# Generalized Linear Models

Martin Endler endlemar@fel.cvut.cz

## Introduction

The aim of this assignment is to practice constructing linear models. You will start with a simple linear model. You will evaluate and interpret it (1p). Consequently, your task will be to improve this model using generalized linear models (GLMs) and feature transformations. You will get 1p for proposal and evaluation of GLM (family, evaluation, interpretation), 1p for correct feature transformations, 1p for proposal and justification of the final model and eventually, 1p for comprehensive evaluation of all the model improvements (ablation study through cross-validation, note that the previous evaluations must be done without cross-validation).

## Input data

In this assignment, you will work with a student dataset. The dataset contains 200 samples and 4 features: *num_awards* is the outcome variable and indicates the number of awards earned by students in a year, *math* is a continuous predictor variable and represents students' scores on their math final exam, *prog* is a categorical predictor variable with three levels indicating the type of program in which the students were enrolled (1 = "General", 2 = "Academic" and 3 = "Vocational"), and *work* is a continuous predictor that gives the number of hours that students spent at work on average per week.

## Load the necessary libraries and the dataset

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

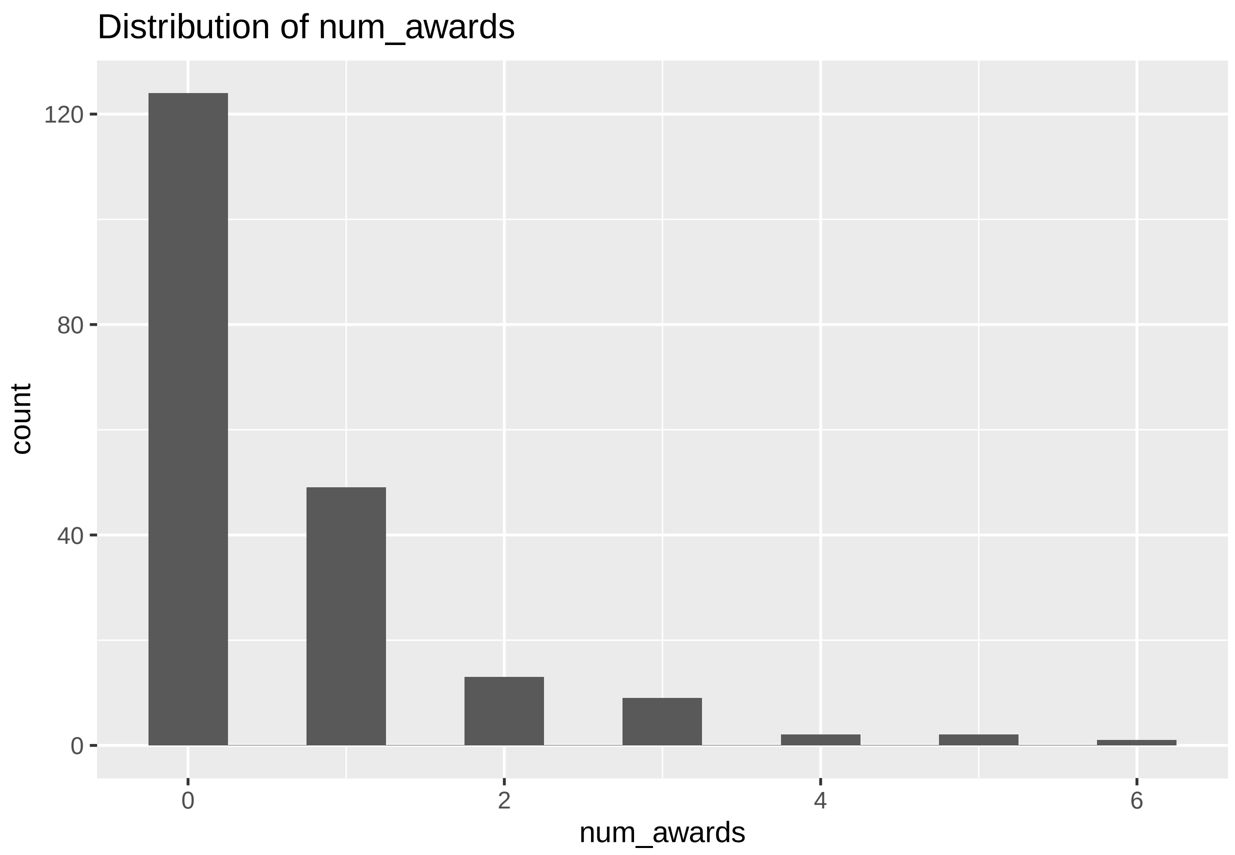
```r
library(caret) # comprehensive model evaluation
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(splines)
library(ggplot2) # visualizations
library(rcompanion) # comparison of GLMs

d <- read.csv("study_data.csv")
prog_labels <- c("1: General", "2: Academic", "3: Vocational")
```

## A few preliminary explorations

```
# distribution of num_awards in our data
# (so that we have an idea of what we are dealing with)
# num_awards is a discrete outcome (aka reponse or depdenent) variable
# that represents count data. It is always non-negative.
d %>%
    ggplot(aes(x = num_awards)) +
    geom_histogram(binwidth = 0.5, center = 0) +
    ggtitle("Distribution of num_awards")
```



```
# d2 - copy of data (d) where prog is a factor
typeof(d$prog)
```

```
## [1] "integer"
```

```
is.factor(d$prog)
```

```
## [1] FALSE
```

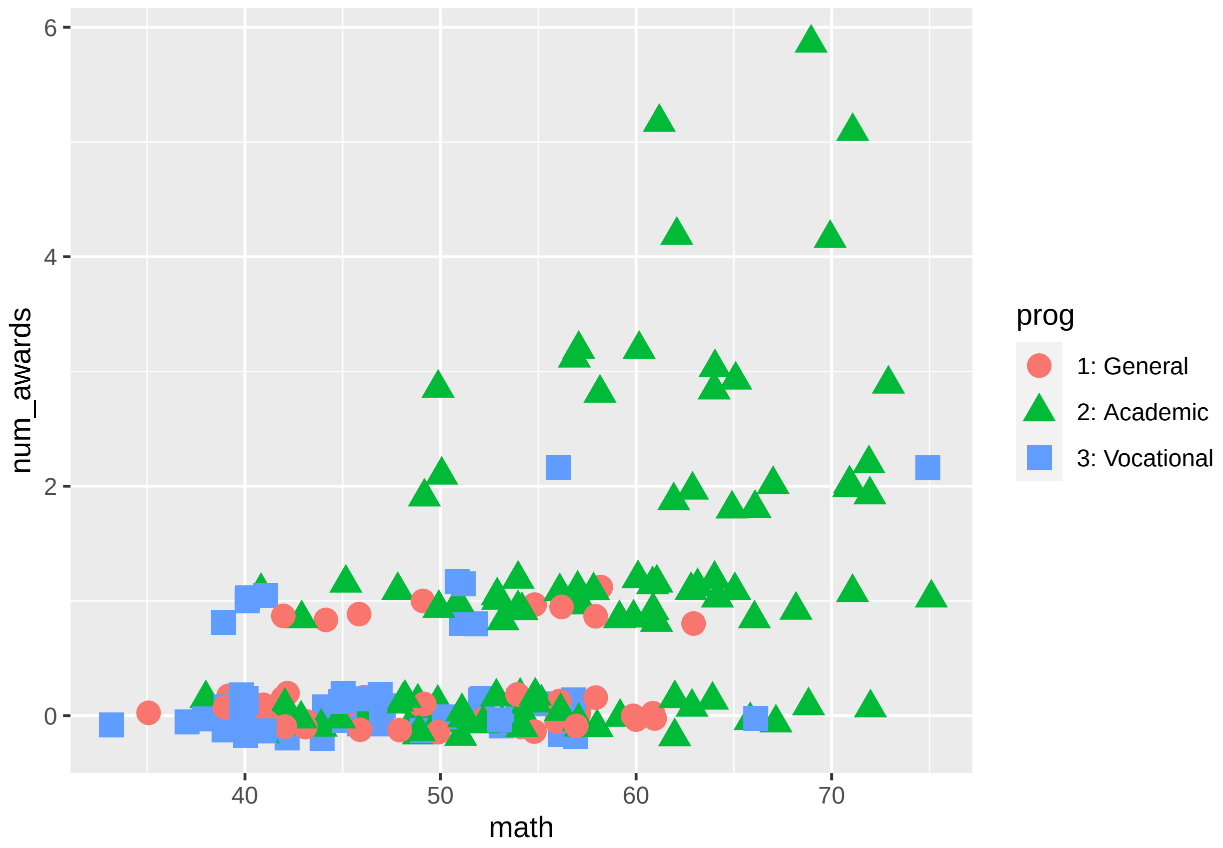
```
# d2 <- d %>% mutate(prog = as.factor(prog))
d2 <- d %>% mutate(prog = factor(prog, labels = prog_labels))
typeof(d2$prog)
```

```
## [1] "integer"
```

