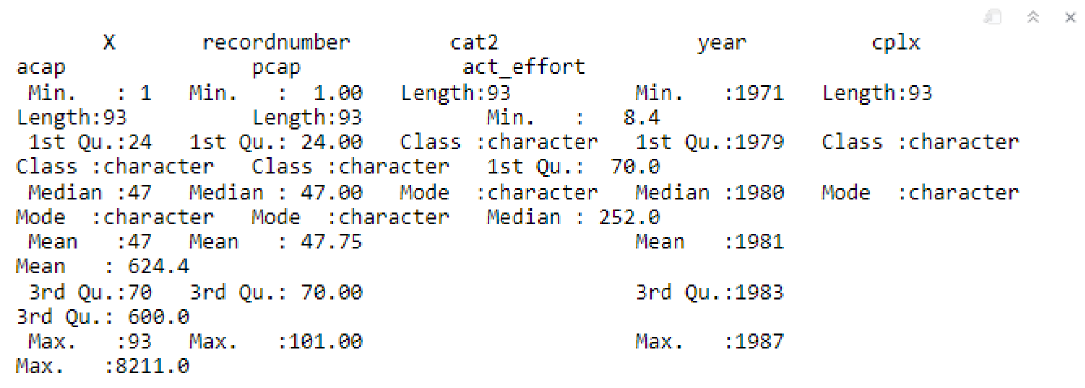
We are working as a group
Berkan Ozturk (V00892651)
Reece Pretorius (V00880300)

Assignment 2 - Bayesian Modeling

1. Description of the data

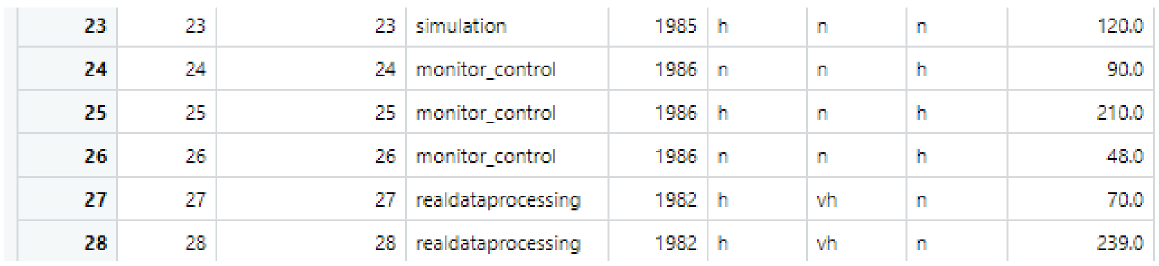
The data we are using is about the software projects at NASA, and it consists of 93 projects. It consists of multiple variables such as project category, year, complexity, analyst capability, programmer capability, and the actual effort. In **figure 1**, all the descriptive statistics regarding the dataset can be found such as mean, median and mode.



```
       X        recordnumber        cat2                year          cplx
 acap               pcap               act_effort
  Min.   : 1    Min.   :  1.00   Length:93          Min.   :1971   Length:93
 Length:93          Length:93            Min.   :   8.4
  1st Qu.:24    1st Qu.: 24.00   Class :character   1st Qu.:1979   Class :character
 Class :character   Class :character   1st Qu.:  70.0
  Median :47    Median : 47.00   Mode  :character   Median :1980   Mode  :character
 Mode  :character   Mode  :character   Median : 252.0
  Mean   :47    Mean   : 47.75                      Mean   :1981
 Mean   : 624.4
  3rd Qu.:70    3rd Qu.: 70.00                      3rd Qu.:1983
 3rd Qu.: 600.0
  Max.   :93    Max.   :101.00                      Max.   :1987
 Max.   :8211.0
```

