Unit 5 Live Session

Discrete Response Model Part 5

Figure 1: South Hall

Class Announcements

No HW this week

Lab-1 due in 1 week

Roadmap

Rearview Mirror

- Model unordered and ordered categorical response

Today

- Poisson probability model
- Poisson regression model, estimation, and statistical inference
- Model Comparison Criteria, Model Assessment, Goodness of Fit

Looking Ahead

- Univariate and multivariate time-series
- Notion of dependency and stationarity

Start-up Code

```
# Insert the function to *tidy up* the code when they are printed out
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

# Start with a clean R environment
rm(list = ls())

# Load libraries
## Load a set of packages inclusing: broom, cli, crayon, dbplyr , dplyr, dtplyr, forcats,
## googledrive, googlesheets4, ggplot2, haven, hms, httr, jsonlite, lubridate , magrittr,
## modelr, pillar, purrr, readr, readxl, reprex, rlang, rstudioapi, rvest, stringr, tibble,
## tidyr, xml2
if(!"tidyverse"%in%rownames(installed.packages())) {install.packages("tidyverse")}
library(tidyverse)

## provide useful functions to facilitate the application and interpretation of regression analysis.
if(!"car"%in%rownames(installed.packages())) {install.packages("car")}
library(car)

## provides  many functions useful for data analysis, high-level graphics, utility operations like describe()
if(!"Hmisc"%in%rownames(installed.packages())) {install.packages("Hmisc")}
library(Hmisc)

##Tools for performing model selection like AICc()
if(!"gamlr"%in%rownames(installed.packages())) {install.packages("gamlr")}
library(gamlr)

## To generate regression results tables and plots
if(!"finalfit"%in%rownames(installed.packages())) {install.packages("finalfit")}
library(finalfit)

## To produces LaTeX code, HTML/CSS code and ASCII text for well-formatted tables
if(!"stargazer"%in%rownames(installed.packages())) {install.packages("stargazer")}
library(stargazer)

## For overdispersion test
if(!"AER"%in%rownames(installed.packages())) {install.packages("AER")}
```

```r
library(AER)

## For negative binomial regression
if(!"MASS"%in%rownames(installed.packages())) {install.packages("MASS")}
library(MASS)
```

Poisson Distribution Review

Recall that the Poisson distribution models count data i.e. the number of events between 0,1,... for a random variable X. The distribution is:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

One key assumption of the distribution is that $E(X) = Var(X) = \lambda$, meaning that the mean and variance of the distribution is the same. As we will see this is a limiting assumption when we do Poisson Regression.

Poisson Regression Review

In Poisson regression we model the log of λ (the mean) as a linear combination of the features:

$$log(\lambda_i) = log(E(Y_i|X_i)) = X_i\beta$$

We use maximum likelihood to estimate the coefficients in β assuming that Y follows a Poisson distribution:

$$max_\beta \ L(\beta|Y_1, ..., Y_n) = P(Y_1 = y_1, ..., Y_n = y_n|\beta) = \Pi_i \frac{e^{y_i X_i \beta} e^{-e^{X_i\beta}}}{y_i!}$$

The log likelihood has no close form solution, we estimate the parameters beta using numerical methods just like in logistic regression.

For each y_i we calculate and predict:

$$\hat{y_i} = E(y_i|X_i) = \lambda_i = e^{X_i\beta}$$

This also means that the $Var(\hat{y_i}) = \lambda_i = e^{X_i\beta}$ so that Poisson regression naturally has heteroskedasticity in the results.

This assumption of equal mean and variance is often not met when actually fitting to data, which results in what is known as overdispersion where the variance in the data is larger than the variance fit in the model. This usually can be remedied by adding more X variables into the model to improve the fit.

Another option is to fit what are known as a quasi poisson model or negative binomial regression model. In both cases, we relax the equal mean and variance assumption by adding an additional parameter to the variance of the response variable, allowing it to be larger than the mean. In quasi poisson regression we set $Var(\hat{y_i}) = \theta\lambda_i$ and in negative binomial regression we set $Var(\hat{y_i}) = \lambda_i + \kappa\lambda_i^2$.

Also, because coefficients are on the log scale, when exponentiated they multiply the expected mean outcome. This is similar to the interpretation in logistic regression except we are not dealing with odds ratios but rather the average outcome.

