Homework 2

Jonah, Declan, Madelyn, Peter

Homework 2

Due Friday, February 18 at 9:00am CST on Moodle

Deliverables: Please use this template to knit an HTML document. Convert this HTML document to a PDF by opening the HTML document in your web browser. *Print* the document (Ctrl/Cmd-P) and change the destination to "Save as PDF". Submit this one PDF to Moodle.

Alternatively, you may knit your Rmd directly to PDF if you have LaTeX installed.

Project Work

Instructions

Goal: Begin an analysis of your dataset to answer your **regression** research question.

Collaboration: Form a team (2-3 members) for the project and this part can be done as a team. Only one team member should submit a Project Work section. Make sure you include the full names of all of the members in your write up.

Data cleaning: If your dataset requires any cleaning (e.g., merging datasets, creation of new variables), first consult the R Resources page to see if your questions are answered there. If not, post on the #rcode-questions channel in our Slack workspace to ask for help. *Please ask for help early and regularly* to avoid stressful workloads.

Required Analyses

1. **Initial investigation: ignoring nonlinearity (for now)**
    a. Use ordinary least squares (OLS) by using the `lm` engine and LASSO (`glmnet` engine) to build a series of initial regression models for your quantitative outcome as a function of the predictors of interest. (As part of data cleaning, exclude any variables that you don't want to consider as predictors.)
        - You'll need two model specifications, `lm_spec` and `lm_lasso_spec` (you'll need to tune this one).
    b. For each set of variables, you'll need a `recipe` with the `formula`, `data`, and pre-processing steps
        - You may want to have steps in your recipe that remove variables with near zero variance (`step_nzv()`), remove variables that are highly correlated with other variables (`step_corr()`), normalize all quantitative predictors (`step_normalize(all_numeric_predictors())`) and add indicator variables for any categorical variables (`step_dummy(all_nominal_predictors())`).
        - These models should not include any transformations to deal with nonlinearity. You'll explore this in the next investigation.

c. Estimate the test performance of the models using CV. Report and interpret (with units) the CV metric estimates along with a measure of uncertainty in the estimate (`std_error` is readily available when you used `collect_metrics(summarize=TRUE)`).
   - Compare estimated test performance across the models. Which models(s) might you prefer?

d. Use residual plots to evaluate whether some quantitative predictors might be better modeled with nonlinear relationships.

e. Which variables do you think are the most important predictors of your quantitative outcome? Justify your answer. Do the methods you've applied reach consensus on which variables are most important? What insights are expected? Surprising?
   - Note that if some (but not all) of the indicator terms for a categorical predictor are selected in the final models, the whole predictor should be treated as selected.

Your Work a & b.

```
# library statements
library(readr)
library(ggplot2)
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
```

```
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip

## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --

## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.0      v tibble       3.1.6
## v infer        1.0.0      v tidyr        1.2.0
## v modeldata    0.1.1      v tune         0.1.6
## v parsnip      0.1.7      v workflows    0.2.4
## v purrr        0.3.4      v workflowsets 0.1.0
## v recipes      0.1.17     v yardstick    0.0.9

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard()  masks scales::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
library(rvest)
```

```
##
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
tidymodels_prefer()
library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
```

```
library(glmnet)
```

```
## Loaded glmnet 4.1-3
```

```
season_data <- read_csv('understat.com.csv')
```

```
## New names:
## * `` -> ...1
## * `` -> ...2

## Rows: 684 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): ...1, team
## dbl (22): ...2, position, matches, wins, draws, loses, scored, missed, pts, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# data cleaning
colnames(season_data)[9] = "goals"
colnames(season_data)[10] = "conceded"

drop = c("X", "X.1", "position", "team", "matches", "draws", "loses", "xpts", "xpts_diff", "wins", "...
season_data_clean = season_data[!colnames(season_data) %in% drop]
```