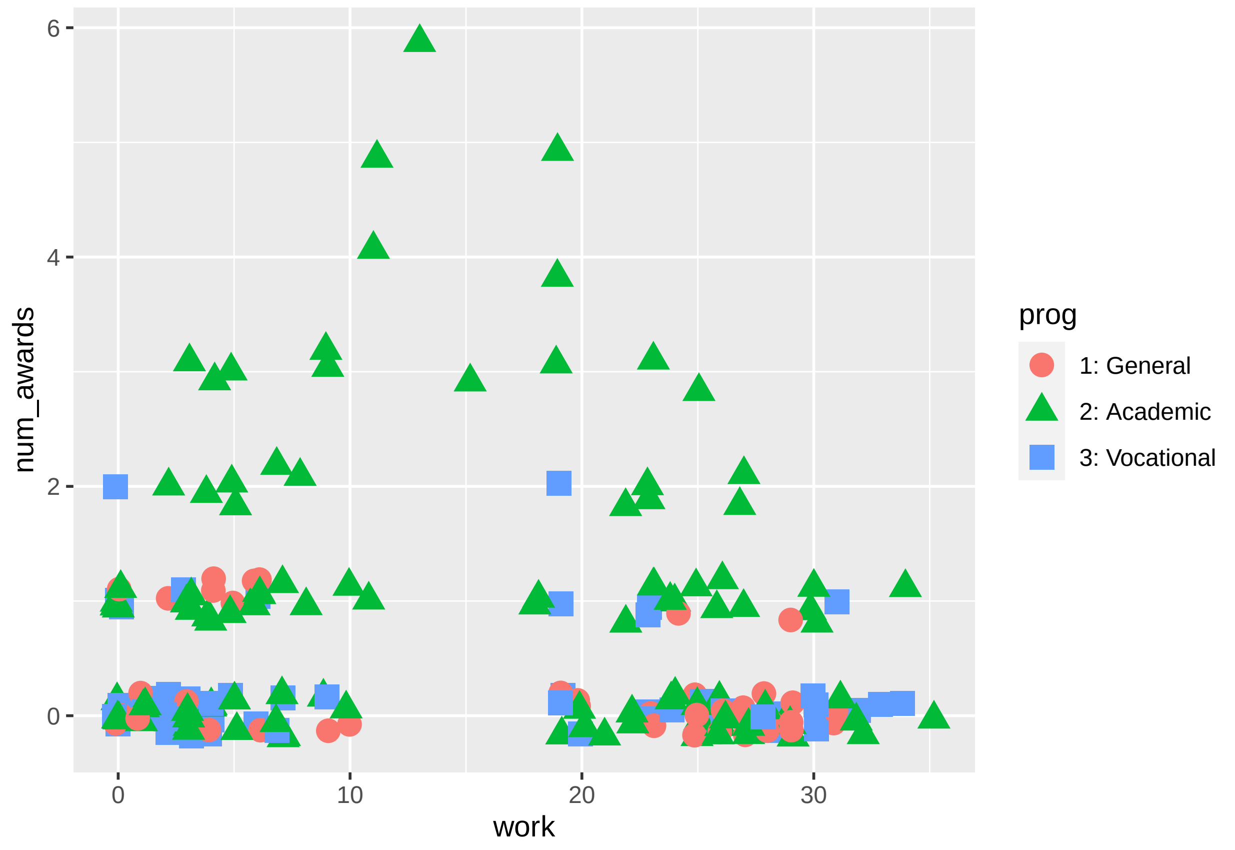
```
is.factor(d2$prog)
```

```
## [1] TRUE
```

```
# relationship between math score and num_awards (also considering the program)
d2 %>%
    ggplot(aes(x = math, y = num_awards, shape = prog, color = prog)) +
    geom_jitter(size = 4, width = 0.2, height = 0.2)
```



```
# relationship between work (the number of hours that student spent at work on average per week)
# and num_awards (also considering the program)
d2 %>%
    ggplot(aes(x = work, y = num_awards, shape = prog, color = prog)) +
    geom_jitter(size = 4, width = 0.2, height = 0.2)
```



```
# covariation and correlation matrices might be useful
d.cov <- cov(d)
d.cor <- cor(d)
```

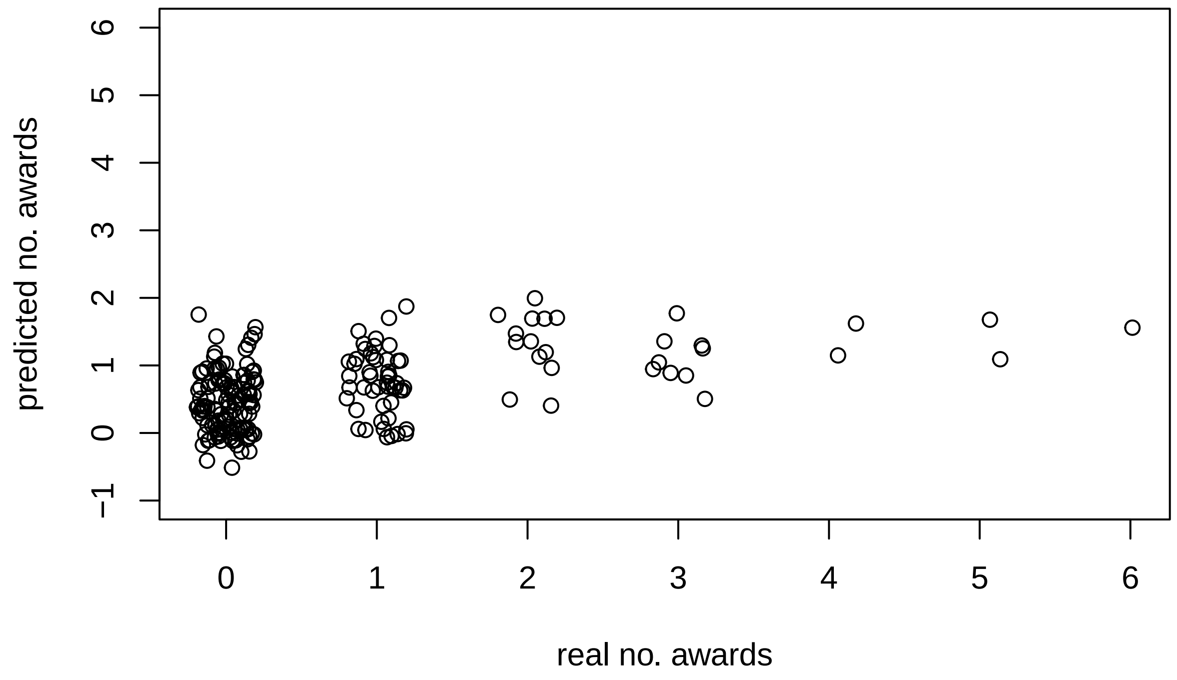
## Simple linear model

Let us start with an ordinary linear model with no feature transformations. Explain how far the model works (does it meet formal assumptions?, does it overcome the null model?). Which predictors would you keep there and which of them are not useful? Use the standard evaluation procedures that we have for linear models.

