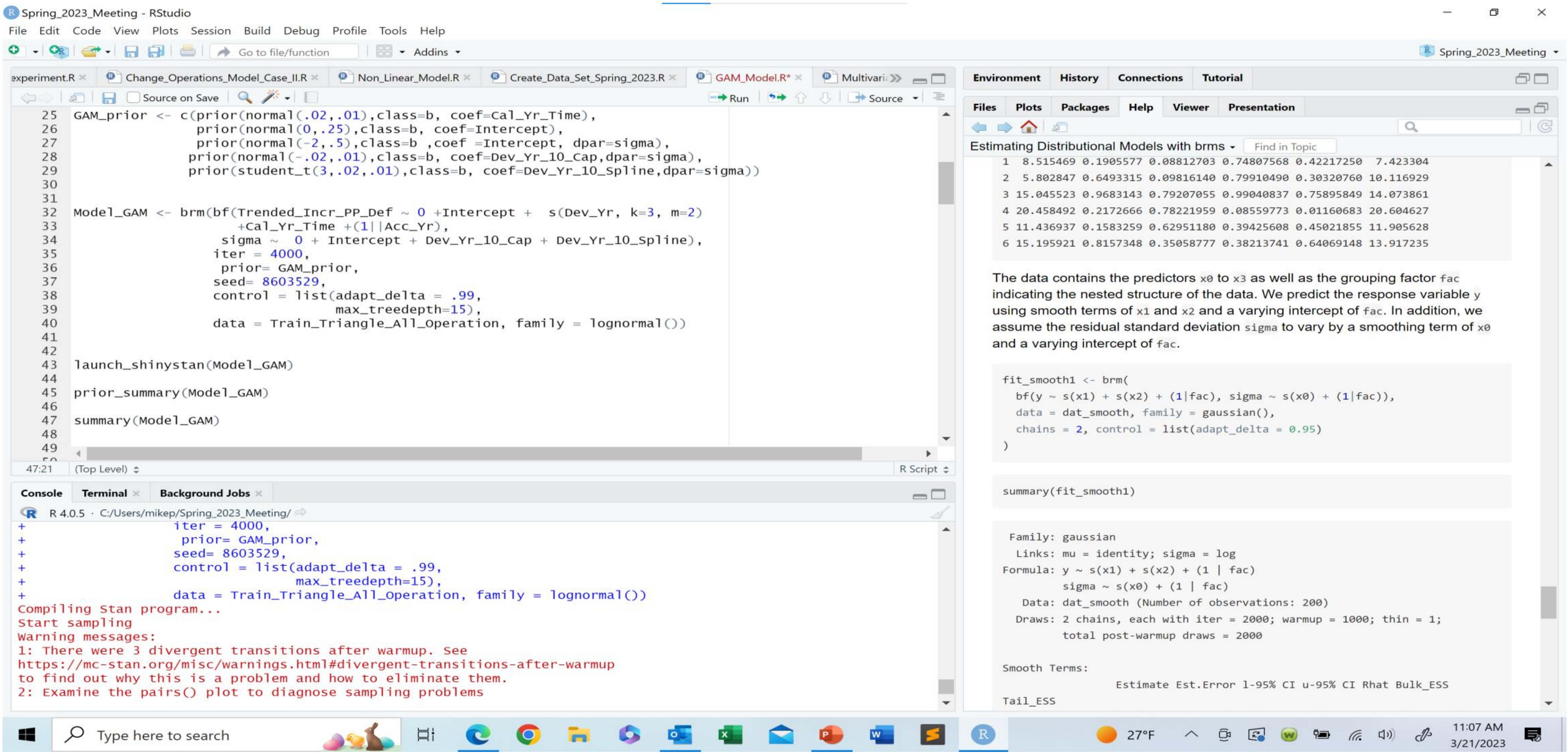
```
Model_GAM <- brm(bf(Trended_Incr_PP_Def ~ 0 +Intercept +  s(Dev_Yr, k=3, m=2)
                    +Cal_Yr_Time +(1||Acc_Yr),
                  sigma ~  0 + Intercept + Dev_Yr_10_Cap + Dev_Yr_10_Spline),
                 iter = 4000,
                  prior= GAM_prior,
                  seed= 8603529,
                 control = list(adapt_delta = .99,
                               max_treedepth=15),
                 data = Train_Triangle_All_Operation, family = lognormal())
```

Generalized Additive Model formula for Dev_Yr invoked in “s(Dev_Yr, k=3, m=2)”

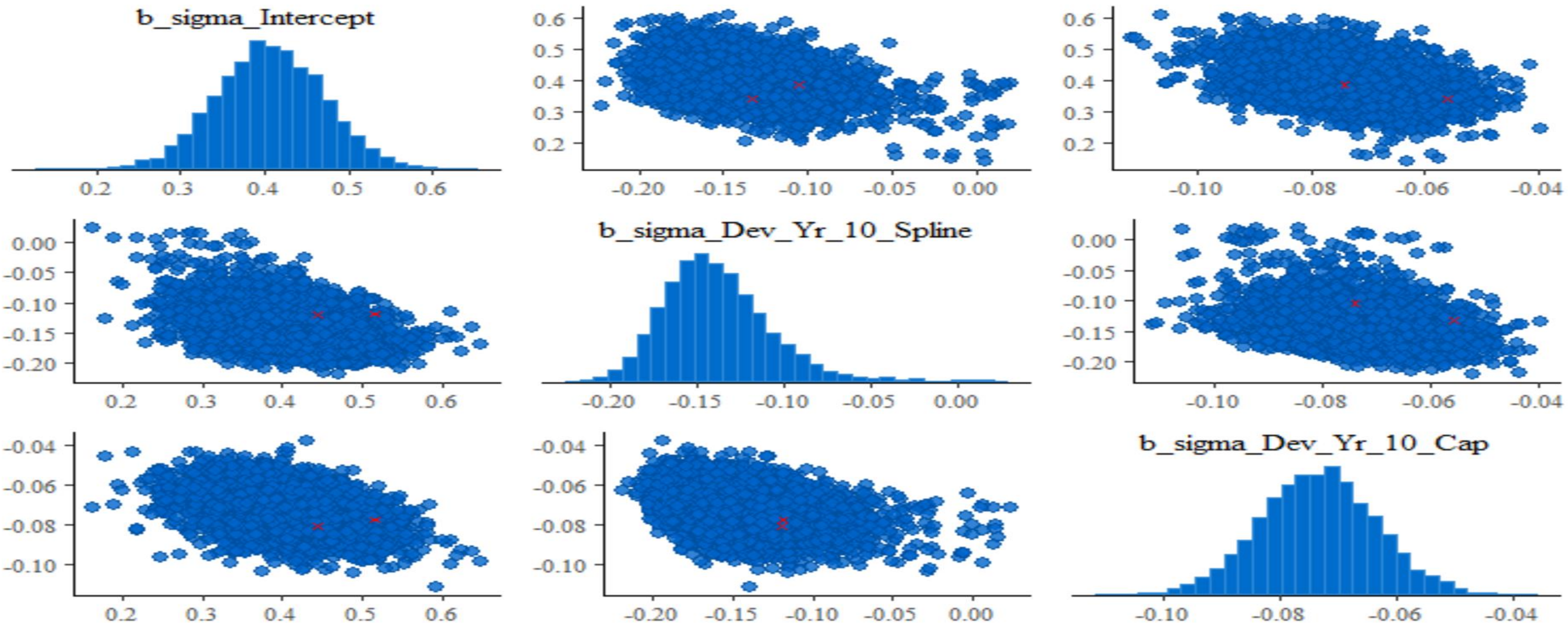
Generalized Additive Model Results Summary