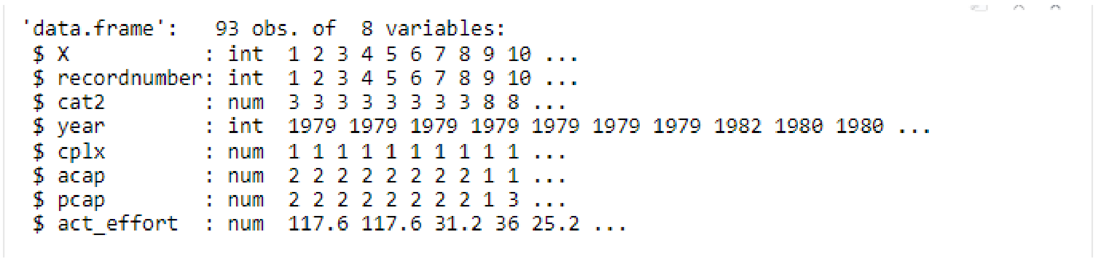
Figure 1. Descriptive information regarding dataset

In **figure 2** we have some entries from the original dataset provided for the assignment. The original data-frame has columns that are of type '**char**' and in order to run the regression model we need them to be numerical, to convert the necessary columns; '**cat2**', '**cplx**', '**pcap**', '**acap**' we first converted them to factors using **as.factor()** and then **as.numeric(unclass())** which gives us **figure 3** below where we can see that they have all been converted. The final step of cleaning the data included dropping the '**recordnumber**' and '**year**' columns as they were not useful for our model.



23	23	23	simulation	1985	h	n	n	120.0
24	24	24	monitor_control	1986	n	n	h	90.0
25	25	25	monitor_control	1986	h	n	h	210.0
26	26	26	monitor_control	1986	n	n	h	48.0
27	27	27	realdataprocessing	1982	h	vh	n	70.0
28	28	28	realdataprocessing	1982	h	vh	n	239.0

Figure 2. Snippet of uncleaned dataset



```
'data.frame':   93 obs. of  8 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ recordnumber: int  1 2 3 4 5 6 7 8 9 10 ...
 $ cat2        : num  3 3 3 3 3 3 3 3 8 8 ...
 $ year        : int  1979 1979 1979 1979 1979 1979 1979 1982 1980 1980 ...
 $ cplx        : num  1 1 1 1 1 1 1 1 1 1 ...
 $ acap        : num  2 2 2 2 2 2 2 2 1 1 ...
 $ pcap        : num  2 2 2 2 2 2 2 2 1 3 ...
 $ act_effort  : num  117.6 117.6 31.2 36 25.2 ...
```

Figure 3. Summary of columns converted to numeric

2. ***Defense of the Likelihood***

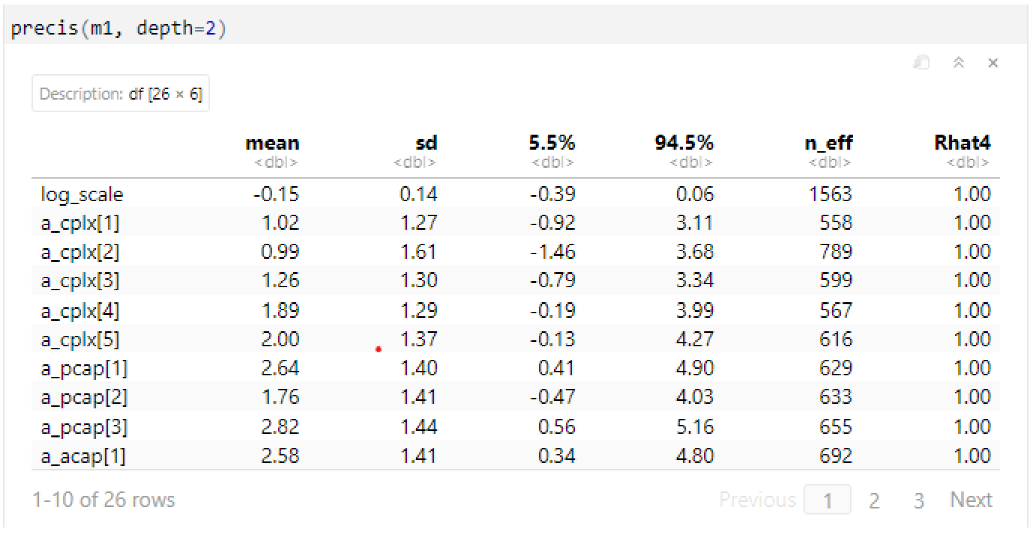
At first, we decided to choose a Poisson distribution because we wanted to predict the value of actual effort and it is a number that can go from 0 to infinity; however, one of the assumptions of the Poisson distribution is that the variance should be equal to the mean or at least approximately equal to the mean, and in our case, this condition was not met. Thus, we decided to try the **Gamma-Poisson distribution**. We have developed two models using Gamma-Poisson distribution.