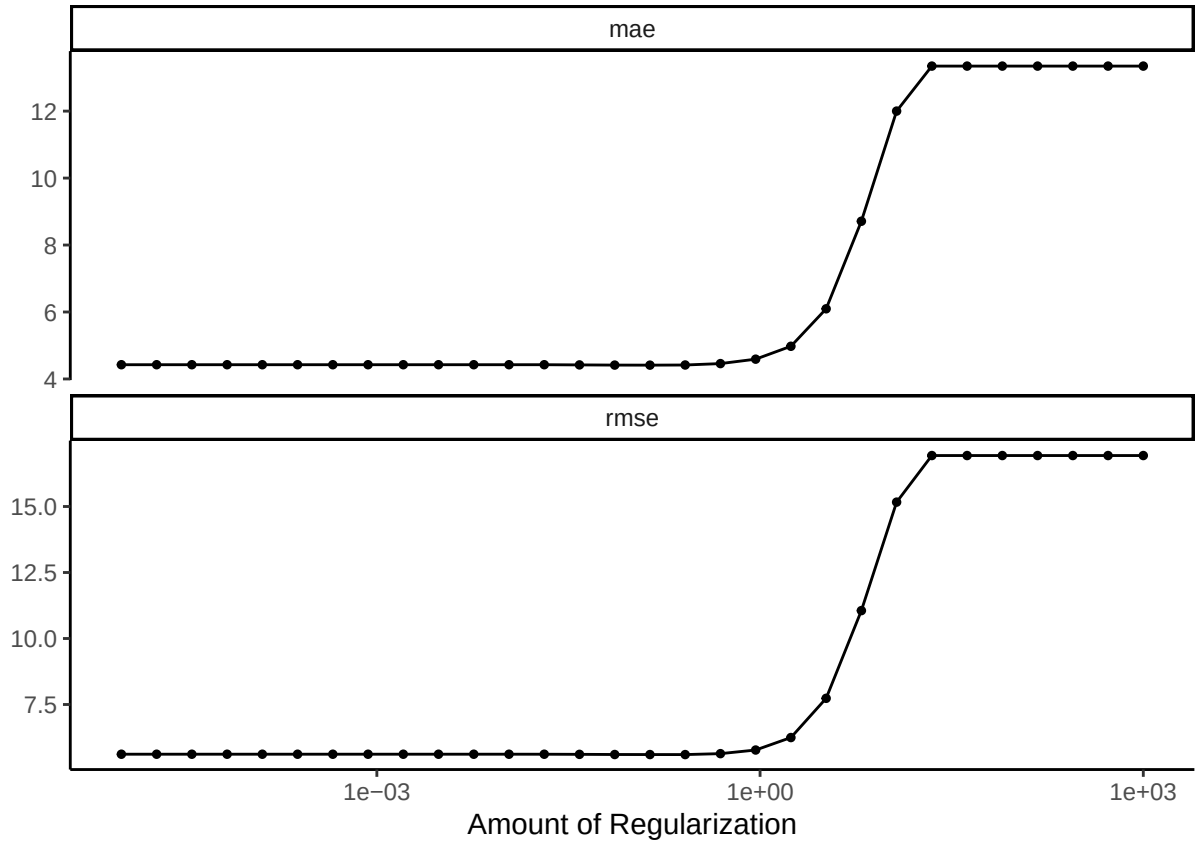
```
# creation of cv folds
set.seed(123)
season_CV9 <- vfold_cv(season_data_clean, v = 9)
```

```r
# OLS model
lm_spec <-
    linear_reg() %>% # this is the type of model we are fitting
    set_engine(engine = 'lm') %>% # you'll learn other engines to fit the model
    set_mode('regression')

#lasso model

lm_lasso_spec <-
  linear_reg()%>%
  set_args(mixture = 1, penalty = tune())%>%
  set_engine(engine = 'glmnet')%>%
  set_mode('regression')
```

```r
# recipes & workflows OLS
season_rec <- recipe(pts ~ ., data = season_data_clean) %>%
  step_lincomb(all_numeric_predictors())%>% #remove predictors that are combinations of each other
  step_nzv(all_numeric_predictors())#remove predictors with near zero variablility

season_model_wf <- workflow() %>%
  add_recipe(season_rec)%>%
  add_model(lm_spec)

#recipe and workflow for LASSO
season_rec_LASSO <-
  recipe(pts ~ ., data = season_data_clean) %>%
  step_lincomb(all_numeric_predictors())%>% #remove predictors that are combinations of each other
  step_nzv(all_numeric_predictors())%>%#remove predictors with near zero variablility
  step_normalize(all_numeric_predictors())%>%#normalize all predictors
  step_corr(all_numeric_predictors())


# Lasso Model Spec with tune
lm_lasso_spec_tune <-
  linear_reg() %>%
  set_args(mixture = 1, penalty = tune()) %>% ## mixture = 1 indicates Lasso
  set_engine(engine = 'glmnet') %>% #note we are using a different engine
  set_mode('regression')
```

```r
# fit & tune models

# Workflow (Recipe + Model)
lasso_wf_tune <- workflow() %>%
  add_recipe(season_rec_LASSO) %>%
  add_model(lm_lasso_spec_tune)

# Tune Model (trying a variety of values of Lambda penalty)
penalty_grid <- grid_regular(
  penalty(range = c(-5, 3)), #log10 transformed
  levels = 30)

tune_output <- tune_grid( # new function for tuning hyperparameters
  lasso_wf_tune, # workflow
```

```
  resamples = season_CV9, # cv folds
  metrics = metric_set(rmse, mae),
  grid = penalty_grid # penalty grid defined above
)

autoplot(tune_output)+theme_classic()
```