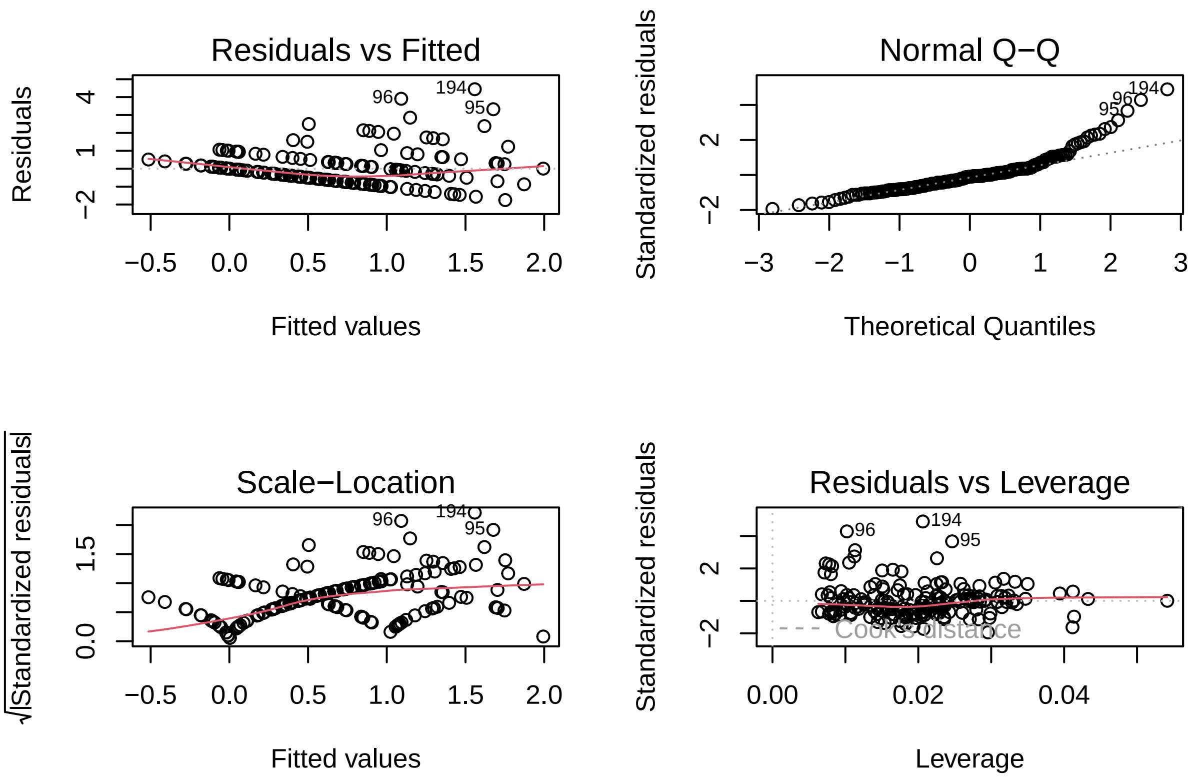
```
simple_lm <- lm(num_awards ~ ., d)
summary(simple_lm)
```

```
##
## Call:
## lm(formula = num_awards ~ ., data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7526 -0.5680 -0.0862  0.2979  4.4411
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.536344   0.481442  -5.268 3.61e-07 ***
## prog         0.109815   0.095972   1.144    0.254
## math         0.056617   0.007130   7.941 1.53e-13 ***
## work        -0.002379   0.005807  -0.410    0.682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9168 on 196 degrees of freedom
## Multiple R-squared:  0.2533, Adjusted R-squared:  0.2418
## F-statistic: 22.16 on 3 and 196 DF,  p-value: 2.133e-12
```

```
simple_lm_preds <- predict(simple_lm, d)

# What are the shortcomings of the simple regression?
# Among other things, it might predict a negative number of awards which makes no sense.
# But we could just consder all values below 0 as 0.
plot(
    x = d$num_awards + runif(nrow(d), -0.2, 0.2),
    y = simple_lm_preds,
    xlab = 'real no. awards',
    ylab = 'predicted no. awards',
    ylim = c(-1, 6)
)
```



```
# lm debug plots
par(mfrow = c(2, 2))
plot(simple_lm)
```



```r
# Refer to the lectures what this call measures
anova(lm(num_awards ~ math + prog + work, d))
```

```
## Analysis of Variance Table
##
## Response: num_awards
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## math        1  54.509  54.509 64.8496 7.69e-14 ***
## prog        1   1.222   1.222  1.4536   0.2294
## work        1   0.141   0.141  0.1679   0.6824
## Residuals 196 164.748   0.841
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**Add your verbal summary here (1p):**

When we fitted an ordinary linear model using `lm` we got the following model:

$$num\_awards = -2.536 + 0.110 * prog + 0.057 * math - 0.002 * work$$

First, from the model summary, we can tell that **our fitted model is significantly better than the intercept-only model** *(using F-test, based on the value of F-statistic, which is 22.16 on 3 and 196 DF, we reject the null hypothesis (p-value « 0.05) that the intercept-only model fits the data as well as our model).*

Before interpreting the model, we should check whether the formal assumptions (noise normality, linearity, homoscedasticity) of OLS are met. Residual plots and the Q-Q plot are useful for this. From the "Residuals vs Fitted" plot, we can note the **homoscedasticity is most probably violated** (variance is not constant, i.e., we have heteroscedasticity): as we move to the right on the x-axis, the spread of the residuals seems to be increasing (and further, the spread is not symmetrical about the x-axis). Next, from the "Normal Q-Q" plot, we can tell that the noise **normality assumption is a bit violated**. All points should be on (or close to) the dashed line, but in our case, the points in the right part of the plot are shifted to the top. This particular shape expresses the fact that the data (their error dist.) are skewed on one side. These violations are no surprise since `num_awards` represents **count data**. Count data are characterized by heteroscedasticity and their error distribution is skewed.

Additionally, the model might also **predict a negative number of awards** which makes no sense.

To sum up, the simple linear regression (OLS) is not a good fit for our data, so we need to be careful when interpreting or trusting the fitted model.

Nevertheless, let's continue with evaluating our fitted model.

Adjusted R-squared is only 0.2418 which means our model explains only 24 % of the variance in the data.

Looking at the coefficients table, we can see that only the **math** predictor is significant. By increasing a math score by one point, the number of awards increases by 0.057. This makes sense as students with higher math exam scores could be more likely to receive a higher number of awards.

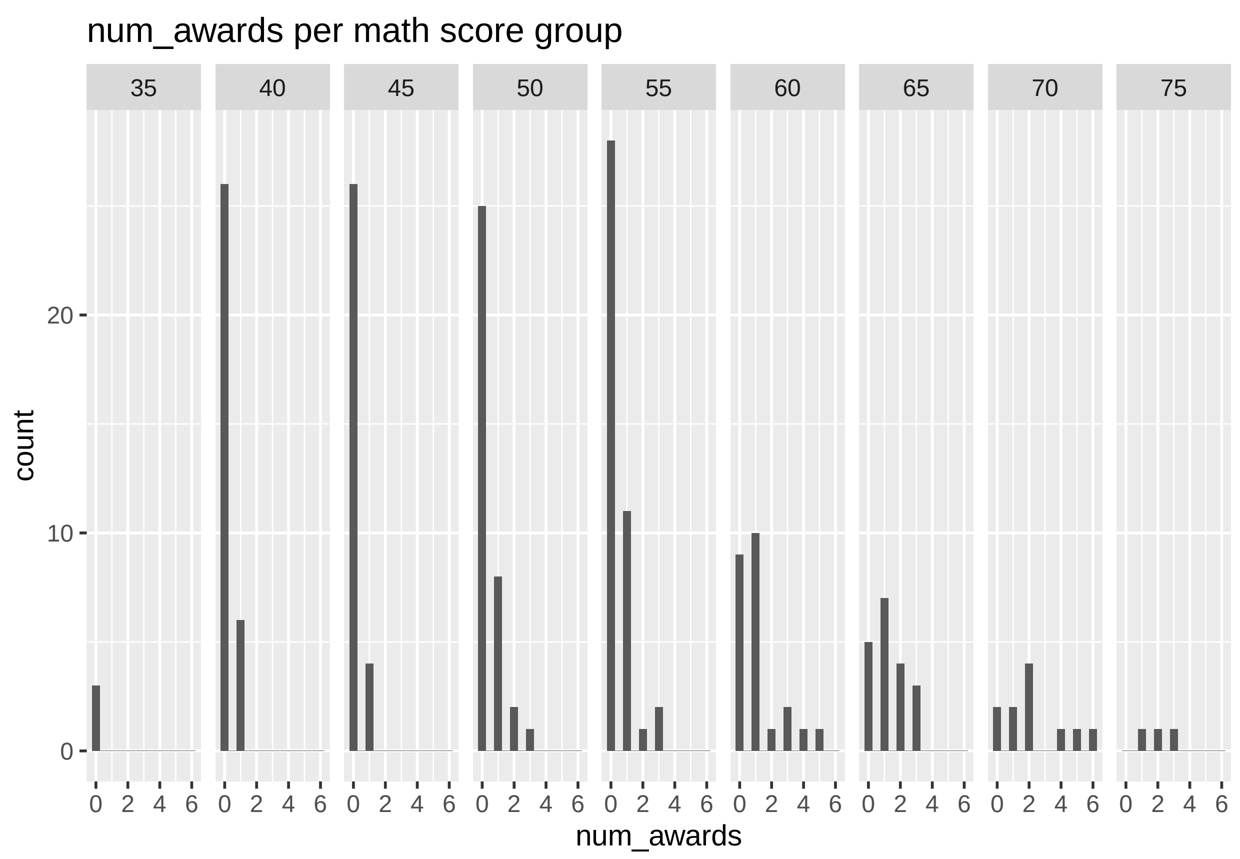
Also, by looking at ANOVA table, we would say that only **math** predictor is worth keeping in our model.

## Generalized linear model

Now, the goal is to implement a generalized linear model that conceptually fits the given task. Do not transform the predictors yet, use them as they are, or omit them from the model. Once you obtain your model, interpret the effect of the *math* predictor on the outcome. How (according to your model) increasing a math score by one point affects the number of awards won?

Explain why the model overcomes the previous linear model, or justify that the generalized model is not needed. Compare the models theoretically as well as technically in terms of a proper quality measure. Note: The difference between the models cannot be statistically tested.