Case Study: Modeling the Number of Awards

Introduction

Imagine we are trying to model the number of awards earned by students based on the type of programs the student was enrolled in using historical admission data. The awards committee provides a small data sample that includes the score of the final math exam in previous years.

Data Description

The dataset *PossionEx1.csv* contains the following variables:

- num_awards: the number of awards earned by students at a high school in a year
- math: students' scores on their final math exam
- prog: the type of program in which the students were enrolled (1 = "General", 2 = "Academic" and 3 = "Vocational").

Descriptive Statistics

Some questions to ask when exploring the dataset:

- What is the number of observations?
- What is the number of variables?
- Are there any redundant variables?
- Are there any missing information?
- Are there any duplicated records?
- Are there any values in each of the variables that seem unreasonable?

```
df <- read.csv("./data/PossionEx1.csv", stringsAsFactors = F, header=TRUE, sep=",")

head(df) %>%
  knitr::kable()
```

X	id	num_awards	prog	math
1	45	0	Vocational	41
2	108	0	General	41
3	15	0	Vocational	44
4	67	0	Vocational	42
5	153	0	Vocational	40
6	51	0	General	42

```r
#str(df)

## convert prog to factor
df$prog = factor(df$prog)


# Checking the number of missing values for each of the variables
# df[!complete.cases(df),]
sapply(df, function(x) sum(is.na(x)))
```

```
##          X         id num_awards       prog       math
##          0          0          0          0          0
```

```r
# Attach the dataset
attach(df)

#describe(df)
```

Univariate Analysis

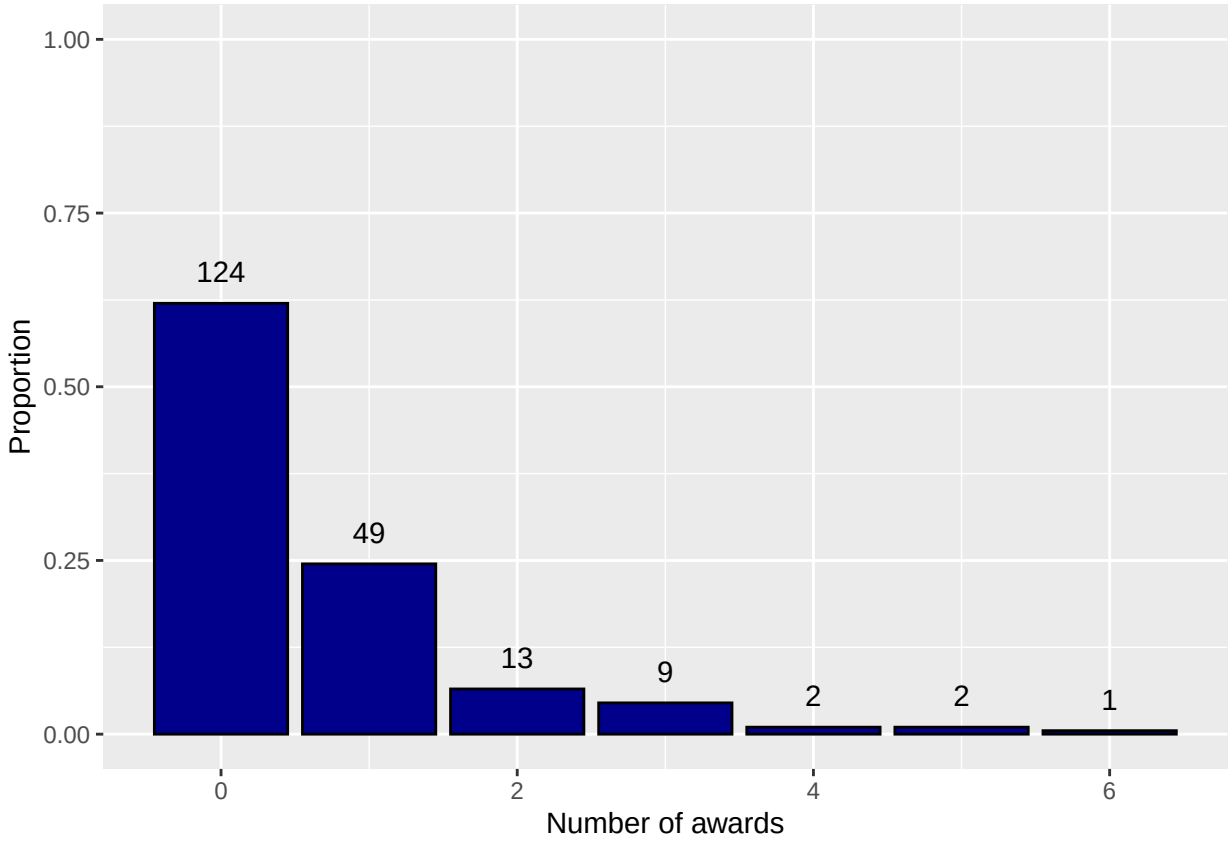
- Use a frequency table and a bar plot to explore the distribution of the response variable(num_awards). What do you learn?

```
df %>%
  count(num_awards) %>%
  mutate(prop = round(prop.table(n),2)) %>%
  kable(col.names = c('Number of awards', 'N', "Proportion"))
```