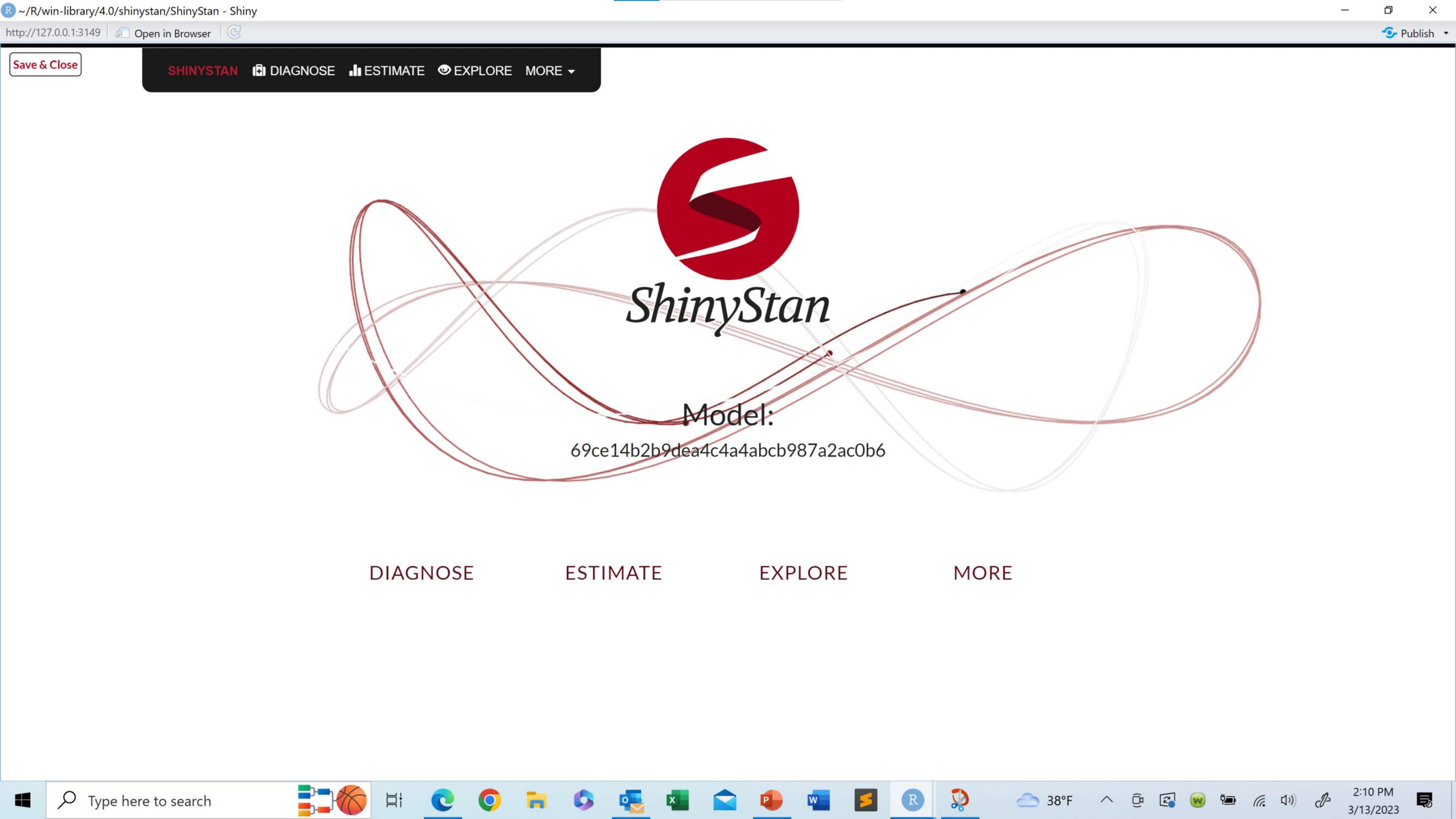
```
 Family: lognormal
  Links: mu = identity; sigma = log
Formula: Trended_Incr_PP_Def ~ 0 + Intercept + s(Dev_Yr, k = 3, m = 2)
        + Cal_Yr_Time + (1 || Acc_Yr)
         sigma ~ 0 + Intercept + Dev_Yr_10_Cap + Dev_Yr_10_Spline
   Data: Train_Triangle_All_Operation (Number of observations: 253)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Smooth Terms:
               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sds(sDev_Yr_1)    11.92      6.23     5.59    27.30 1.00     4251     4354

Group-Level Effects:
~Acc_Yr (Number of levels: 22)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     2.83      0.54     1.93     4.05 1.01      993     1948

Population-Level Effects:
                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept                  0.47      0.27    -0.06     1.01 1.00     4380     4602
Cal_Yr_Time                0.03      0.01     0.01     0.05 1.00     3852     4740