```r
# # Step 1:
# num_awards is a discrete outcome (aka reponse or depdenent) variable
# that represents count data. It is always non-negative.
# Let's assess the applicability of Poisson regression.
# Slice the data at each math-points-group and observe the distribution of the count response (num_awar
bin_size <- 5 # math points
d %>%
    mutate(bin = round(math / bin_size) * bin_size) %>%
    ggplot(aes(x = num_awards)) +
    geom_histogram(binwidth = 0.5, center = 0) +
    facet_grid(cols = vars(bin)) +
    ggtitle("num_awards per math score group")
```



```
# # Step 2:
# Let's try the Poisson regression, which should be a better fit than OLS
# - we have count data and its distribution seems to be Poisson.
# Use only math predictor.
poiss_v1_glm <- glm(num_awards ~ math, family = "poisson", data = d)
print('poiss_v1_glm:')
```

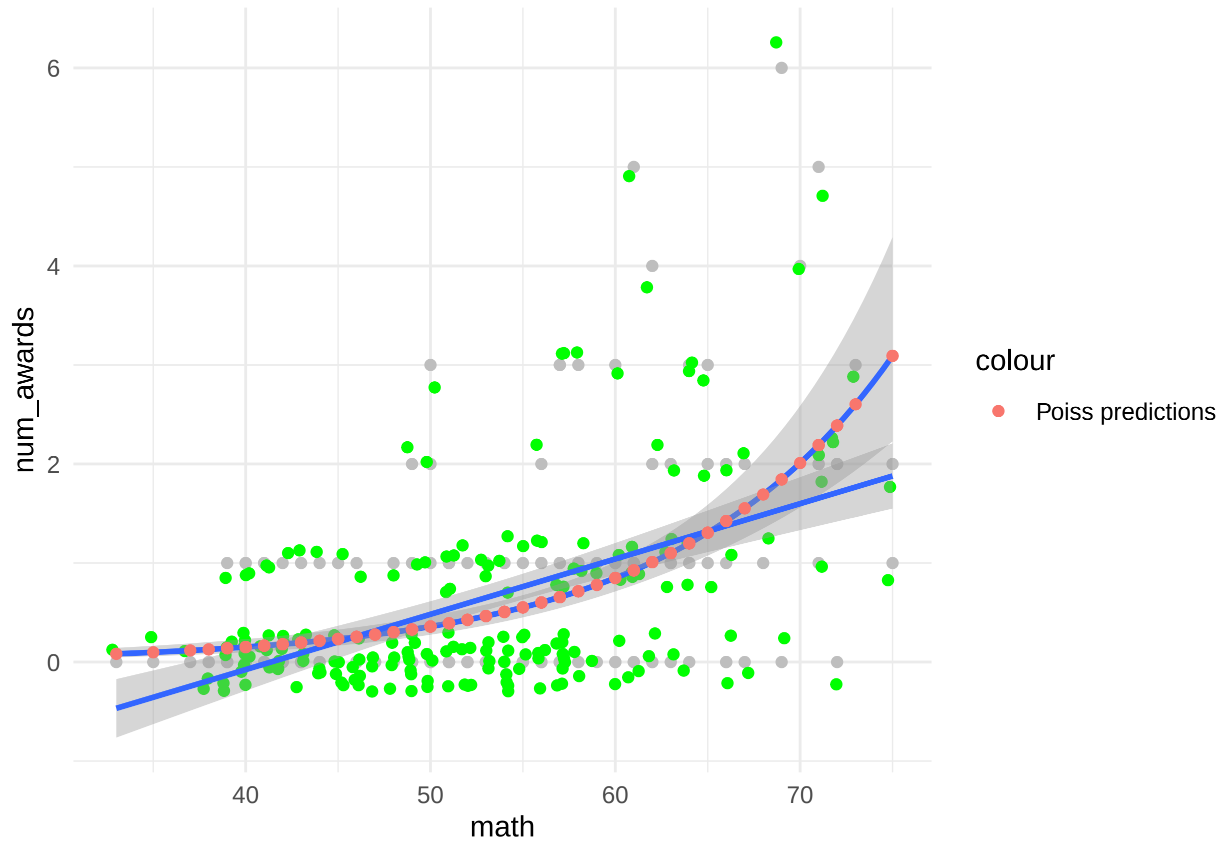
```
## [1] "poiss_v1_glm:"
```

```
summary(poiss_v1_glm)
```

```
##
## Call:
## glm(formula = num_awards ~ math, family = "poisson", data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1853  -0.9070  -0.6001   0.3246   2.9529
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.333532   0.591261  -9.021   <2e-16 ***
## math         0.086166   0.009679   8.902   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
```

```
##
##     Null deviance: 287.67  on 199  degrees of freedom
## Residual deviance: 204.02  on 198  degrees of freedom
## AIC: 384.08
##
## Number of Fisher Scoring iterations: 6
```

```
# predict
poiss_v1_glm_preds <- predict(poiss_v1_glm, d, type = "response")
poiss_v1_glm_preds_r <- round(poiss_v1_glm_preds)

# vizualize
d %>%
    mutate(preds = poiss_v1_glm_preds) %>%
    ggplot(aes(x = math, y = num_awards)) +
    geom_point(size = 1.5, color = 'gray') +
    geom_jitter(size = 1.5, width = 0.3, height = 0.3, color = 'green') +
    # ggplot can do glm by itself
    geom_smooth(method = "glm", formula = y ~ x, method.args = list(family = "poisson")) +
    geom_smooth(method = "glm", formula = y ~ x, method.args = list(family = "gaussian")) +
    # but let's also vizualize the predictions manually
    geom_point(aes(x = math, y = poiss_v1_glm_preds, color = "Poiss predictions")) +
    theme_minimal()
```



```
# Step 3: Compare with the previous simple linear regression model
# We need some common ground for comparison,
# so let's fit the linear regression again, but this time with the glm() function
```

```r
# (family = "gaussian" is equivalent to the simple linear regression).
simple_glm_full <- glm(num_awards ~ ., family = "gaussian", data = d)
print('simple_glm_full:')
```

```
## [1] "simple_glm_full:"
```

```r
summary(simple_glm_full)
```

```
##
## Call:
## glm(formula = num_awards ~ ., family = "gaussian", data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7526  -0.5680  -0.0862   0.2979   4.4411
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.536344   0.481442  -5.268 3.61e-07 ***
## prog         0.109815   0.095972   1.144    0.254
## math         0.056617   0.007130   7.941 1.53e-13 ***
## work        -0.002379   0.005807  -0.410    0.682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.8405497)
##
##     Null deviance: 220.62  on 199  degrees of freedom
## Residual deviance: 164.75  on 196  degrees of freedom
## AIC: 538.8
##
## Number of Fisher Scoring iterations: 2
```

```r
simple_glm_math <- glm(num_awards ~ math, family = "gaussian", data = d)
print('simple_glm_math:')
```

```
## [1] "simple_glm_math:"
```

```r
summary(simple_glm_math)
```

```
##
## Call:
## glm(formula = num_awards ~ math, family = "gaussian", data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7113  -0.5940  -0.0968   0.2901   4.4563
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.311023   0.370566  -6.236 2.65e-09 ***
## math         0.055865   0.006931   8.061 7.06e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.8389427)
```

```
##
##     Null deviance: 220.62  on 199  degrees of freedom
## Residual deviance: 166.11  on 198  degrees of freedom
## AIC: 536.44
##
## Number of Fisher Scoring iterations: 2
```

**Add your verbal summary here (1p):**

Our response variable (number of awards) represents count data. For that Poisson regression could be more appropriate.

First, we assess the applicability of Poisson regression by slicing the data at each math-points-group and observing the distribution (histogram) of the count response (num_awards). We conclude, that, indeed, for each level of x (math), the responses (num_awards) follows a Poisson distribution.

We fit the model using `glm(num_awards ~ math, family = "poisson", data = d)` (Poisson regression, using only math predictor).

According to our model, relationship between **math** and **num_awards** won is multiplicative. By increasing a math score by one point, the *num_awards* won multiplies by $e^{0.086}$ where 0.086 is the math coefficient from the Poisson reg. coefficients table.