The Math notation for our model : $L_i \sim \text{GammaPoisson}(n_i, p_i)$

3. ***Discussion regarding the Priors***

For our initial priors we based them off of our prior knowledge and analysis of the dataset '**act_effort**', noting from the dataset the actual effort **maximum is around 8211** hours and the **minimum effort hours is around 8.4** which seems like an outlier for how low it is. Depending on the type of project in the software field the actual effort hours can get very high based on many factors so our prior for the effort is generated using '**max(rlnorm(1e2, 0, 3))**' which gives us a range that should cover the range of act_effort and gives a bit more room for larger/longer projects not included in our dataset.

As for what happens when we change these priors, when adjusting the priors by increasing (we should see that the coefficients will be closer to the prior mean and exhibit less variability.) to norm(0, 100), we observe the standard deviation going up from around 1.02 for cmplx[1] to 49.1, we also see that the effective sample size decreases drastically, we were able to use this information to narrow our priors for the final model to increase our precision. We ended up going with **dnorm(0, 1)** for our priors for each of the predictor variables and **normal(0, 3)** for the target variable '**act_effort**' which increased the effective sample size for our model. No other noticeable effects could be seen in the other statistics when lowering the priors for the predictor variables. Below in **figure 4** is a breakdown of the deviation, effective sample size and more.



```
precis(m1, depth=2)
```

Description: df [26 × 6]

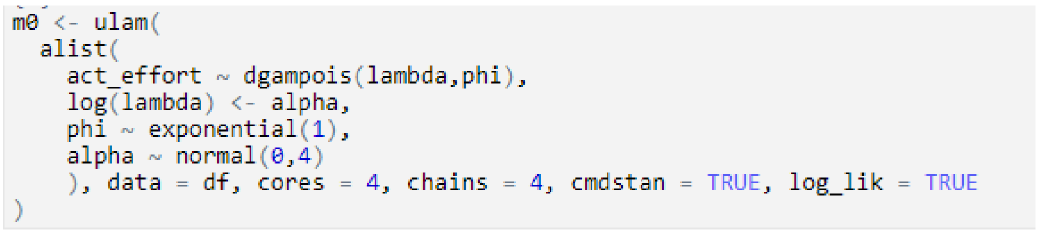
	mean <dbl>	sd <dbl>	5.5% <dbl>	94.5% <dbl>	n_eff <dbl>	Rhat4 <dbl>
log_scale	-0.15	0.14	-0.39	0.06	1563	1.00
a_cplx[1]	1.02	1.27	-0.92	3.11	558	1.00
a_cplx[2]	0.99	1.61	-1.46	3.68	789	1.00
a_cplx[3]	1.26	1.30	-0.79	3.34	599	1.00
a_cplx[4]	1.89	1.29	-0.19	3.99	567	1.00
a_cplx[5]	2.00	1.37	-0.13	4.27	616	1.00
a_pcap[1]	2.64	1.40	0.41	4.90	629	1.00
a_pcap[2]	1.76	1.41	-0.47	4.03	633	1.00
a_pcap[3]	2.82	1.44	0.56	5.16	655	1.00
a_acap[1]	2.58	1.41	0.34	4.80	692	1.00