Number of awards	N	Proportion
0	124	0.62
1	49	0.24
2	13	0.06
3	9	0.04
4	2	0.01
5	2	0.01
6	1	0.00

```
df %>%
  ggplot(aes(x= num_awards, y = ..prop.., group = 1)) +
  geom_bar(fill = 'DarkBlue', color = 'black') +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  xlab("Number of awards") +
  ylab("Proportion") +
  ylim(0,1)
```

```
## Warning: The dot-dot notation (‘..prop..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(prop)‘ instead.
```

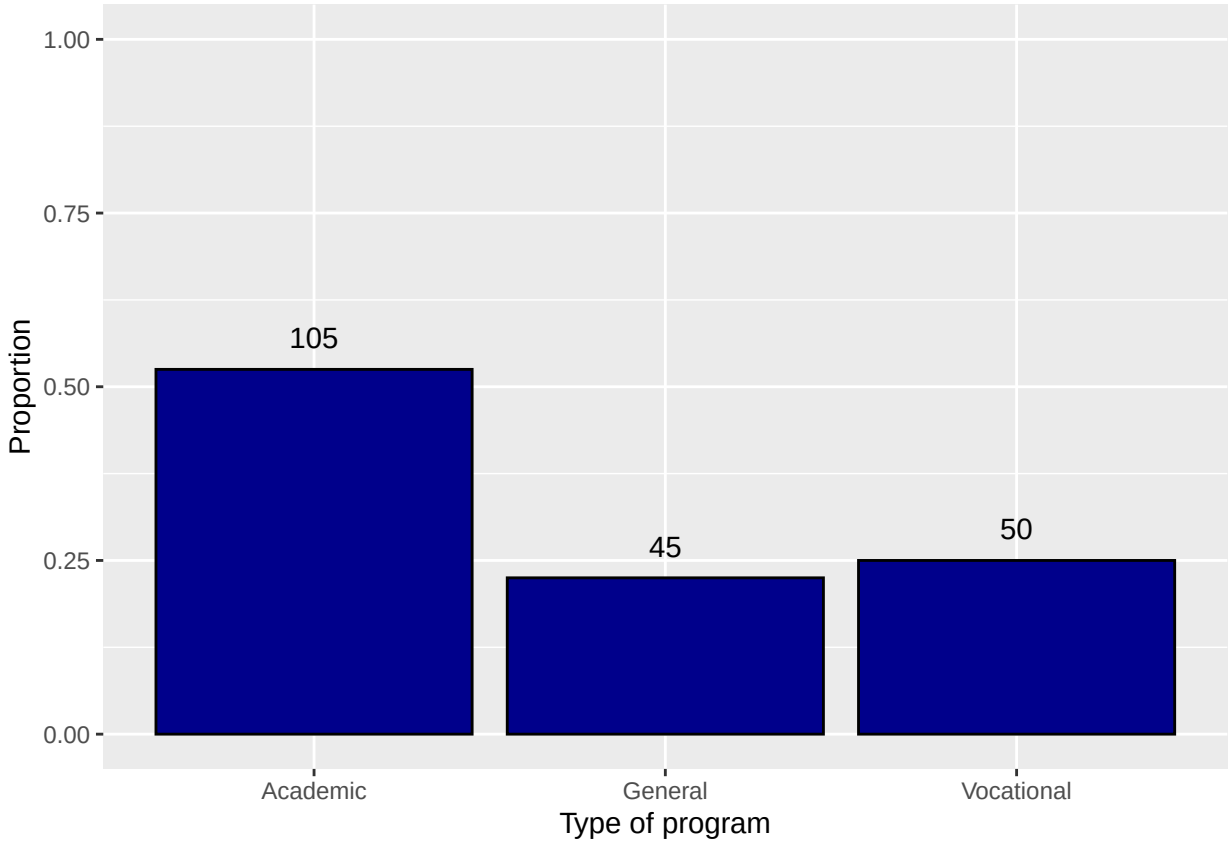

1.00
0.75
0.50
0.25
0.00
Proportion
124
49
13
9
2
2
1
0
2
4
6
Number of awards