sigma_Intercept            0.41      0.06     0.28     0.53 1.00     4571     3990
sigma_Dev_Yr_10_Cap       -0.07      0.01    -0.09    -0.05 1.00     5081     4702
sigma_Dev_Yr_10_Spline    -0.14      0.03    -0.19    -0.07 1.00     2905     1705
sDev_Yr_1                  1.53      0.08     1.38     1.68 1.00     3360     3652

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
Warning message:
There were 3 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help.
See http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

Checking Syntax for GAM Model

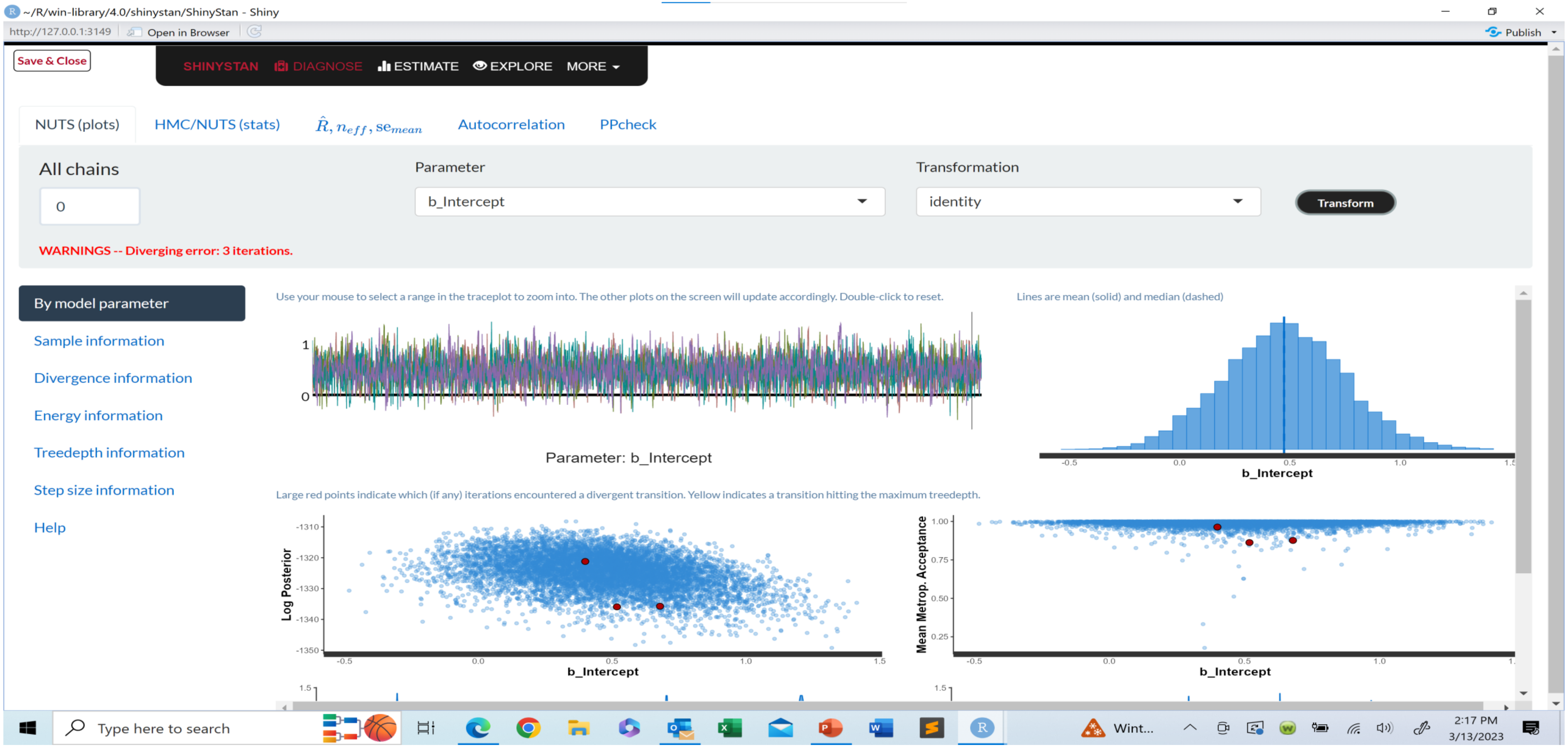


Generalized Additive Model bayesplot example for pairs() graph

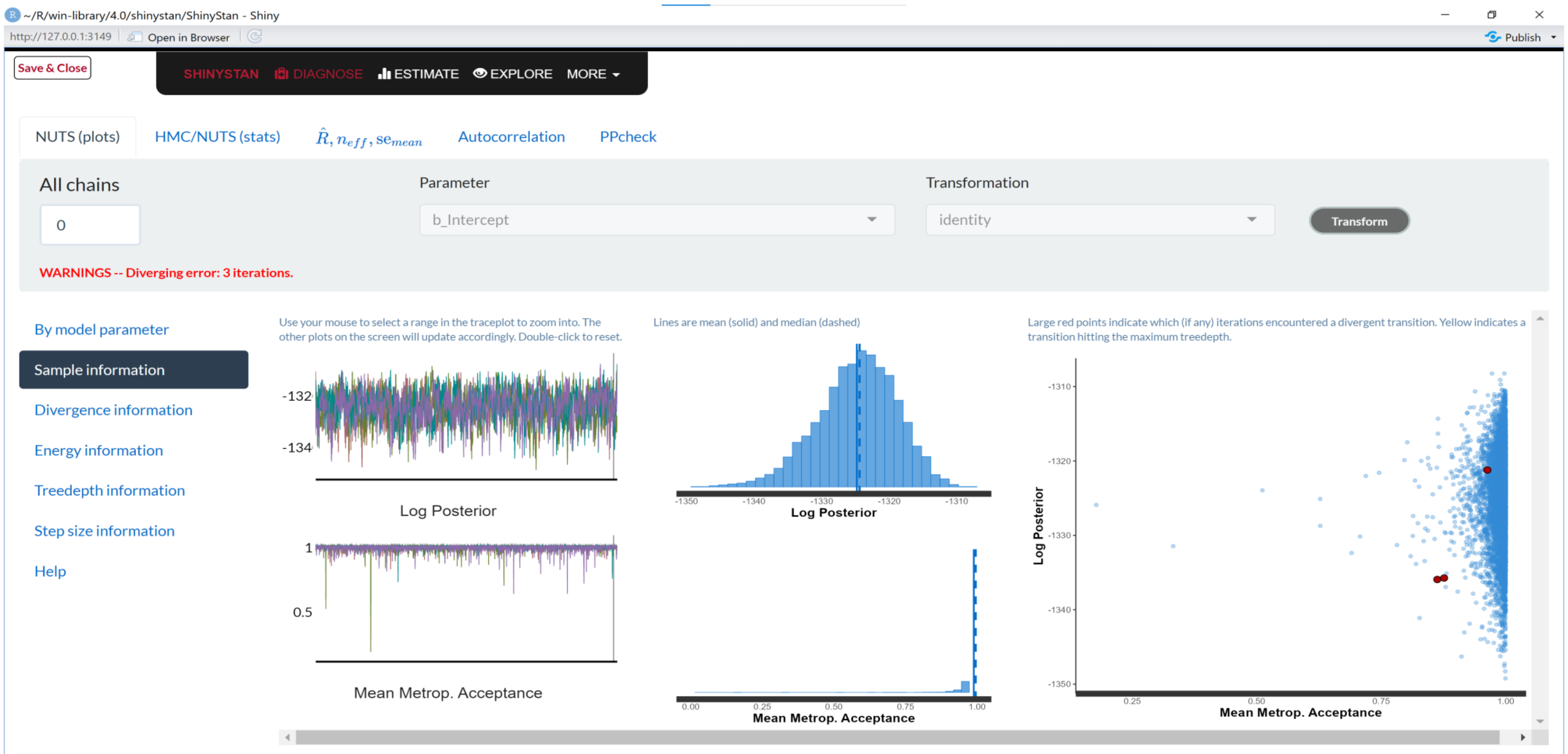



~/R/win-library/4.0/shinystan/ShinyStan - Shiny
http://127.0.0.1:3149
Open in Browser
Publish
Save & Close
SHINYSTAN
DIAGNOSE
ESTIMATE
EXPLORE
MORE
ShinyStan
Model:
69ce14b2b9dea4c4a4abcb987a2ac0b6
DIAGNOSE