1-10 of 26 rows   Previous 1 2 3 Next

Figure 4. Precision check for initial priors before adjusting

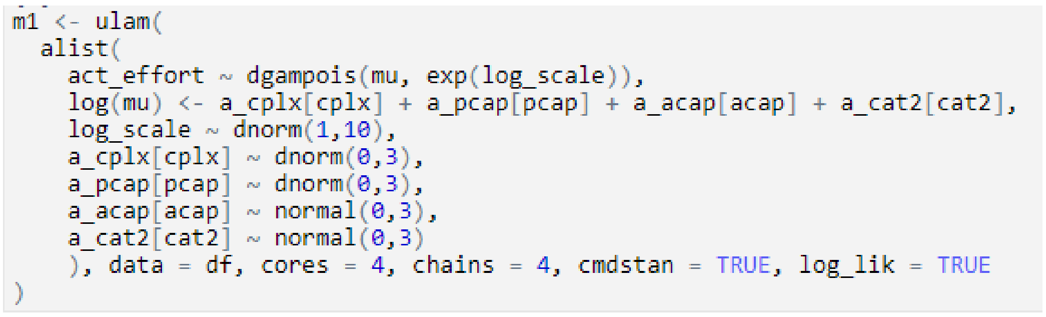
4. ***Results and Interpretation***

In the figure(s) below, we run our first and second models using the ulam() function, and then we check the diagnostics of our final model '**m1**' using the precis() function. The results of the diagnostics can be found in **figure 4** above.



```
m0 <- ulam(
  alist(
    act_effort ~ dgampois(lambda,phi),
    log(lambda) <- alpha,
    phi ~ exponential(1),
    alpha ~ normal(0,4)
    ), data = df, cores = 4, chains = 4, cmdstan = TRUE, log_lik = TRUE
)
```

Figure 5. First model using ulam()

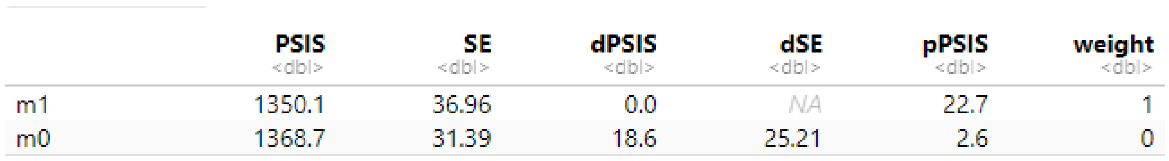


```
m1 <- ulam(
  alist(
    act_effort ~ dgampois(mu, exp(log_scale)),
    log(mu) <- a_cplx[cplx] + a_pcap[pcap] + a_acap[acap] + a_cat2[cat2],
    log_scale ~ dnorm(1,10),
    a_cplx[cplx] ~ dnorm(0,3),
    a_pcap[pcap] ~ dnorm(0,3),
    a_acap[acap] ~ normal(0,3),
    a_cat2[cat2] ~ normal(0,3)
    ), data = df, cores = 4, chains = 4, cmdstan = TRUE, log_lik = TRUE
)
```

Figure 6. Second model using ulam()

Next, we compare our first model '**m0**' with our second model '**m1**' using the LOO function in **figure 7**, we can see a lower PSIS score but only slightly, the difference in PSIS is 0.0 so there is no difference in the PSIS score from our second model

compared to our first, in the end our second model using the initial and adjusted priors did not show much if any improvement over our first model, we suspect that this is a result of the first model not using all of the same predictor variables and only the target variable in common with each other. We also used gamma poisson for both of our models, and in hindsight we should have tested out different approaches instead of sticking to the same one and trying to improve it through adjusting the priors.



	PSIS <dbl>	SE <dbl>	dPSIS <dbl>	dSE <dbl>	pPSIS <dbl>	weight <dbl>
m1	1350.1	36.96	0.0	NA	22.7	1
m0	1368.7	31.39	18.6	25.21	2.6	0

Figure 7. Results of comparing two models

In **figure 8,** we have our posterior predictive check to see how well our model performed. We can see that our predictions are mostly within the 90% credible interval and our estimated values are close to the empirical data and we rarely underestimate points. You can also see our estimated mean uncertainty is quite large.