- The prog is the committee's key explanatory variable of interest. It has three levels: academic, general, and vocational. Use a frequency table and a bar plot to examine its distribution. What do you discover?

```
df %>%
  count(prog) %>%
  mutate(prop = round(prop.table(n),2)) %>%
  kable(col.names = c(' Type of program', 'N', "Proportion"))
```

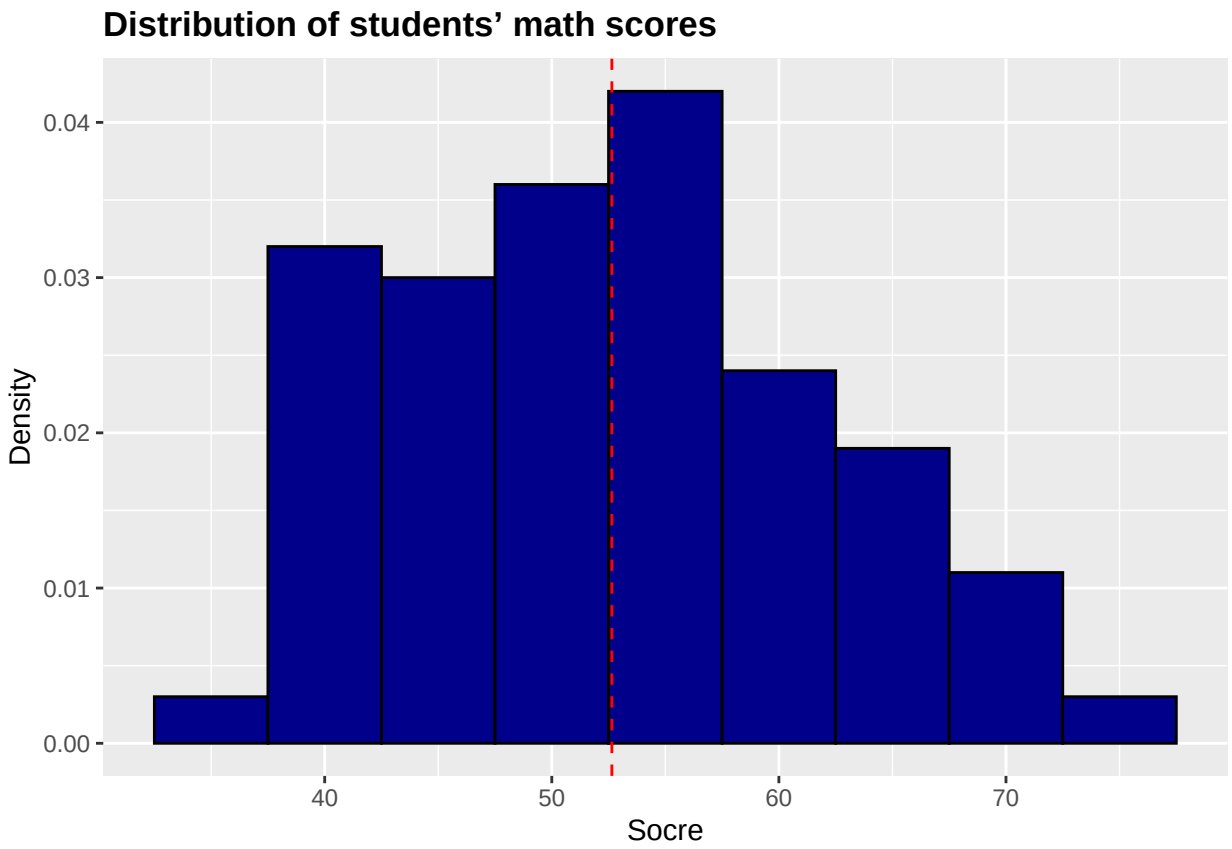
Type of program	N	Proportion
Academic	105	0.52
General	45	0.22
Vocational	50	0.25

```
df %>%
  ggplot(aes(x= prog, y = ..prop.., group = 1)) +
  geom_bar(fill = 'DarkBlue', color = 'black') +
  geom_text(stat='count', aes(label=..count..), vjust=-1) +
  xlab("Type of program") +
  ylab("Proportion") +
  ylim(0,1)
```