```
num_awards_math_1 <- predict(poiss_v1_glm, newdata = data.frame(math = 45), type = "response")[[1]]
num_awards_math_2 <- predict(poiss_v1_glm, newdata = data.frame(math = 46), type = "response")[[1]]

num_awards_math_2_test <- num_awards_math_1 * exp(poiss_v1_glm$coefficients[["math"]])

# should be the same (diff 0):
print(paste("diff =", num_awards_math_2 - num_awards_math_2_test))
```

```
## [1] "diff = 0"
```

In order to compare Poisson model with the simple linear model, we need to find some common ground. We fit simple linear model again, but this time using `glm(num_awards ~ math, family = "gaussian", data = d)`. Then we can compare **AIC**. Our Poisson model has a lower (which is better) AIC **384.08** than the simple linear model (AIC 538.8).

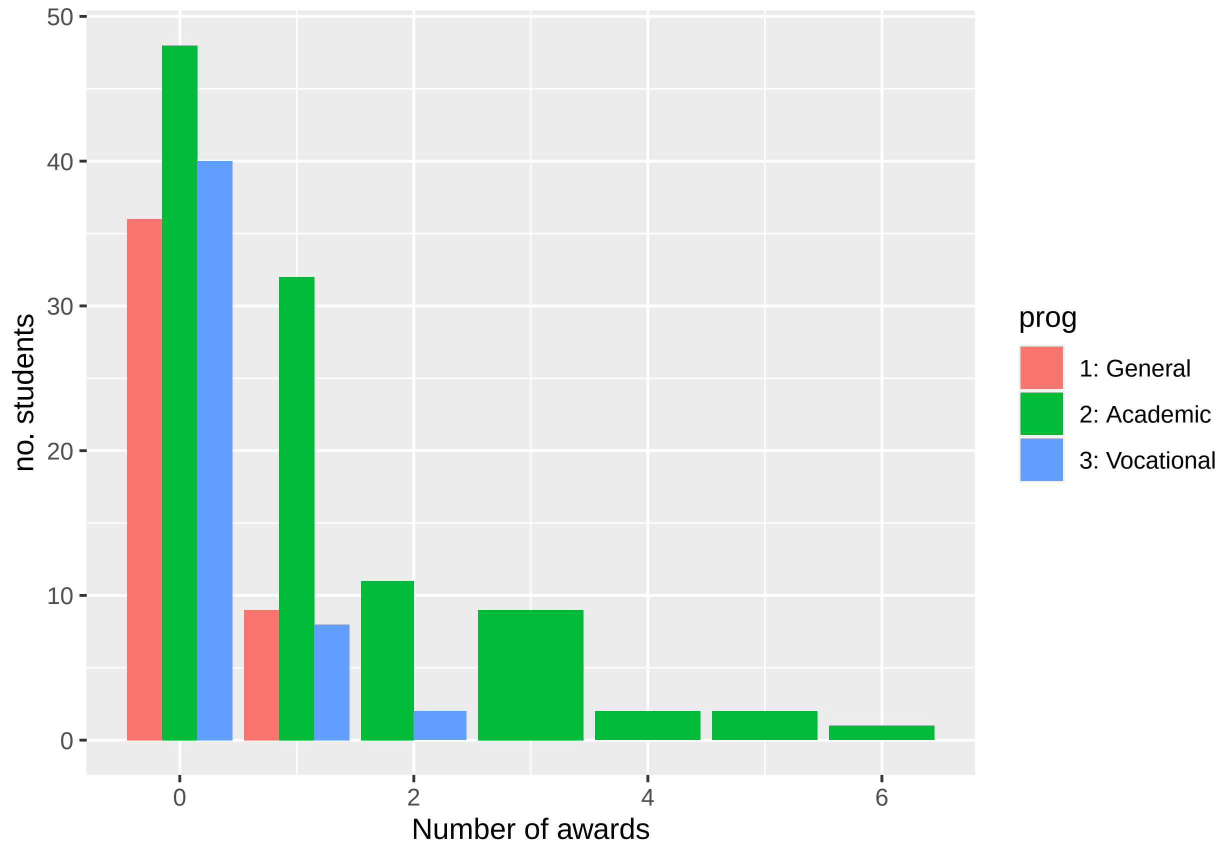
Our Poisson model overcomes the simple linear model because Poisson regression is better suited for count data. Among other things, Poisson regression does not predict negative counts.

## Feature transformations and final model

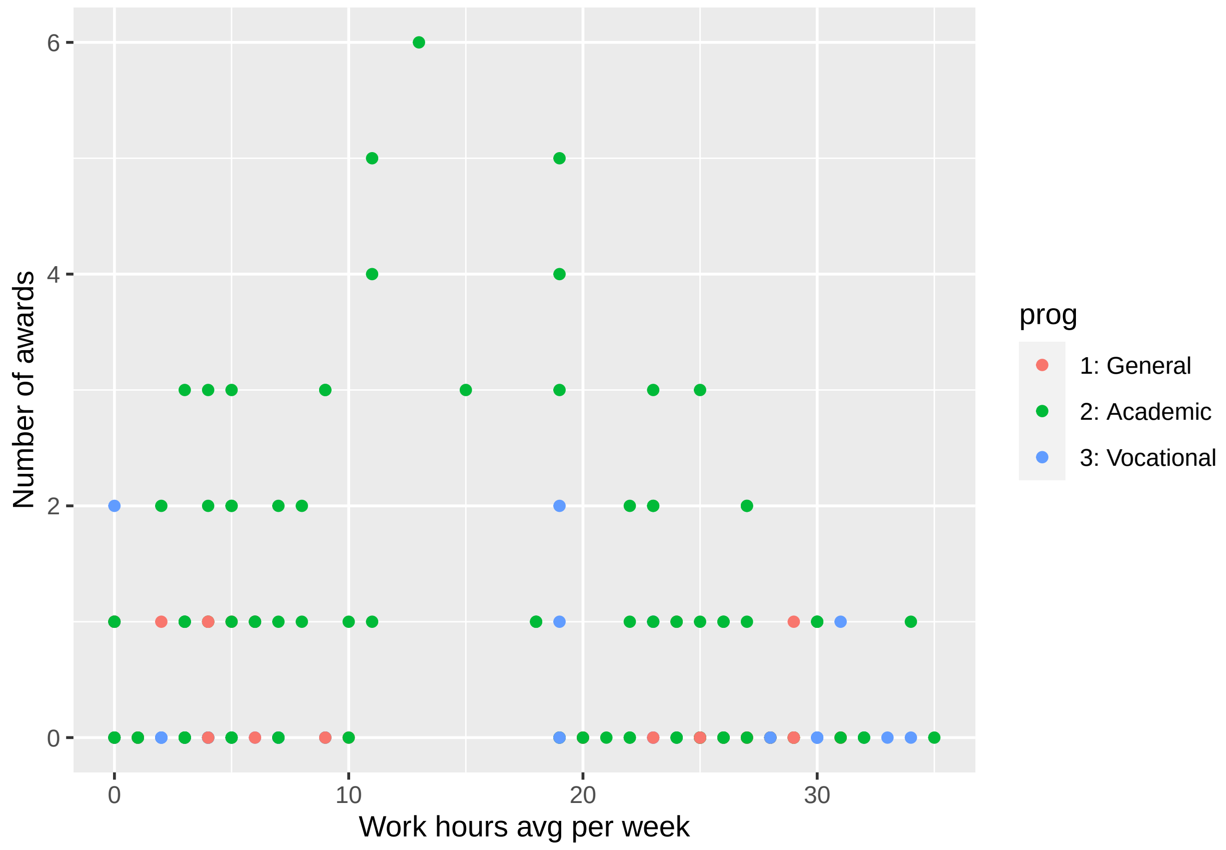
*prog* and *work* did not prove to be effective predictors so far. Visualize these predictors as well as their relationship with the outcome variable. Based on the observations, propose suitable transformations for them (or, justify that they are truly uninformative for prediction of *num_awards*) and implement them into the best model found by now. Use the *compareGLM()* function to validate that your new GLMs indeed improved over the simple one.