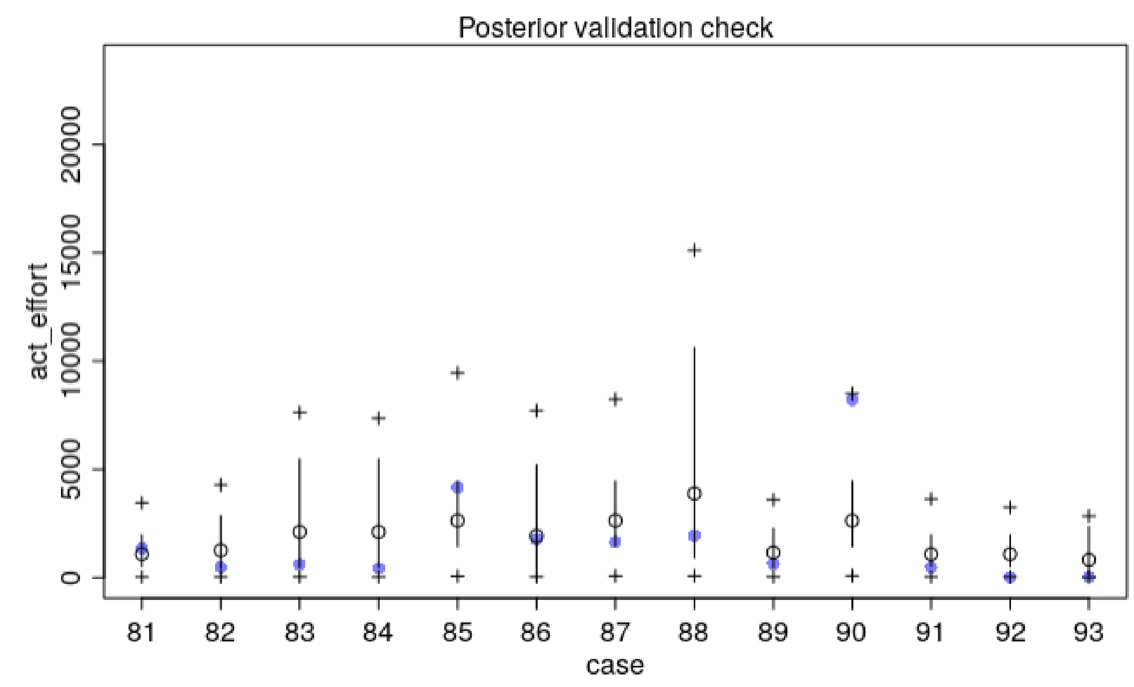


Figure 8. Posterior Validation check

5. DAG

The DAG in **figure 8** shows the relationship between the variables of the dataset, and which variables are affected by which variables. This is based on our prior knowledge and how we interpreted the dataset features. We think that the category does not directly affect effort but instead the category of project affects how complex the project is which in turn affects the actual effort required. Programmer capability and analyst capability both directly affect effort where analyst capability affects programmer capability in some ways where the programmer may be helped by what the analyst can bring to the project.