c.

```
#  calculate/collect CV metrics
Season_CV_metrics <-fit_resamples(season_model_wf,
    resamples = season_CV9,
    metrics = metric_set(rmse, rsq, mae)
    ) %>%
    collect_metrics(summarize = TRUE) #CV Metrics (averages over the 10 folds)

Season_CV_metrics
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   4.33      9 0.115   Preprocessor1_Model1
## 2 rmse    standard   5.50      9 0.137   Preprocessor1_Model1
## 3 rsq     standard   0.896     9 0.00527 Preprocessor1_Model1
```

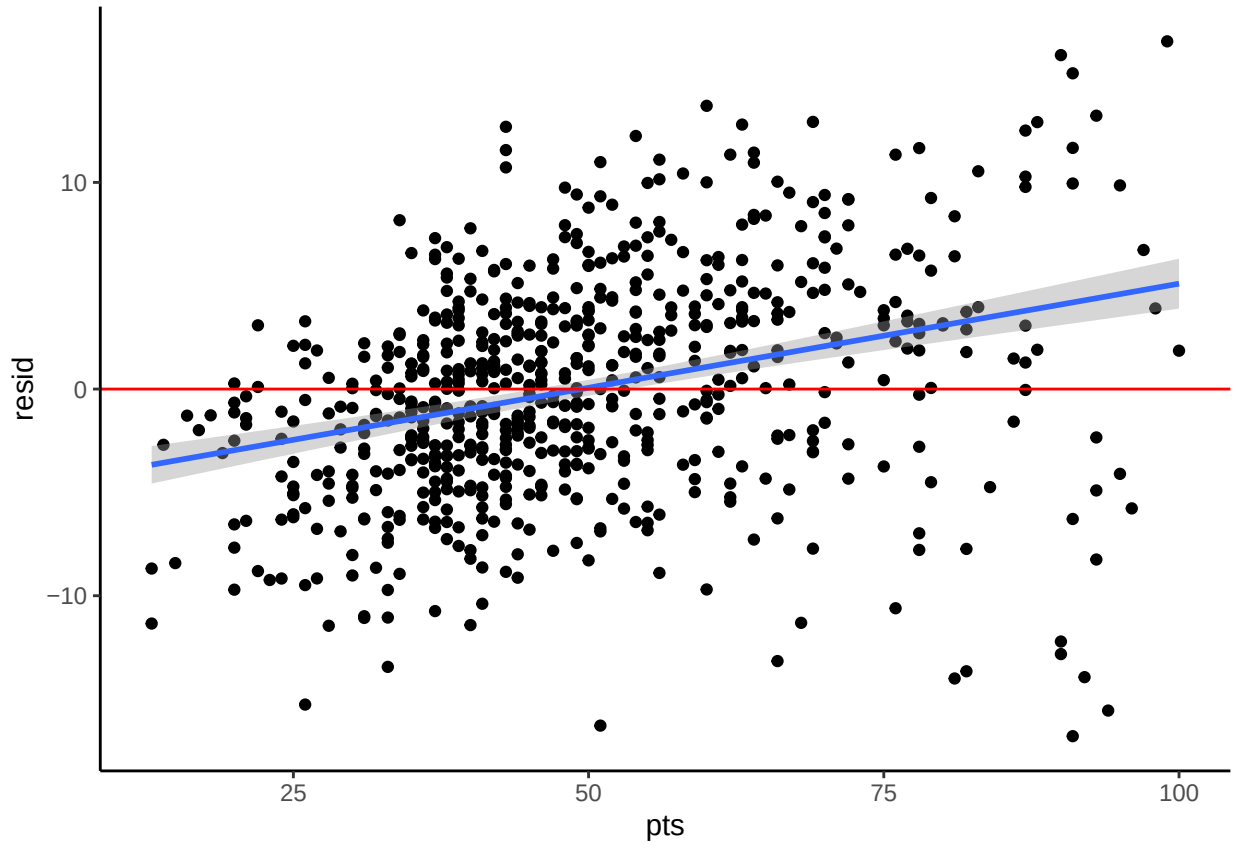
```r
#LASSO metrics
best_se_penalty <- select_by_one_std_err(tune_output, metric = 'mae', desc(penalty))

final_wf<- finalize_workflow(lasso_wf_tune, best_se_penalty)

final_fit <- fit(final_wf, data = season_data_clean)

tidy(final_fit)
```

```
## # A tibble: 9 x 3
##   term         estimate penalty
##   <chr>           <dbl>   <dbl>
## 1 (Intercept)   49.4      0.489
## 2 xG            11.0      0.489
## 3 xG_diff       -3.96     0.489
## 4 xGA_diff       3.48     0.489
## 5 npxGA         -4.28     0.489
## 6 ppda_coef     -0.0891   0.489
## 7 oppda_coef     0        0.489
## 8 deep           0        0.489
## 9 deep_allowed   0        0.489
```