```
# Note:
#   We defined d2 with the prog column of type factor in Section "A few preliminary explorations",
#   where we also did a few visualizations some of which we will repeat below.

d2 %>%
    group_by(num_awards, prog) %>%
    summarise(count = n()) %>%
    ggplot(aes(x = num_awards, y = count, fill = prog)) +
    geom_col(position = "dodge") +
    xlab("Number of awards") +
    ylab("no. students")
```

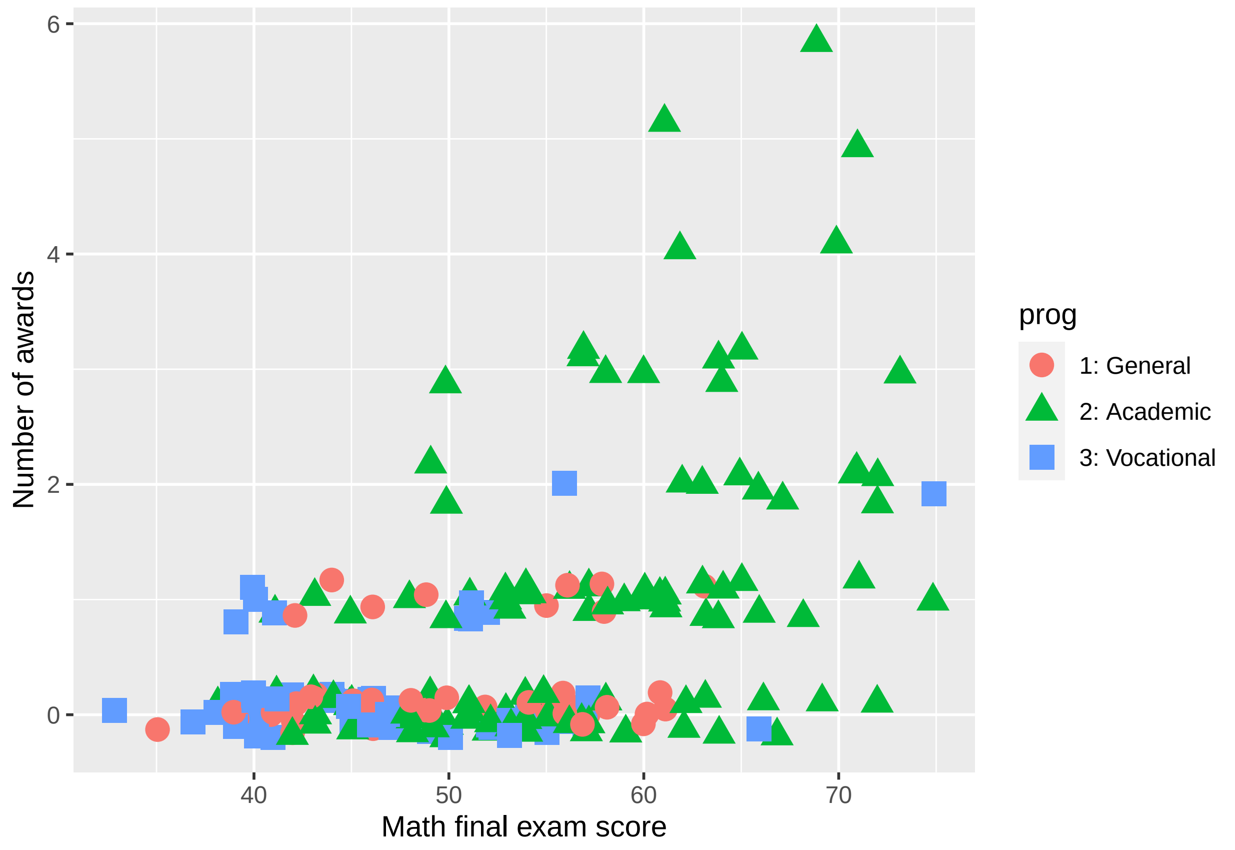
```
## `summarise()` has grouped output by 'num_awards'. You can override using the
## `.groups` argument.
```



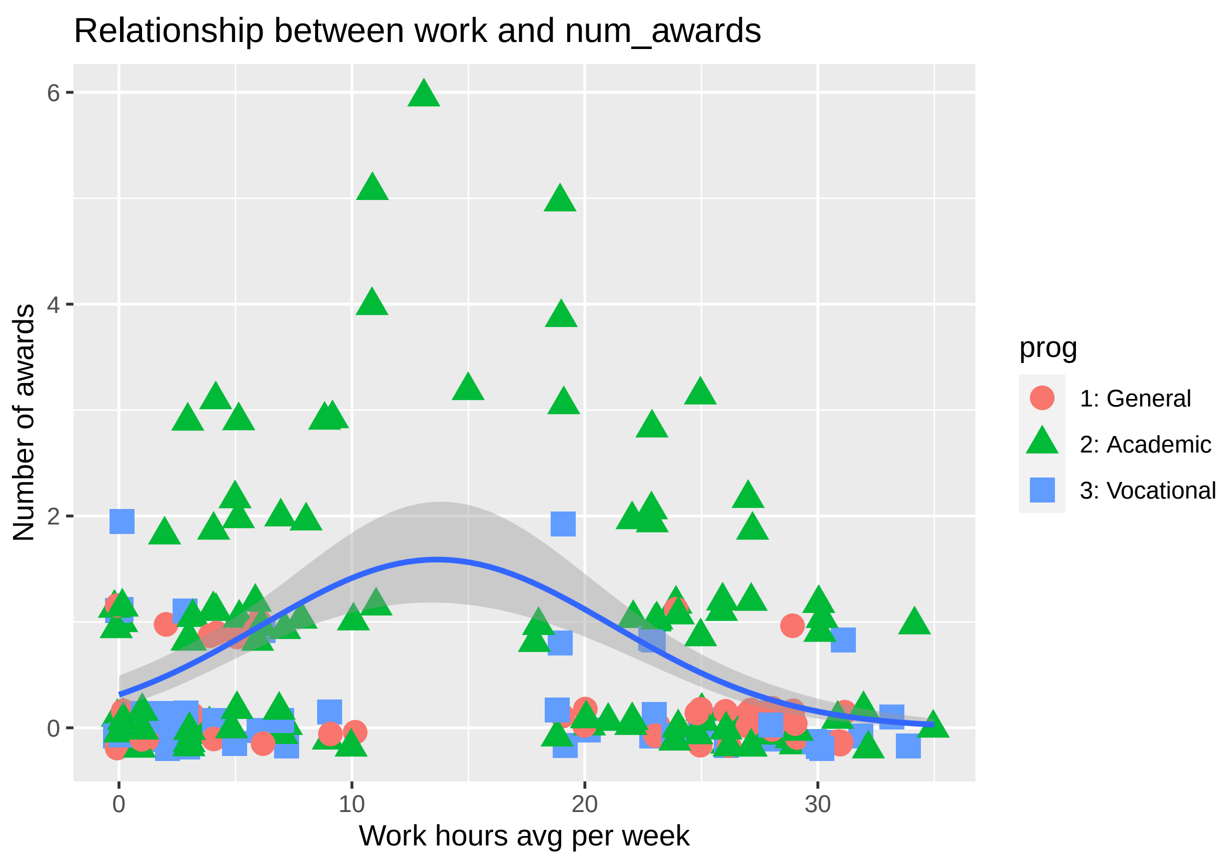
```
d2 %>%
    ggplot(aes(x = work, y = num_awards, color = prog)) +
    geom_point() +
    xlab("Work hours avg per week") +
    ylab("Number of awards")
```



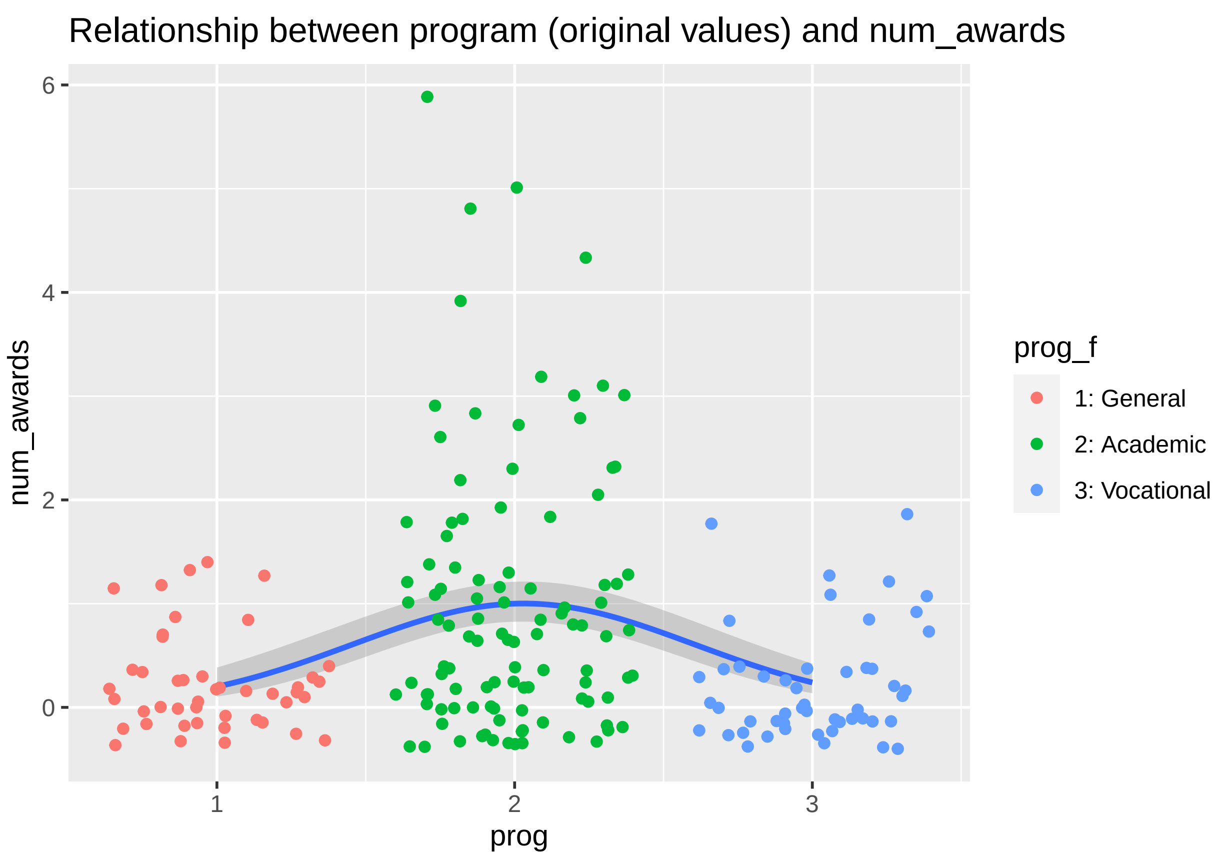
```
# relationship between math score and num_awards (also considering the program)
d2 %>%
    ggplot(aes(x = math, y = num_awards, shape = prog, color = prog)) +