- Plot the distribution of math scores. What are the range and average math scores?

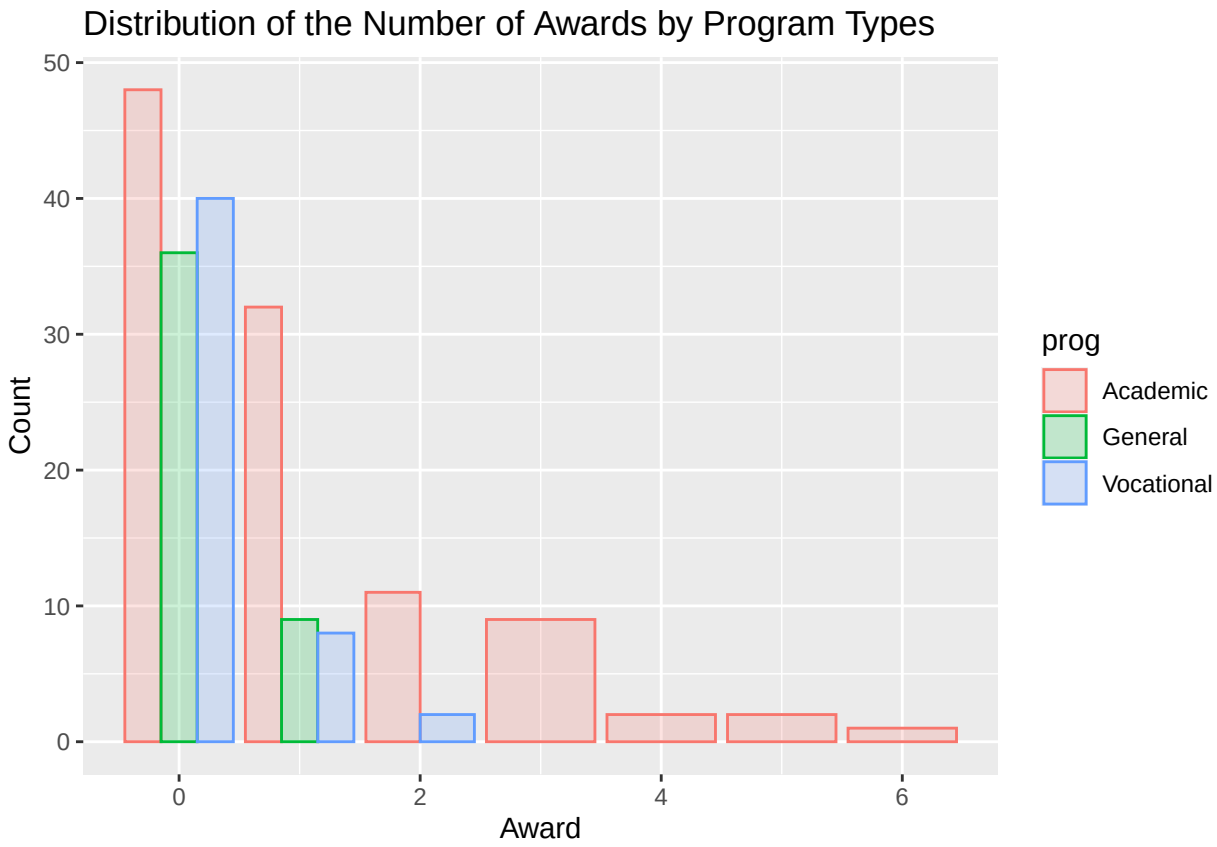
```
df %>%
  ggplot(aes(x = math)) +
  geom_histogram(aes(y = ..density..), binwidth = 5 ,fill = "DarkBlue", color = "black") +
  geom_vline(aes(xintercept = mean(math)), color = "red", linetype = "dashed") +
  ggtitle("Distribution of students' math scores") +
  theme(plot.title = element_text(lineheight=1, face="bold")) +
  xlab("Socre") +
  ylab("Density")
```