d.

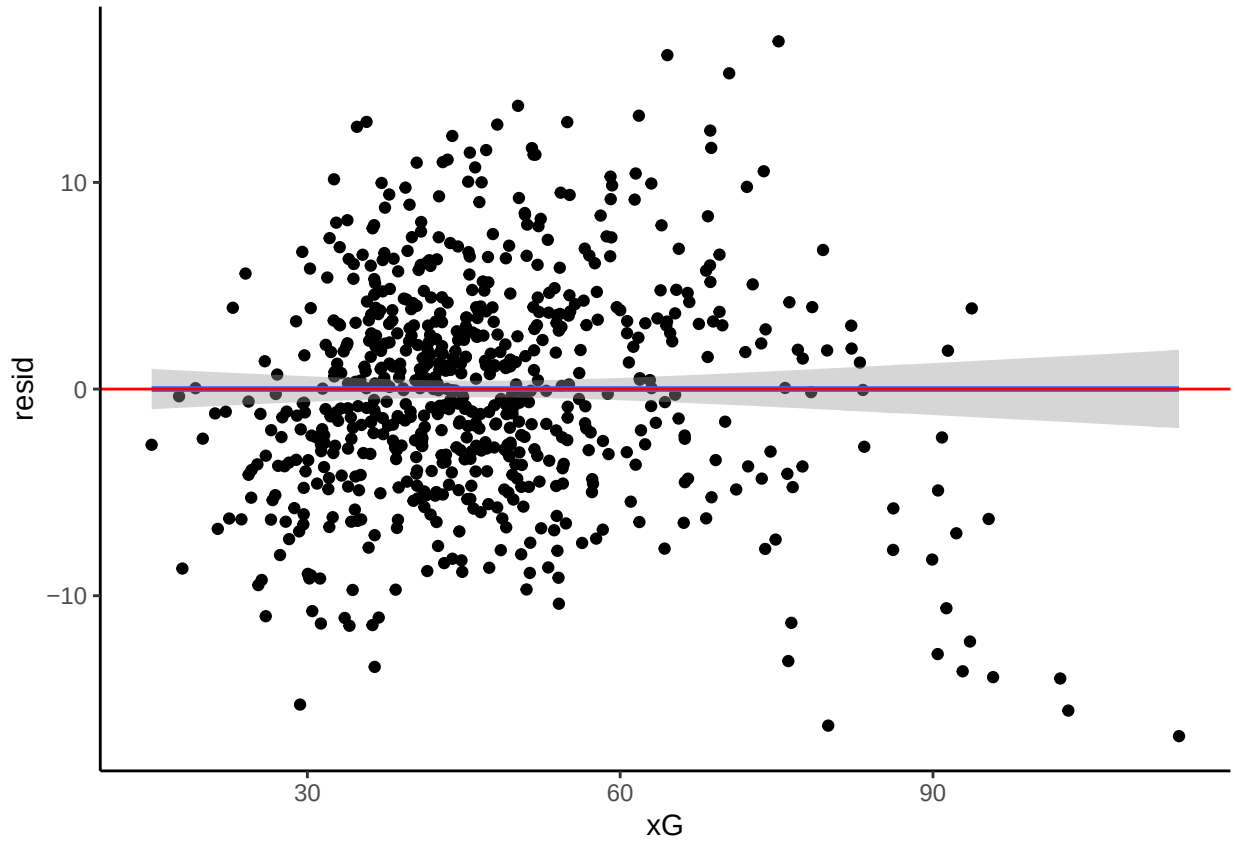
```r
# visual residuals

#OLS Residuals
season_CV_mod <- fit(season_model_wf, data = season_data_clean)

season_CV_modoutput <- season_data_clean%>%
  bind_cols(predict(season_CV_mod, new_data = season_data_clean))%>%