ESTIMATE
EXPLORE
MORE
Type here to search
38°F
2:10 PM
3/13/2023

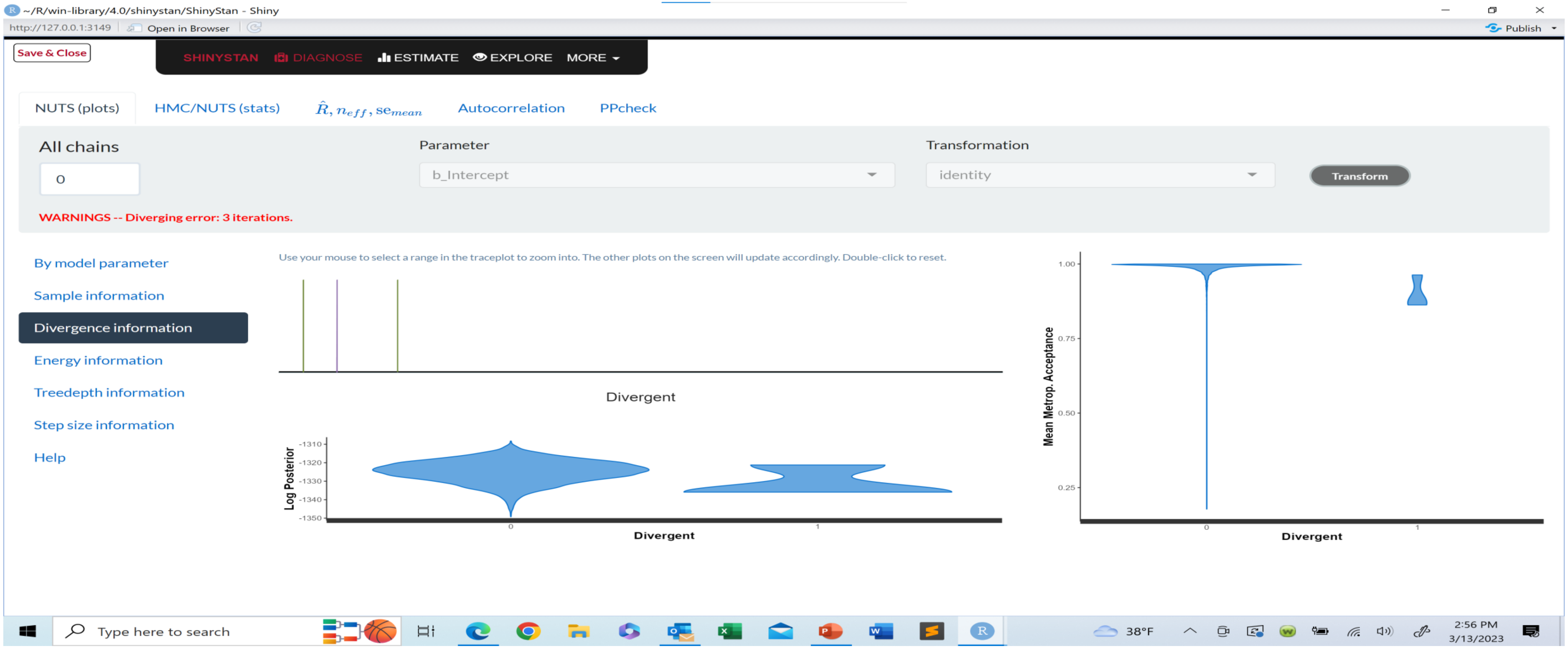
Generalized Additive Model ShinyStan



Generalized Additive Model ShinyStan

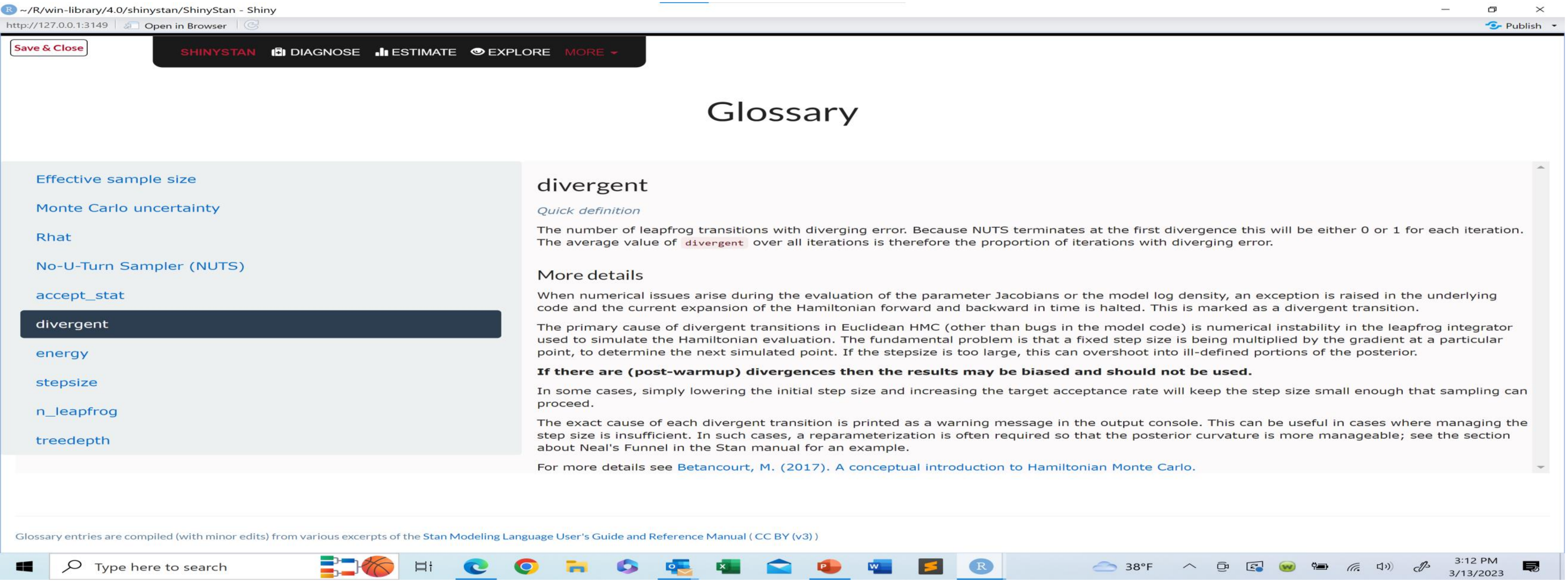


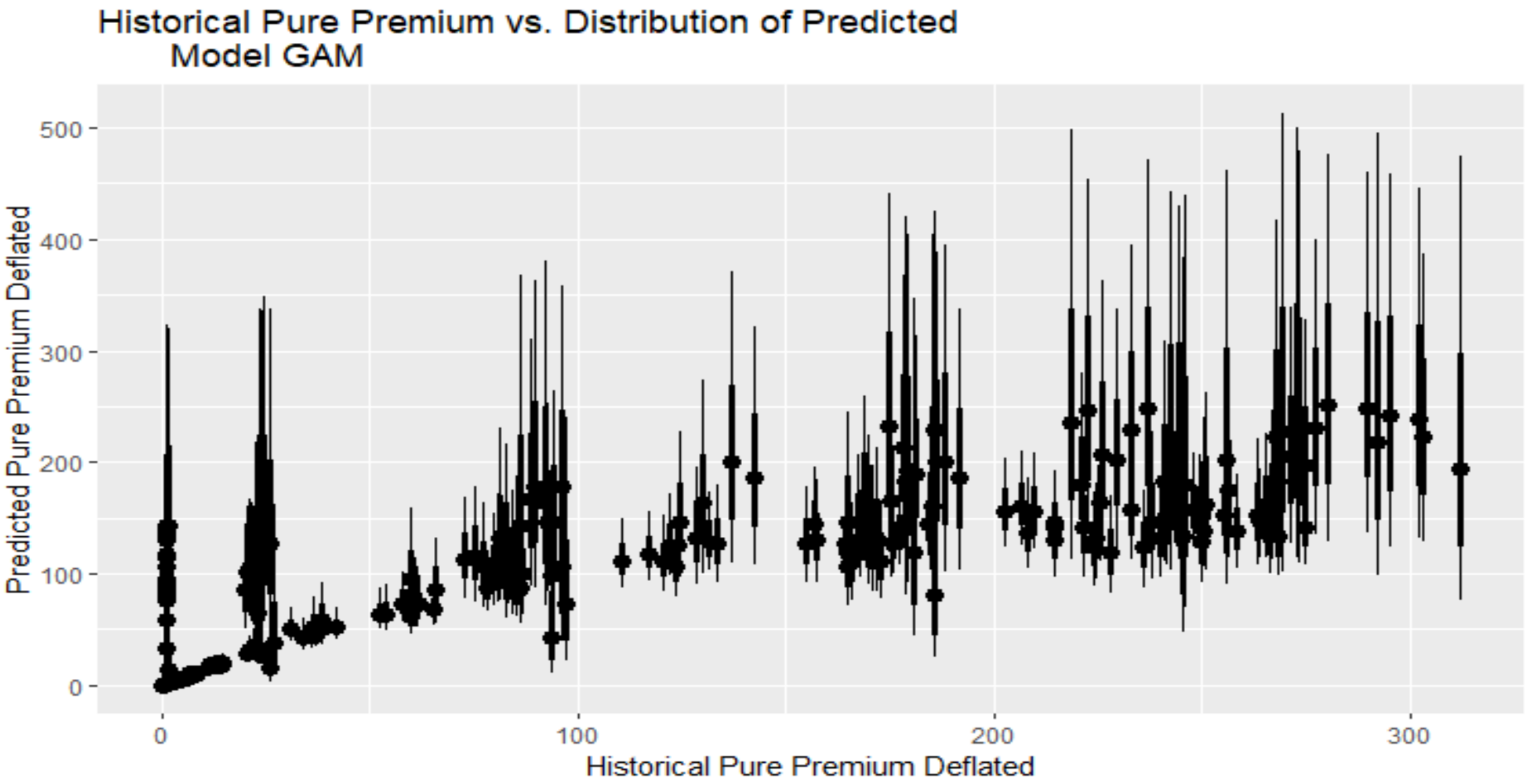
Generalized Additive Model ShinyStan

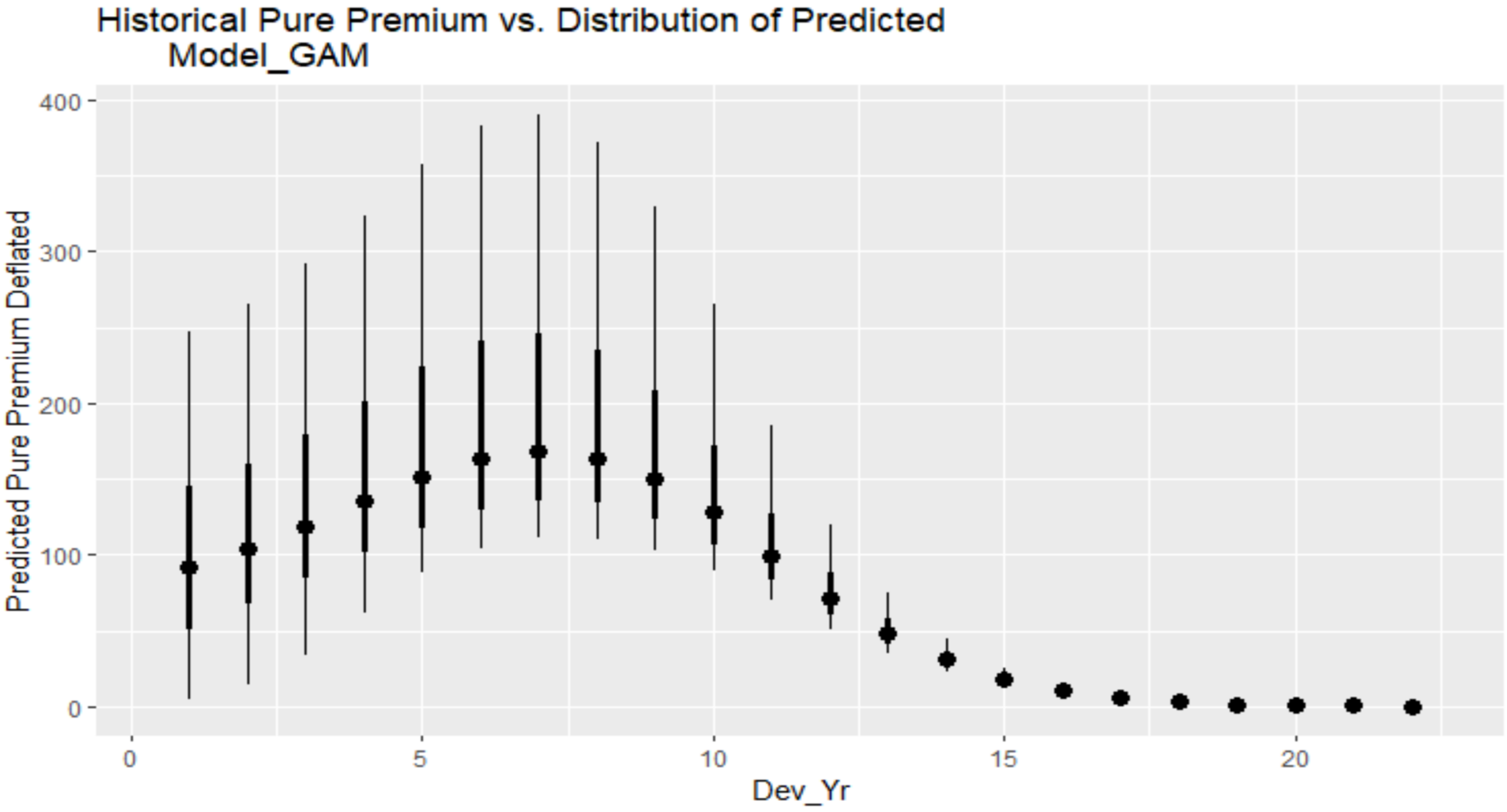


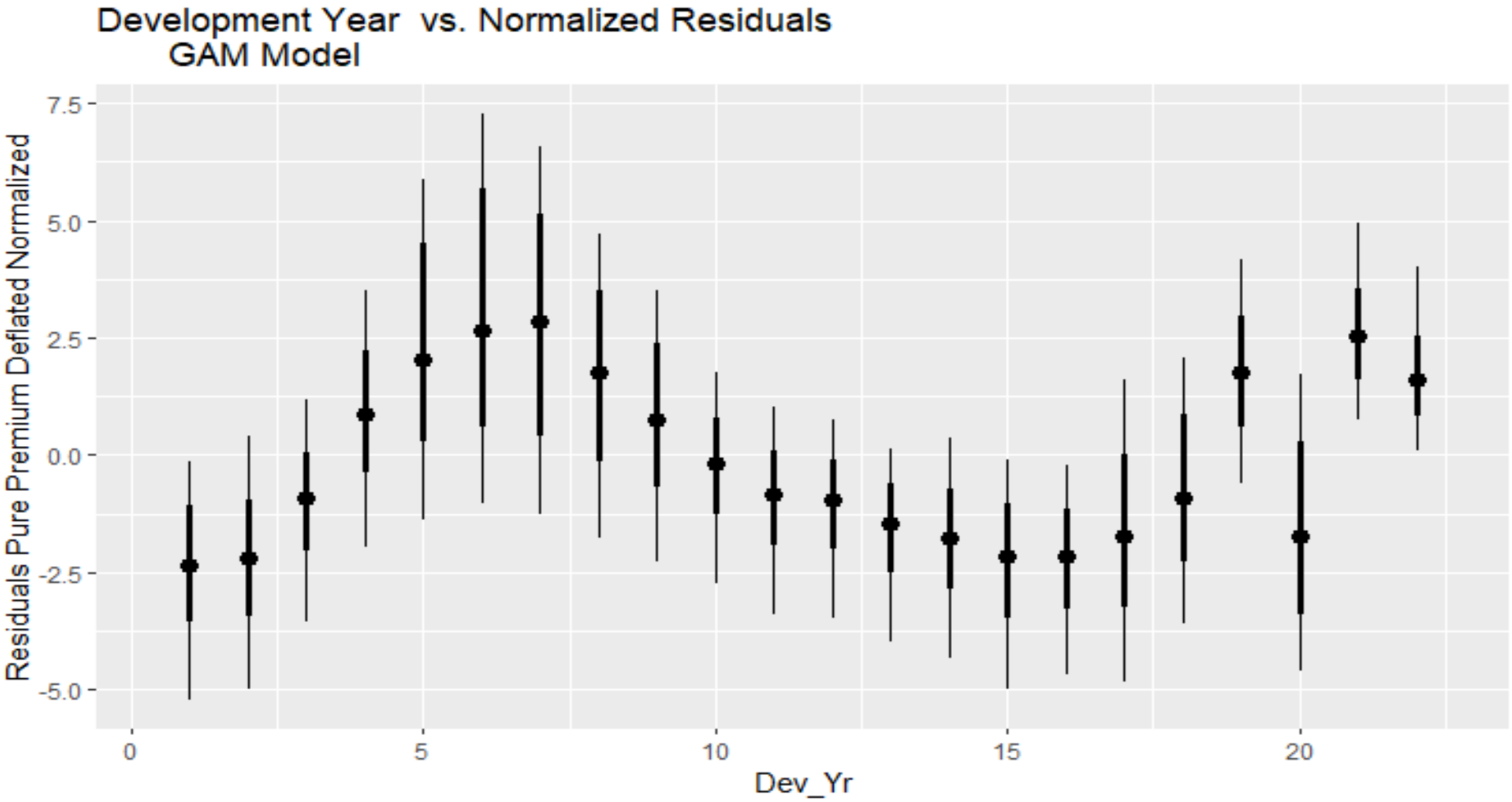
Generalized Additive Model ShinyStan

Generalized Additive Model ShinyStan

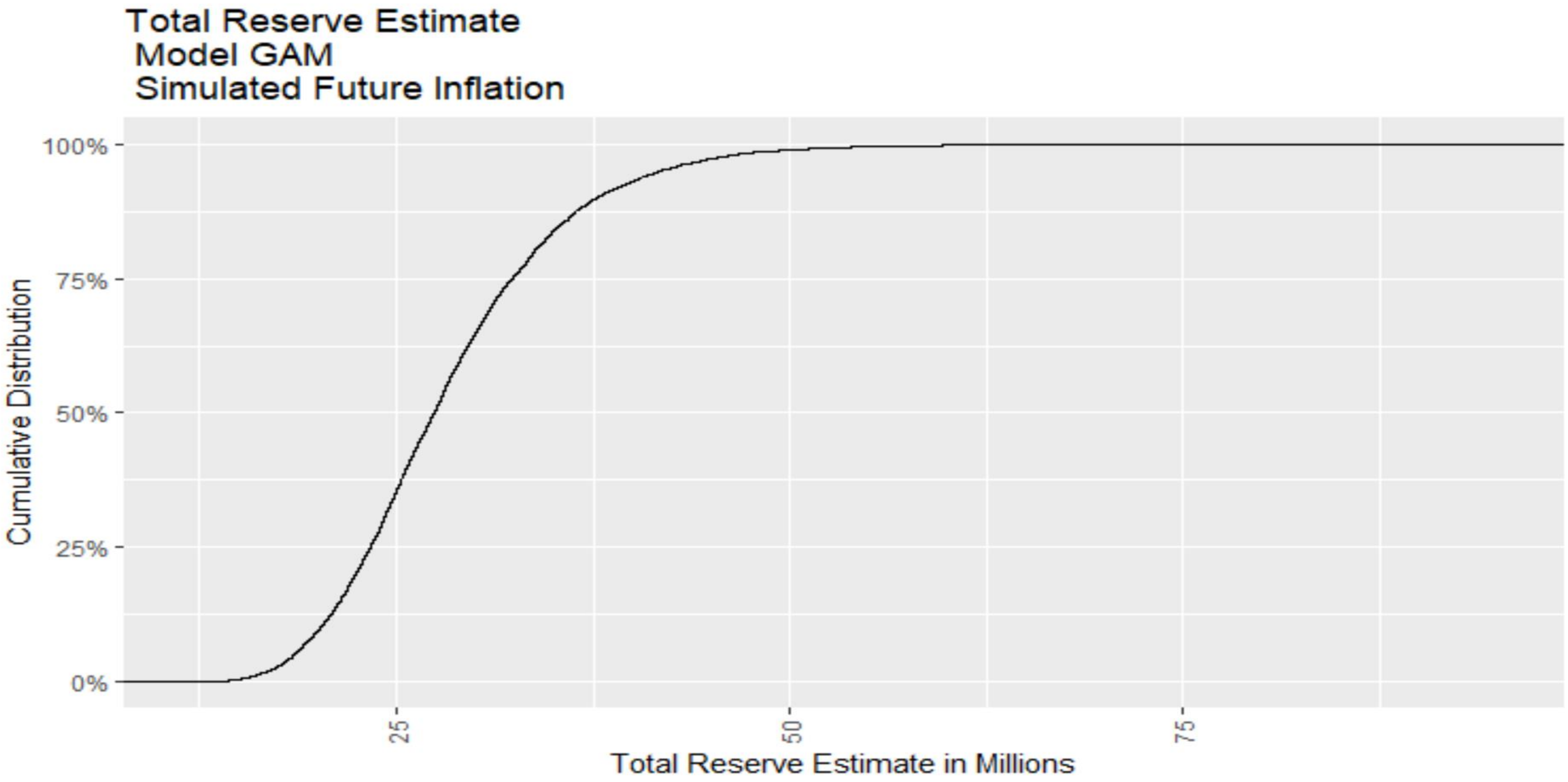



Historical Pure Premium vs. Distribution of Predicted
Model GAM
Predicted Pure Premium Deflated
Historical Pure Premium Deflated
0
100
200
300
400
500
0
100
200
300


Historical Pure Premium vs. Distribution of Predicted
Model_GAM
Predicted Pure Premium Deflated
0
100
200
300
400
0
5
10
15
20
Dev_Yr



Development Year vs. Normalized Residuals
GAM Model
Residuals Pure Premium Deflated Normalized
7.5
5.0
2.5
0.0
-2.5
-5.0
0
5
10
15
20
Dev_Yr


Total Reserve Estimate
Model GAM
Simulated Future Inflation
100%
75%
50%
25%
0%
Cumulative Distribution
25
50
75
Total Reserve Estimate in Millions

Model Selection

Bayesian MCMC Model Comparison Metrics

- PSIS
  - Pareto Smoothed Importance Sampling Cross Validation
  - Approximation of leave one out cross validation
  - Uses individual observation log pointwise predictive distribution details
  - Uses relative effect of a given observation as input to weighting
  - Avoids computational cost of literal leave one out cross validation
  - Diagnostics on safety of using approximation furnished
- WAIC:
  - Widely Applicable Information Criteria
  - Uses log pointwise predictive distribution details
  - Similar to AIC in that there is a penalty term
  - Informed guess on out-of-sample KL divergence
  - Useful for comparison between models
  - Diagnostics on safety of using furnished

Example of Criteria for Comparison Safeguards