Distribution of students' math scores
Density
0.04
0.03
0.02
0.01
0.00
40
50
60
70
Socre

Bivariate Analysis

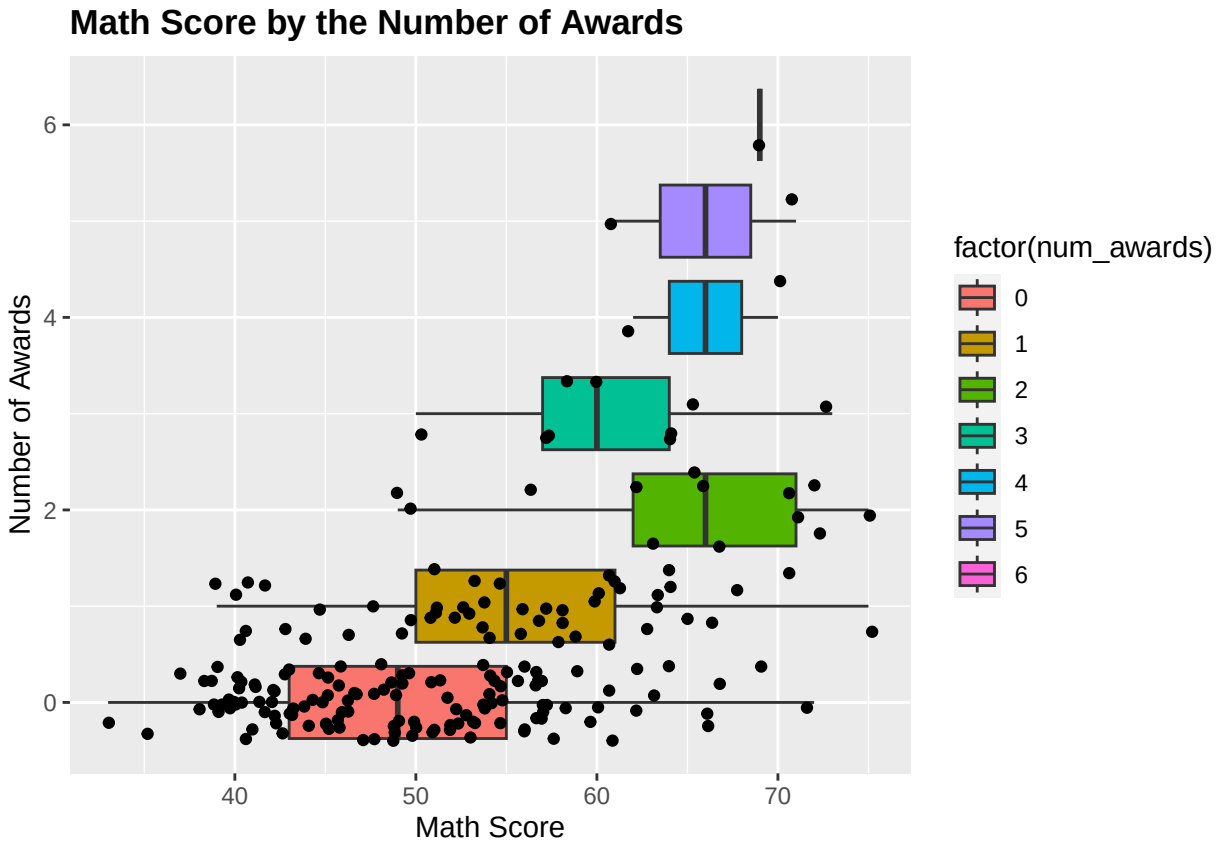
- Examine the associations between the number of awards and program and math scores.
- The graph below shows the distribution of the number of awards by program types. How are awards distributed among different programs?

```
df %>%
  ggplot(aes(x = num_awards)) +
  geom_bar(aes(color = prog, fill = prog),alpha=0.2, position = "dodge" ) +
  ggtitle("Distribution of the Number of Awards by Program Types") +
  xlab("Award") +
  ylab("Count")
```