  mutate(resid = pts - .pred)

ggplot(season_CV_modoutput, aes(x = pts, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = "red")+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



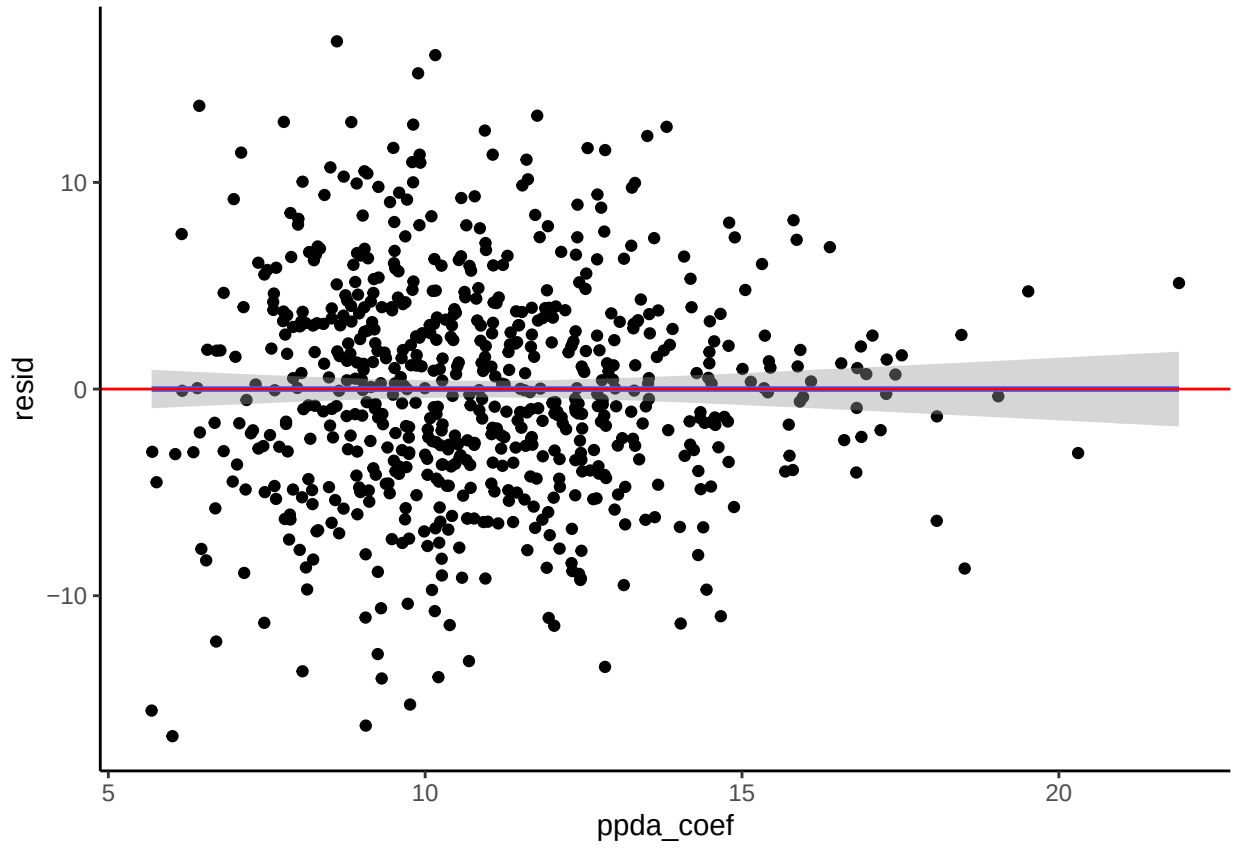