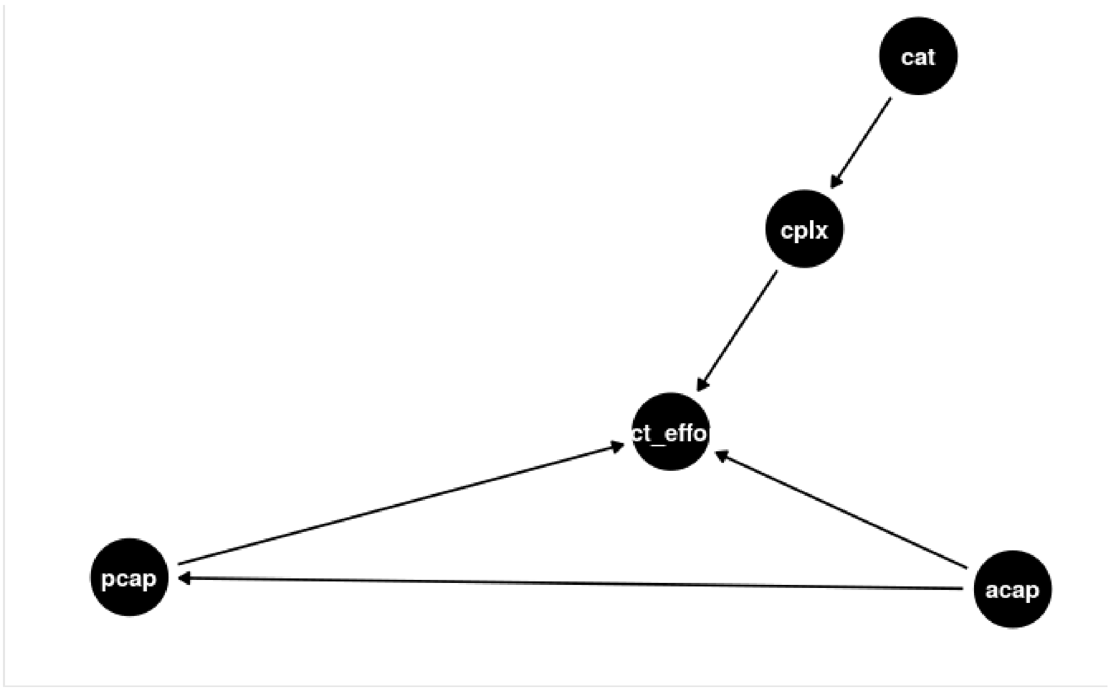


Figure 9. DAG

6. ***Presentation of diagnostics***

The figure below represents the transplot of the final model.

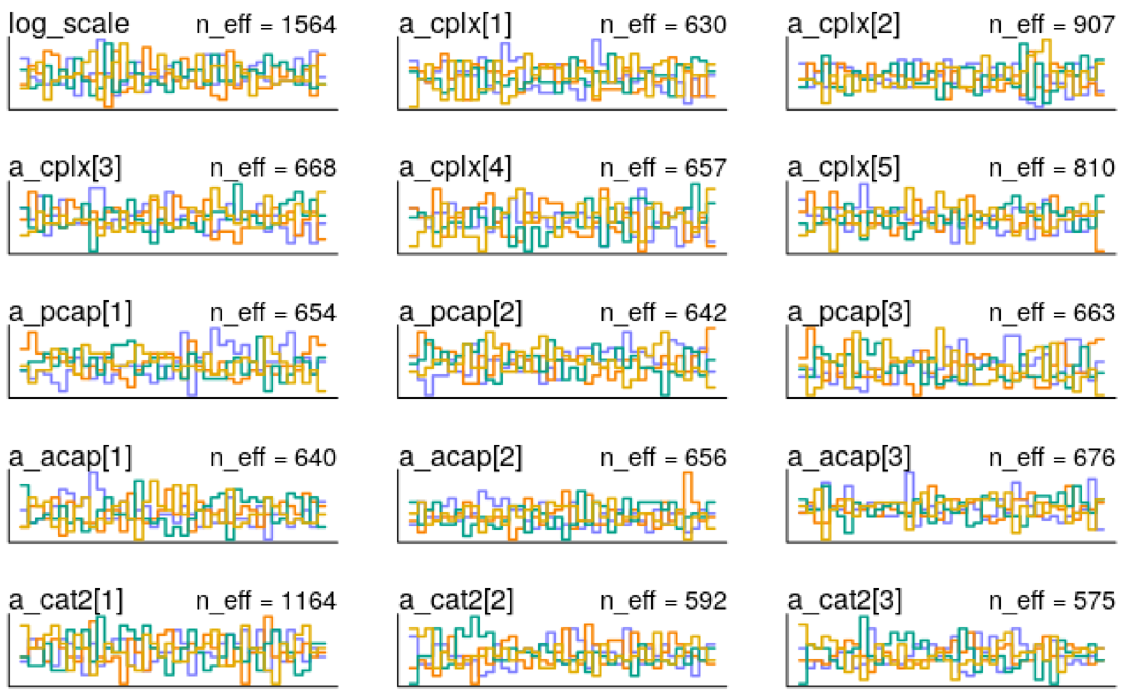


Figure 10. Transplot of the final model

If we examine the transplots above, we can assume that they are all good as they all look like a big mess of colors and none of them have any chains diverging.