- The graph below shows the distribution of the number of awards and students' math scores. Is there any clear relationship between them?

```
df %>%
  ggplot(aes(num_awards, math)) +
  geom_boxplot(aes(fill = factor(num_awards))) +
  geom_jitter()+
  coord_flip() +
  ggtitle("Math Score by the Number of Awards") +
  theme(plot.title = element_text(lineheight=1, face="bold")) +
  xlab("Number of Awards") +
  ylab("Math Score")
```



- Use summary_factorlist() function from the finalfit package to tabulate data. What do you learn from the EDA?

```
dependent <- "num_awards"
explanatory <- c("prog", "math")
df %>%
  summary_factorlist(dependent, explanatory, add_dependent_label = TRUE) %>%
  knitr::kable()
```

Dependent: num_awards		unit	value
prog	Academic	Mean (sd)	1.0 (1.3)
	General	Mean (sd)	0.2 (0.4)
	Vocational	Mean (sd)	0.2 (0.5)
math	[33.0,75.0]	Mean (sd)	0.6 (1.1)

Model Development

- Given the specification of a poisson regression model below,

$$\log(mean\_num\_awards) = \beta_0 + \beta_1 prog + \beta_2 math + u$$

- Estimate and interpret the model results:

```
#poisson.mod.1 <- # uncomment and replace with your code
#summary(poisson.mod.1) # uncomment
```

- Construct and interpret the confidence intervals for each variable.

```
# Confident intervals for the original coefficient estimates

# Replace with your code

# Convert the confidence intervals to percentage change, corresponding to the coefficient estimates

# Replace with your code
```

- Test the overall effect of prog,

```
## test model differences with chi square test

#Anova() # uncomment and replace with your code
```

- Plot the fitted values across the three programs and discuss how the number of awards is associated with math scores.

```
# uncomment and run the code

## calculate and store predicted values
#fitted_values <- predict(poisson.mod.1, type="response")

## create the plot

#ggplot(df, aes(x = math, y = fitted_values, colour = prog)) +
#  geom_point(aes(y = num_awards), alpha=.5, position=position_jitter(h=.2)) +
#  geom_line(size = 1) +
#  labs(x = "Math Score", y = "Expected number of awards")
```

Directly Testing for Over Dispersion

We can use a dispersion test for Poisson regression from the AER package that tests the null hypothesis that $\theta = 1$ vs. not in a regression of the form $Var(Y_i) = (1 + \alpha) * \lambda_i$.

Note that if we set $(1 + \alpha) = \theta$ we get the variance form for a quasipoisson above. So this test is examining whether the variance of our outcome variable appears to come from a Poisson or quasipoisson distribution.

If we reject the null hypothesis due to a small p-value, we have overdispersion if $\alpha > 0$ and underdispersion (smaller variance in reality which is less common) if $\alpha < 0$.

The test itself reports the estimated dispersion value along with a p-value.

```
# replace with your code
```

If we reject the null hypothesis in an overdispersion test, that means we should fit a quasipoisson or negative binomial regression.

We can fit quasipoisson directly using glm and specifying the appropriate family. For a negative binomial using regression, we need to use glm.nb from the MASS package.

```
# quasipoisson regression

# quasi.poisson <- uncomment and replace with your code
```

```
# negative binomial regression

# neg.binom <- uncomment and replace with your code
```

```
# stargazer(poisson.mod.1, quasi.poisson, neg.binom, type="text")
```

Model Comparison Criteria

Recall that the general form of most information criteria is:

$$IC(k) = -2log(L(\hat{\beta}|y_1, ..., y_n)) + kr$$

Where $log(L(\hat{\beta}|y_1, ..., y_n))$ is the log-likelihood of an estimated model, n is the sample size, r is the number of parameters in the model, and k is a penalty term on the number of parameters.