    geom_jitter(size = 4, width = 0.2, height = 0.2) +
    xlab("Math final exam score") +
    ylab("Number of awards")
```



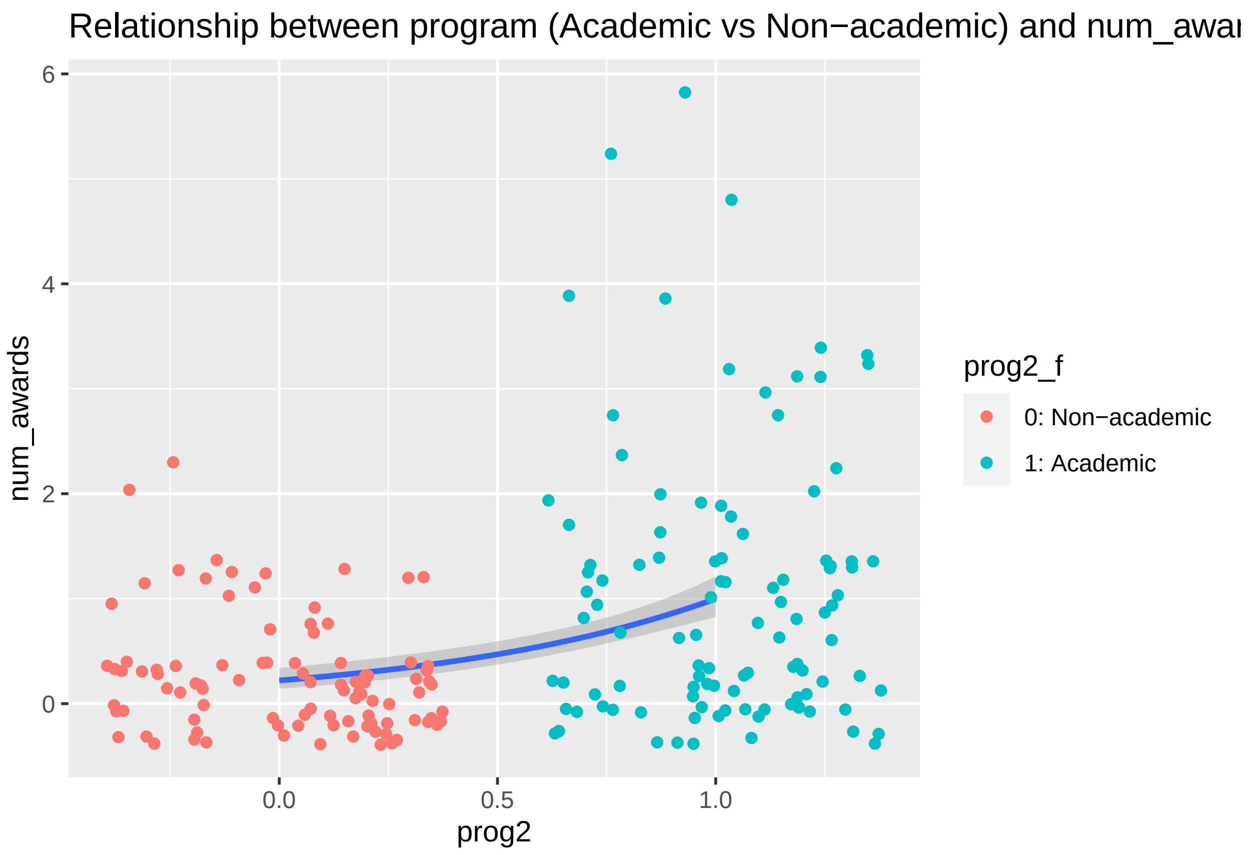
```
# relationship between work (the number of hours that student spent at work on average per week)
# and num_awards (also considering the program)
d2 %>%
    ggplot(aes(x = work, y = num_awards)) +
    geom_jitter(aes(shape = prog, color = prog), size = 4, width = 0.2, height = 0.2) +
    geom_smooth(method = "glm", formula = y ~ poly(x, 2), method.args = list(family = "poisson")) +
    xlab("Work hours avg per week") +
    ylab("Number of awards") +
    ggtitle("Relationship between work and num_awards")
```



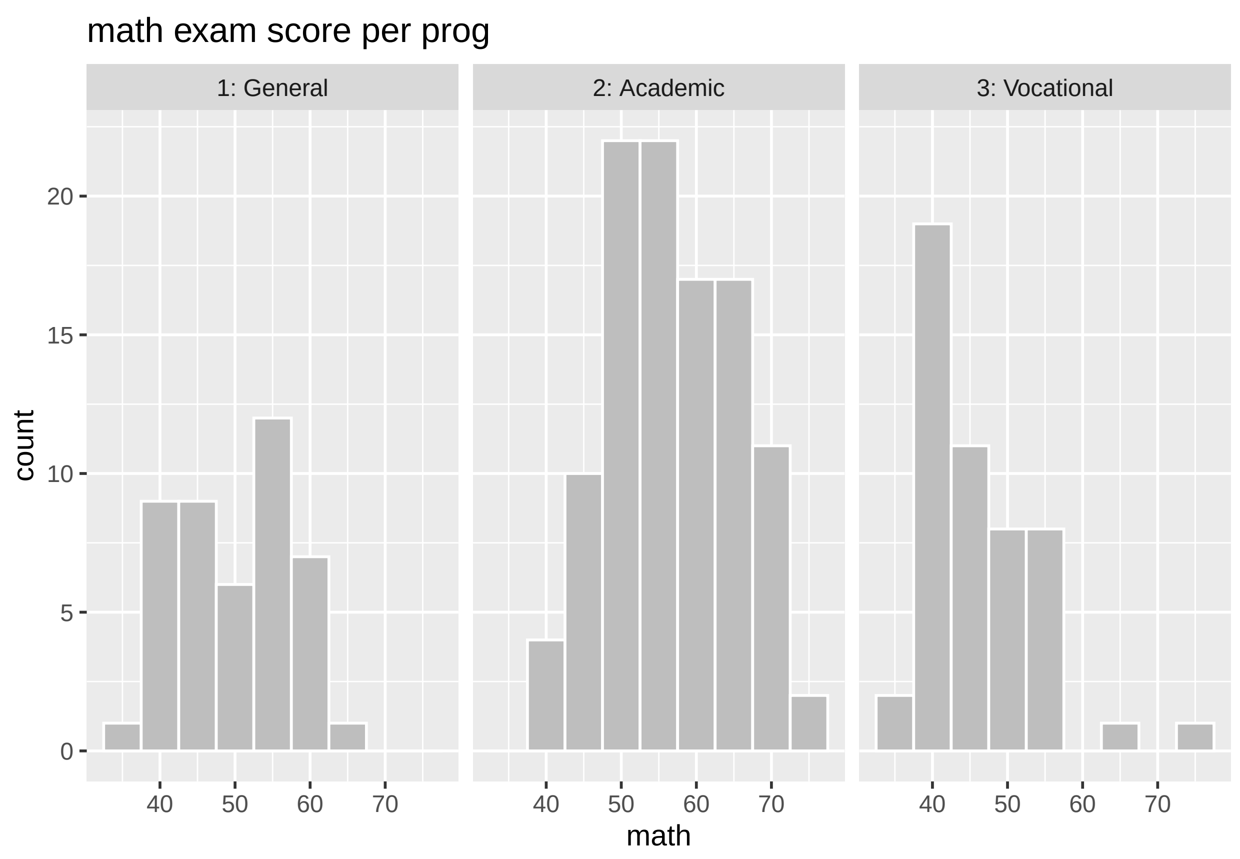
```r
d3 <- d %>% mutate(
    prog2 = (abs(prog - 2L) * -1L) + 1L,
    prog2_f = factor(prog2, labels = c("0: Non-academic", "1: Academic")),
    prog_f = factor(prog, labels = prog_labels)
)