```
Poly_1_loo <- add_criterion(Model_Polynomial_1,c("loo", "waic"),
+                moment_match=TRUE)
Warning messages:
1: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

    2:00
11 (4.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
>
> Poly_1_loo_prep <- loo(Model_Polynomial_1)
>
> Poly_1_loo_mm <-loo_moment_match(Model_Polynomial_1, loo=Poly_1_loo_prep)
Warning message:
Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for details.

>
> Poly_1_loo_mm

Computed from 8000 by 253 log-likelihood matrix

         Estimate   SE
elpd_loo   -801.1 25.0
p_loo        18.7  3.3
looic      1602.2 49.9
------
Monte Carlo SE of elpd_loo is 0.1.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     250   98.8%   2273
 (0.5, 0.7]   (ok)         3    1.2%   137
   (0.7, 1]   (bad)        0    0.0%   <NA>
   (1, Inf)   (very bad)   0    0.0%   <NA>

All Pareto k estimates are ok (k < 0.7).
```

Model Comparison Using loo (approximate leave one out)

- Comparison of log pointwise predictive density
- Need both the difference and the se_diff (standard error of difference)
- Comparison is to best model Polynomial 2 in this case (0.0 is best)
- Sometimes standard error relative to difference is so large that differences may not be large enough to say one model is better