The three most common information criteria are:

$$AIC = IC(2) = -2log(L(\hat{\beta}|y_1, ..., y_n)) + 2r$$

$$AIC_c = IC(\frac{2n}{n-r-1}) = -2log(L(\hat{\beta}|y_1, ..., y_n)) + r\frac{2n}{n-r-1} = AIC + \frac{2r(r+1)}{n-r-1}$$

$$BIC = IC(log(n)) = -2log(L(\hat{\beta}|y_1, ..., y_n)) + rlog(n)$$

- Compute these three information criteria for the following three models and then rank the models based on each criterion.

$$\log(mean\_num\_awards) = \beta_0 + \beta_1 prog + u$$
$$\log(mean\_num\_awards) = \beta_0 + \beta_1 math + u$$

$$\log(mean\_num\_awards) = \beta_0 + \beta_1 prog + \beta_2 math + u$$

```
## fit models
#mod1 <-
#mod2 <-
#mod3 <-

## compute AIC

## compute corrected AIC

## compute BIC
```

Model Assessment

Recall that the Pearson residuals correct for unequal variance in the raw residuals by dividing by the standard deviation:

$$e_m = \frac{y_m - \widehat{y_m}}{\sqrt{\widehat{Var(Y_m)}}}$$

Standardized Pearson residuals also correct for overestimates of the standard deviation of $y_m - \widehat{y_m}$:

$$r_m = \frac{y_m - \widehat{y_m}}{\sqrt{\widehat{Var(Y_m - \hat{Y_m})}}} = \frac{y_m - \widehat{y_m}}{\sqrt{\widehat{Var(Y_m)} - (1 - h_m)}}$$

where h_m is the mth diagonal element of the hat matrix.

- For the first Poisson model using prog and math as predictors, plot the standardized Pearson residuals against explanatory variables, fitted values, and the linear predictor to assess whether the model assumptions are satisfied.

```
#pred <- # uncomment and replace with your code
#res <- # uncomment and replace with your code
#s.res <- # uncomment and replace with your code
#lin.pred <- # uncomment and replace with your code

#df1 <- data.frame(df, pred ,res ,s.res , lin.pred)

#Standardized Pearson residual vs math plot

#df1 %>%
#  ggplot(aes(x = df1$math , y = df1$s.res)) +
#  geom_point() +
#  geom_hline(yintercept=c(3, 2, 0, -2, -3), color = "red", linetype = "dashed")+
#  geom_smooth(se = FALSE)+
#  ggtitle("Standardized residuals vs. Math") +
#  xlab("Math") +
#  ylab("Standardized Pearson residuals")

#Standardized Pearson residual vs fitted values

#df1 %>%
#  ggplot(aes(x = df1$pred , y = df1$s.res)) +
```

```
#  geom_point() +
#  geom_hline(yintercept=c(3, 2, 0, -2, -3), color = "red", linetype = "dashed")+
#  geom_smooth(se = FALSE)+
#  ggtitle("Standardized residuals vs. Math") +
#  xlab("Fitted values") +
#  ylab("Standardized Pearson residuals")

#Standardized Pearson residual vs linear predictor

#df1 %>%
#  ggplot(aes(x = df1$lin.pred , y = df1$s.res)) +
#  geom_point() +
#  geom_hline(yintercept=c(3, 2, 0, -2, -3), color = "red", linetype = "dashed")+
#  geom_smooth(se = FALSE)+
#  ggtitle("Standardized residuals vs. Math") +
#  xlab("Linear predictor") +
#  ylab("Standardized Pearson residuals")
```

Goodness of Fit

The Pearson Statistic χ^2 and Residual Deviance D are often used to test the goodness of fit, where the null hypothesis is that our model is correct. Under asymptotic theory, both of these follow a chi-squared distribution with the same degrees of freedom as the residuals from the Poisson model.

We can also use these to test for overdispersion in our model since if our model is a good fit to the data we should not have overdispersion.

Recall:

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - \exp\{\mathbf{X}_i\hat{\beta}\})^2}{\exp\{\mathbf{X}_i\hat{\beta}\}}$$

and

$$D = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\exp\{\mathbf{X}_i\hat{\beta}\}}\right) - (y_i - \exp\{\mathbf{X}_i\hat{\beta}\})\right]$$

- Calculate the Pearson statistic and residual deviance and perform the test of goodness of fit using model 3 from above (with math and program as predictors).

```
# Calculate Pearson statistic residuals

# Replace with your code

# Get p value associated with the pearson statistic

# Replace with your code

#Calculate deviance p value

# Replace with your code
```

Reminders

1. Before next live session:
    1. Complete and turn in the Lab-1
    2. Complete all videos and reading for unit 6