d3 %>%
    ggplot(aes(x = prog, y = num_awards)) +
    geom_smooth(method = "glm", formula = y ~ poly(x, 2), method.args = list(family = "poisson")) +
    geom_jitter(aes(color = prog_f)) +
    ggtitle("Relationship between program (original values) and num_awards")
```



```
d3 %>%
    ggplot(aes(x = prog2, y = num_awards)) +
    geom_smooth(method = "glm", formula = y ~ x, method.args = list(family = "poisson")) +
    geom_jitter(aes(color = prog2_f)) +
    ggtitle("Relationship between program (Academic vs Non-academic) and num_awards")
```



```
d3 %>%
    ggplot(aes(x = math)) +
    geom_histogram(binwidth = 5, center = 0, fill = 'gray', color = 'white') +
    facet_grid(cols = vars(prog_f)) +
    ggtitle("math exam score per prog")
```



```
# correlation matrix might be useful
d3.cor <- d3 %>% select(-prog_f, -prog2_f) %>% cor()

poiss_v2_glm <- glm(num_awards ~ math + poly(prog, 2) + poly(work, 2), family = "poisson", data = d)
# print("poiss_v2_glm:")
# summary(poiss_v2_glm)

poiss_v3_glm <- glm(num_awards ~ math + prog2 + poly(work, 2), family = "poisson", data = d3)
# print("poiss_v3_glm:")
# summary(poiss_v3_glm)

poiss_v4_glm <- glm(num_awards ~ math + prog2, family = "poisson", data = d3)
# print("poiss_v4_glm:")
# summary(poiss_v4_glm)

poiss_v5_glm <- glm(num_awards ~ math + poly(work, 2), family = "poisson", data = d3)
# print("poiss_v5_glm:")
# summary(poiss_v5_glm)

# predict
# poiss_v2_glm_preds <- predict(poiss_v2_glm, d, type = "response")
# poiss_v2_glm_preds_r <- round(poiss_v2_glm_preds)

# compare with the simpler GLM
print("Compare poiss_v1_glm and poiss_v2_glm and poiss_v3_glm and poiss_v4_glm and poiss_v5_glm:")
```

```
## [1] "Compare poiss_v1_glm and poiss_v2_glm and poiss_v3_glm and poiss_v4_glm and poiss_v5_glm:"
```

```
compareGLM(poiss_v1_glm, poiss_v2_glm, poiss_v3_glm, poiss_v4_glm, poiss_v5_glm)
```

```
## $Models
##   Formula
## 1 "num_awards ~ math"
## 2 "num_awards ~ math + poly(prog, 2) + poly(work, 2)"
## 3 "num_awards ~ math + prog2 + poly(work, 2)"
## 4 "num_awards ~ math + prog2"
## 5 "num_awards ~ math + poly(work, 2)"
##
## $Fit.criteria
##   Rank Df.res   AIC  AICc   BIC McFadden Cox.and.Snell Nagelkerke   p.value
## 1    2    198 386.1 386.2 396.0   0.1804        0.3418     0.3791 2.986e-20
## 2    6    194 364.3 364.9 387.4   0.2445        0.4327     0.4800 3.837e-23
## 3    5    195 362.8 363.3 382.6   0.2434        0.4313     0.4784 8.701e-24
## 4    3    197 374.2 374.4 387.4   0.2103        0.3859     0.4280 3.347e-22
## 5    4    196 372.5 372.8 389.0   0.2184        0.3973     0.4407 4.109e-22
```

**Add your verbal summary here (2p):**

First, we visualize various relationship among the response variable and predictors.

There seems to be a relationship between `prog` and `num_awards` in our data. Students of *Academic* program seem to receive higher number of awards. There does not seem to be much difference in `num_awards` for students of the other two programs (*General* and *Vocational*). That's our motivation for introducing a *binary* feature `prog2` that has value `1` when prog is Academic program and value `0` otherwise. One thing we can notice from the features' correlation matrix is that `prog2` is highly correlated with `math` score.

Next, there seems to be a quadratic relationship between `work` and `num_awards` in our data. It seems that if students work a little (but not too much), they tend to receive higher number of awards. This is consistent with the notion that successful students manage to balance time between school and work.

Based on these observations, we fit 4 new more complex GLMs using different combinations of `math` predictor (always included) and transformed `work`, `prog` and `prog2` predictors.

Finally, we compare all 5 different GLMs. We see that our **v3** model (`num_awards ~ math + prog2 + poly(work, 2)`) is the best in terms of AIC (362.8) and BIC (382.6).

## Ablation study through cross-validation

Recap all the models considered previously and evaluate them through cross-validation. You can start with the most simple null model and gradually add the previously discussed improvements. See their role in terms of MAE, RMSE and other commonly used criteria. The procedure outlined below proposes to work with *train* function from *caret* package, you can only add more models to evaluate and compare.

```
train_control <- trainControl(method = "cv", number = 10)

results <- list()

results[["lm model null"]] <- train(
    x = data.frame(rep(1, nrow(d))),
    y = d$num_awards,
    method = "lm",
    trControl = train_control
)$results
```

```r
results[["lm model all"]] <- train(
    x = d %>% select(prog, math, work),
    y = d$num_awards,
    method = "lm",
    trControl = train_control
)$results

results[["lm model all"]] <- train(
    x = d %>% select(math),
    y = d$num_awards,
    method = "lm",
    trControl = train_control
)$results

results[["glm poisson model v1: math-only"]] <- train(
    x = d %>% select(math),
    y = d$num_awards,
    trControl = train_control,
    method = "glm",
    family = "poisson",
)$results

results[["glm poisson model v2: math + poly(prog, 2) + poly(work, 2)"]] <- train(
    x = d3 %>% select(math, prog, work) %>% mutate(prog = poly(prog, 2), work = poly(work, 2)),
    y = d3$num_awards,
    trControl = train_control,
    method = "glm",
    family = "poisson",
)$results

results[["glm poisson model v3: math + prog2 + poly(work, 2)"]] <- train(
    x = d3 %>% select(math, prog2, work) %>% mutate(work = poly(work, 2)),
    y = d3$num_awards,
    trControl = train_control,
    method = "glm",
    family = "poisson",
)$results

results[["glm poisson model v4: math + prog2"]] <- train(
    x = d3 %>% select(math, prog2),
    y = d3$num_awards,
    trControl = train_control,
    method = "glm",
    family = "poisson",
)$results

results[["glm poisson model v5: math + poly(work, 2)"]] <- train(
    x = d3 %>% select(math, work) %>% mutate(work = poly(work, 2)),
    y = d3$num_awards,
    trControl = train_control,
    method = "glm",
    family = "poisson",
)$results
```

```
# https://stackoverflow.com/questions/16138693/rbind-multiple-data-sets
all_results <- bind_rows(results, .id = "name")
# rownames(all_results) <- names(results)

all_results
```

```
##                                                          name intercept
## 1                                               lm model null      TRUE
## 2                                                lm model all      TRUE
## 3                              glm poisson model v1: math-only        NA
## 4 glm poisson model v2: math + poly(prog, 2) + poly(work, 2)        NA
## 5         glm poisson model v3: math + prog2 + poly(work, 2)        NA
## 6                          glm poisson model v4: math + prog2        NA
## 7                  glm poisson model v5: math + poly(work, 2)        NA
##        RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD parameter
## 1 1.0206353       NaN 0.7830292 0.2629535         NA 0.11332701      <NA>
## 2 0.8882432 0.2966416 0.6297608 0.2391531  0.1500591 0.12718623      <NA>
## 3 0.8757433 0.3179296 0.6209119 0.2553278  0.2356503 0.15509174      none
## 4 0.8162399 0.4367746 0.5872537 0.1986150  0.2120631 0.14042249      none
## 5 0.8118800 0.4047454 0.5824629 0.1830539  0.2075201 0.10767588      none
## 6 0.8641619 0.3303125 0.5983143 0.2098186  0.1384228 0.10970012      none
## 7 0.8620362 0.4189169 0.6102196 0.1224988  0.2041556 0.05594767      none
```

**Add your verbal summary here (1p):**

We evaluated all previously discussed models using cross-validation. Our `glm poisson model v3: math + prog2 + poly(work, 2)` that we constructed in the previous part performed best during repeated evaluation (random CV folds). It has lowest both RMSE and MAE.