```
> model_compare <- loo_compare(Poly_1_loo,Poly_2_loo,
+                              Nonlinear_loo,
+                              criterion = "loo")
> model_compare
              elpd_diff se_diff
Poly_2_loo        0.0       0.0
Nonlinear_loo   -94.3      13.6
Poly_1_loo     -132.2      16.0
```

Model Averaging Results

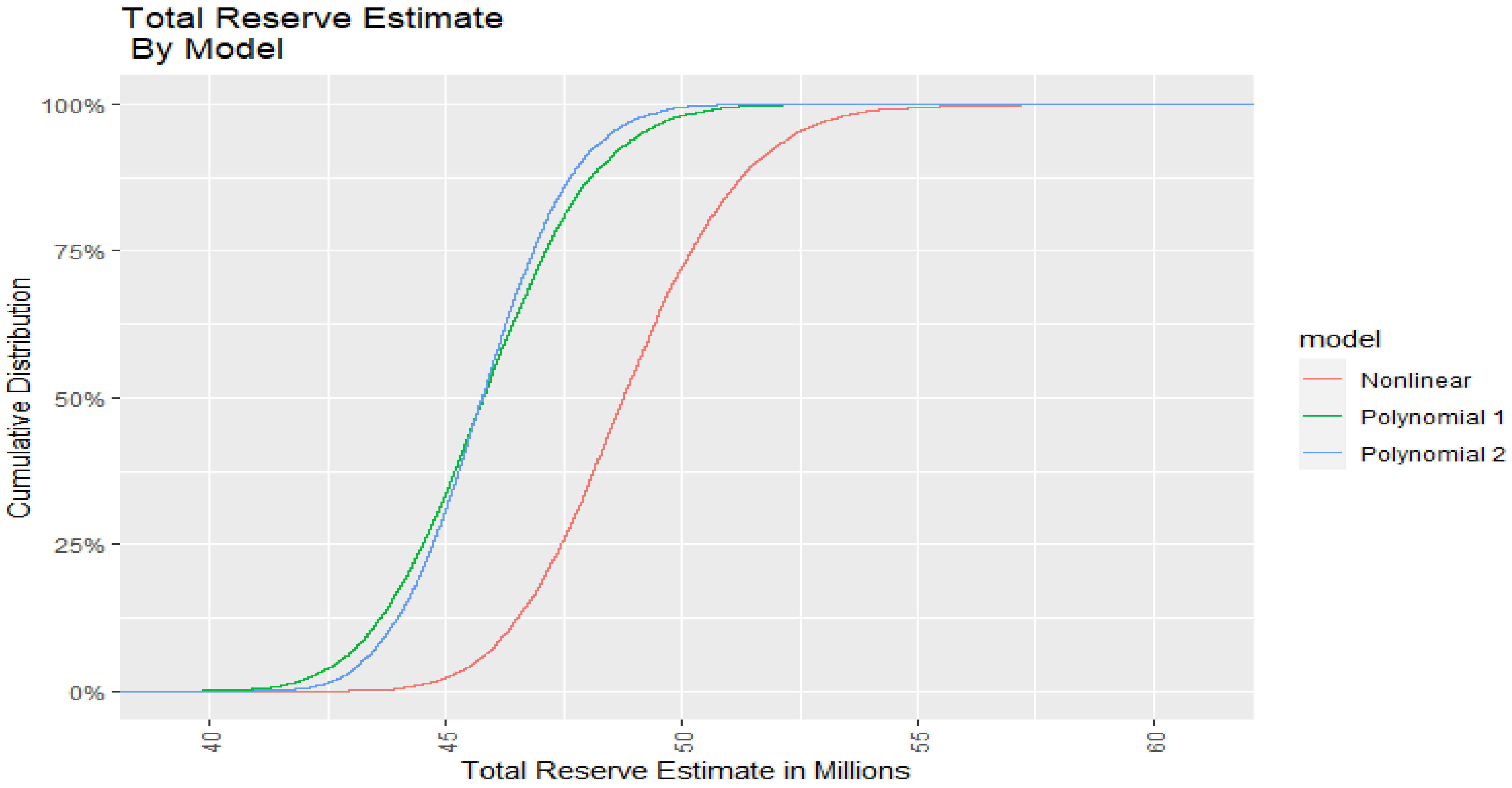
- Weight shows the sampling of forecast losses to obtain the best results in terms of estimated expected log pointwise predictive distribution results.
- The weights are selected to obtain the optimal approximate leave one out (loo) result from the weighted forecast.
- Note that weights for the predicted forecast not the parameter values.

```
loo_model_weights(Model_NL_4,Model_Polynomial_1,Model_Polynomial_2)
Method: stacking
------
                   weight
Model_NL_4         0.053
Model_Polynomial_1 0.000
Model_Polynomial_2 0.947
Warning message:
Some Pareto k diagnostic values are slightly high.
See help('pareto-k-diagnostic') for details.
```

Model Forecast Comparison (millions)

	Polynomial 1	Polynomial 2	Non_Linear	GAM
Min	39.21	40.09	42.38	11.85
1st Quartile	44.52	44.67	47.46	23.33
Median	45.81	45.76	48.77	27.39
Mean	45.85	45.77	48.88	28.4
3rd Quartile	47.12	46.8	50.21	32.36
Max	53.67	51.61	61.05	95.17

Note: Forecasts assume base inflation continues with lognormal distribution mu = .03 and sigma= .02 and loss cost multiplier continues with lognormal distribution mu= .01 and sigma = .01.


Total Reserve Estimate
By Model
Cumulative Distribution
100%
75%
50%
25%
0%
40
45
50
55
60
Total Reserve Estimate in Millions
model
Nonlinear
Polynomial 1
Polynomial 2

Conclusion

- Bayesian probability models are not new
  - Reverend Thomas Bayes in 1754
- Combination of software, algorithm development and computing power makes application practical
  - Metropolis Hastings: First described in 1953 publication
  - Tidyverse programming concepts: ggplot2 first released in 2005
  - STAN (Hamiltonian): Released in 2015
  - Brms released in 2017
- Use requires picking up new vocabulary and concepts
- Value to actuaries:
  - Links credibility weighting to variety of parametric models
  - Document one’s prior knowledge of the problem at hand

Appendix

Additional Resources

- Brms
  - brms: An R Package for Bayesian Multilevel Models using Stan, Paul-Christian Burkner, Journal of Statistical Software 2017
  - Advanced Bayesian Multilevel Modeling with the R Package brms by Paul-Christian Bürkner, The R Journal Vol. 10/1, July 2018
- STAN
  - https://mc-stan.org/
- Rstudio
  - https://rstudio.com/
- Tidyverse
  - Rstudio help
  - R for Data Science: Import, Tidy, Transform, Visualize, and Model Data 1st Edition, Hadley Wickham, Garret Grolemund
- Bayesian MCMC textbooks
  - Statistical Rethinking: A Bayesian Course with Examples in R and STAN (Chapman & Hall/CRC Texts in Statistical Science) 2nd Edition, by Richard McElreath
  - Bayesian Data Analysis (Chapman & Hall/CRC Texts in Statistical Science) 3rd Edition, Gelman et.al.
- Tidybayes
  - https://cran.r-project.org/web/packages/tidybayes

Create Count and Severity Distributions (Page 1)

```
Change_Operation <- Triangle_Dates_Operation %>%
 group_by(Acc_Yr, Dev_Yr) %>%
 mutate(Ln_Dev_Yr =log(Dev_Yr),
      p_rate = case_when(
       Acc_Yr == 2000 ~ exp(-.98 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2001 ~ exp(-.97 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2002 ~ exp(-.98 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2003 ~ exp(-.99 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2004 ~ exp(-1.01 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2005 ~ exp(-1.02 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2006 ~ exp(-1 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2007 ~ exp(-.98 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2008 ~ exp(-.99 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2009 ~ exp(-1.02 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2010 ~ exp(-1.01 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2011 ~ exp(-.98 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2012 ~ exp(-.99 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2013 ~ exp(-1 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2014 ~ exp(-1.01 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2015 ~ exp(-1.02 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2016 ~ exp(-.97 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2017 ~ exp(-.98+ Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2018 ~ exp(-1.02 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2019 ~ exp(-1.03 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2020 ~ exp(-.99 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
       Acc_Yr == 2021 ~ exp(-.98 + Dev_Yr * (-.30) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
      ),
 #     p_rate = exp(-1 + Dev_Yr * (-.35) + Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (1.0)),
      p_rate_adj =Rptd_Cnt * p_rate,
```

Create Count and Severity Distributions (Page 2)

```
mu_ln = case_when(
  Acc_Yr == 2000 ~ (1.97+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2001 ~ (1.98+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2002 ~ (1.99+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2003 ~ (2.02+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2004 ~ (2.03+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2005 ~ (2.01+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2006 ~ (1.97+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2007 ~ (1.98+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2008 ~ (1.99+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2009 ~ (2+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2010 ~ (2.02+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2011 ~ (2.03+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2012 ~ (2.04+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2013 ~ (1.99+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2014 ~ (2.01+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2015 ~ (2.02+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2016 ~ (1.98+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2017 ~ (1.97+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2018 ~ (2.02+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2019 ~ (2.04+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2020 ~ (1.98+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),
  Acc_Yr == 2021 ~ (2.02+ Dev_Yr * (-.6 )+ Dev_Yr_Sqrd * (0) + Ln_Dev_Yr * (4.5)),),
    sigma_ln = exp(-2.0 +Dev_Yr *.015+ Dev_Yr_Sqrd * 0 + (.1)*Ln_Dev_Yr),
    Paid_Cnt = rpois(1,lambda =p_rate_adj) ,
    Expected_Payment= exp(mu_ln + .5 * (sigma_ln**2)),
    Mean_Payment = if_else (Paid_Cnt ==0, 0,
                   mean(rlnorm(Paid_Cnt,meanlog = mu_ln,sdlog=sigma_ln))),
    Incr_Payment = Paid_Cnt * Mean_Payment,
    Pure_Prem =Incr_Payment/Rptd_Cnt)
```

Create Transformed Variables (Page 1)

```
Complete_Triangle_Operation <- left_join(Change_Operation,Cal_Yr_Trend_Acc, by=c('Cal_Yr')) %>%
  mutate(Trended_Incr_Payment = Incr_Payment * Accum_Loss_Cost_Trend_Mid,
         Trended_Incr_Payment_Def =Trended_Incr_Payment/Accum_Infl_Index_Mid,
         Trended_Incr_PP =Trended_Incr_Payment/Rptd_Cnt,
         Trended_Mean_Payment = Mean_Payment * Accum_Loss_Cost_Trend_Mid,
         Trended_Mean_Payment_Def = Trended_Mean_Payment/Accum_Infl_Index_Mid,
         Trended_Incr_PP_Def =Trended_Incr_Payment_Def/Rptd_Cnt,
         Paid_Cnt_Freq =Paid_Cnt/Rptd_Cnt,
         Ln_Paid_Cnt_Freq = log(Paid_Cnt_Freq),
         Inv_Rptd_Cnt =1/Rptd_Cnt,
         Ln_Trd_Incr_PP =log(Trended_Incr_PP),
         Ln_Trd_Incr_PP_Def =log(Trended_Incr_PP_Def),
         Ln_Mean_Payment_No_Trend =log(Mean_Payment),
         Ln_Mean_Payment_Def =log(Trended_Mean_Payment_Def),
         Cal_Yr_Time = Cal_Yr - 2000,
         Dev_Yr_4_Cap = if_else(Dev_Yr < 4, Dev_Yr,
                          as.integer(4) ),
         Dev_Yr_4_Spline=if_else(Dev_Yr < 4,0,
                           Dev_Yr-4),
         Dev_Yr_4_Cap_Sqrd = Dev_Yr_4_Cap * Dev_Yr_4_Cap ,
         Dev_Yr_4_Spline_Sqrd =Dev_Yr_4_Spline *Dev_Yr_4_Spline,
         Dev_Yr_5_Cap = if_else(Dev_Yr < 5, Dev_Yr,
                          as.integer(5) ),
         Dev_Yr_5_Spline=if_else(Dev_Yr < 5,0,
                           Dev_Yr-5),
         Dev_Yr_5_Spline_Sqrd = Dev_Yr_5_Spline* Dev_Yr_5_Spline,
         Dev_Yr_6_Cap = if_else(Dev_Yr < 6, Dev_Yr,
                          as.integer(6) ),
```

Create Transformed Variables (Page 2)

```
Dev_Yr_6_Cap_Sqrd = Dev_Yr_6_Cap * Dev_Yr_6_Cap ,
Dev_Yr_6_Spline=if_else(Dev_Yr < 6,0,
                Dev_Yr-6),
Dev_Yr_6_Spline_Sqrd = Dev_Yr_6_Spline* Dev_Yr_6_Spline,
Dev_Yr_6_Cap_Sqrd = Dev_Yr_6_Cap * Dev_Yr_6_Cap ,
Dev_Yr_GT_1 = if_else(Dev_Yr< 2,0,1),
Dev_Yr_1_Factor = as.factor(case_when(
  Dev_Yr == 1 ~ 1,
  Dev_Yr >1 ~2
)),
Dev_Yr_8_Cap = if_else(Dev_Yr < 8, Dev_Yr,
               as.integer(8) ),
Dev_Yr_8_Spline=if_else(Dev_Yr < 8,0,
                Dev_Yr-8),
Dev_Yr_8_Cap_Sqrd = Dev_Yr_8_Cap * Dev_Yr_8_Cap ,
Dev_Yr_8_Spline_Sqrd =Dev_Yr_8_Spline * Dev_Yr_8_Spline,
Dev_Yr_1_Factor = as.factor(case_when(
  Dev_Yr == 1 ~ 1,
  Dev_Yr >1 ~2
)),
Dev_Yr_3_Factor = as.factor(case_when(
  Dev_Yr == 1 ~ 1,
  Dev_Yr == 2 ~2,
  Dev_Yr == 3 ~3,
  Dev_Yr > 3 ~4
)),
Dev_Yr_2_Factor = as.factor(case_when(
  Dev_Yr == 1 ~ 1,
  Dev_Yr == 2 ~2,
  Dev_Yr > 2 ~3
)),
```

Create Transformed Variables (Page 3)

```
Dev_Yr_3_14_Factor = as.factor(case_when(
  Dev_Yr == 1 ~ 1,
  Dev_Yr == 2 ~2,
  Dev_Yr == 3 ~3,
  Dev_Yr == 4 ~4,
  Dev_Yr == 5 ~4,
  Dev_Yr == 6 ~4,
  Dev_Yr == 7 ~4,
  Dev_Yr == 8 ~4,
  Dev_Yr == 9 ~4,
  Dev_Yr == 10 ~4,
  Dev_Yr == 11 ~4,
  Dev_Yr == 12 ~4,
  Dev_Yr == 13 ~4,
  Dev_Yr >13  ~5,
)),
Dev_Yr_2_14_Factor = as.factor(case_when(
  Dev_Yr == 1 ~ 1,
  Dev_Yr == 2 ~2,
  Dev_Yr == 3 ~3,
  Dev_Yr == 4 ~3,
  Dev_Yr == 5 ~3,
  Dev_Yr == 6 ~3,
  Dev_Yr == 7 ~3,
  Dev_Yr == 8 ~3,
  Dev_Yr == 9 ~3,
  Dev_Yr == 10 ~3,
  Dev_Yr == 11 ~3,
  Dev_Yr == 12 ~3,
  Dev_Yr == 13 ~3,
  Dev_Yr >13  ~4,
)),
Dev_Yr_10_Spline =if_else(Dev_Yr < 10,0,
                  Dev_Yr - 10),
Dev_Yr_10_Cap =if_else(Dev_Yr < 10,Dev_Yr,
                  as.integer(10)),
Dev_Yr_12_Cap =if_else(Dev_Yr < 10,Dev_Yr,
                as.integer(12)),
Dev_Yr_15_Cap =if_else(Dev_Yr < 15,Dev_Yr,
                as.integer(15)),
Dev_Yr_12_Spline =if_else(Dev_Yr < 12,0,
                  Dev_Yr - 12),
Dev_Yr_10_Spline_Cap_15 = case_when(
  Dev_Yr < 10 ~0,
  Dev_Yr < 16 ~Dev_Yr-10,
  Dev_Yr >=16 ~5),
Dev_Yr_10_Spline_Ln =if_else(Dev_Yr < 11,0,
                    log(Dev_Yr_10_Spline)),
Dev_Yr_15_Spline =if_else(Dev_Yr < 15,0,
                  Dev_Yr - 15),
Dev_Yr_15_Spline_Ln =if_else(Dev_Yr < 16,0,
                    log(Dev_Yr_15_Spline)))
```