```
ggplot(season_CV_modoutput, aes(x = xG, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = 'red')+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



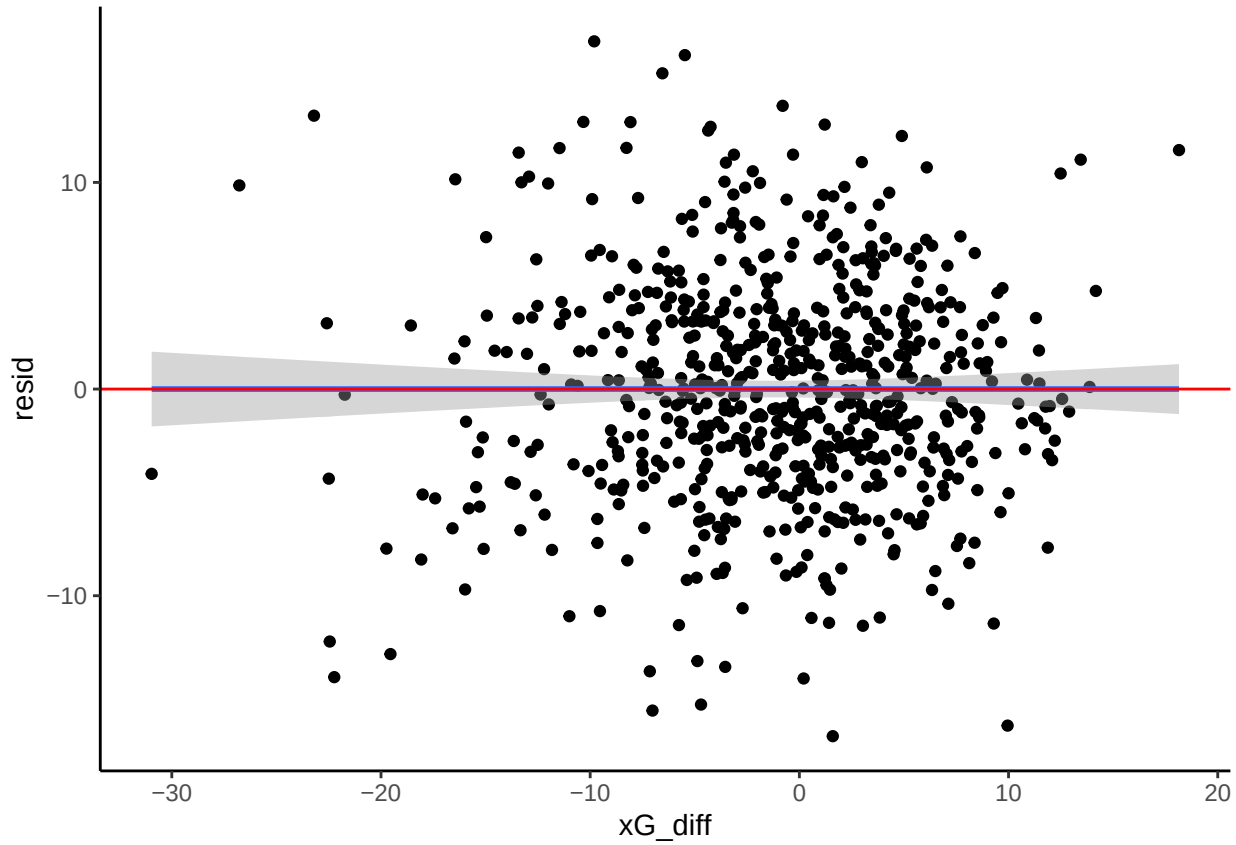

```
ggplot(season_CV_modoutput, aes(x = ppda_coef, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = 'red')+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(season_CV_modoutput, aes(x = xG_diff, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = 'red')+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```

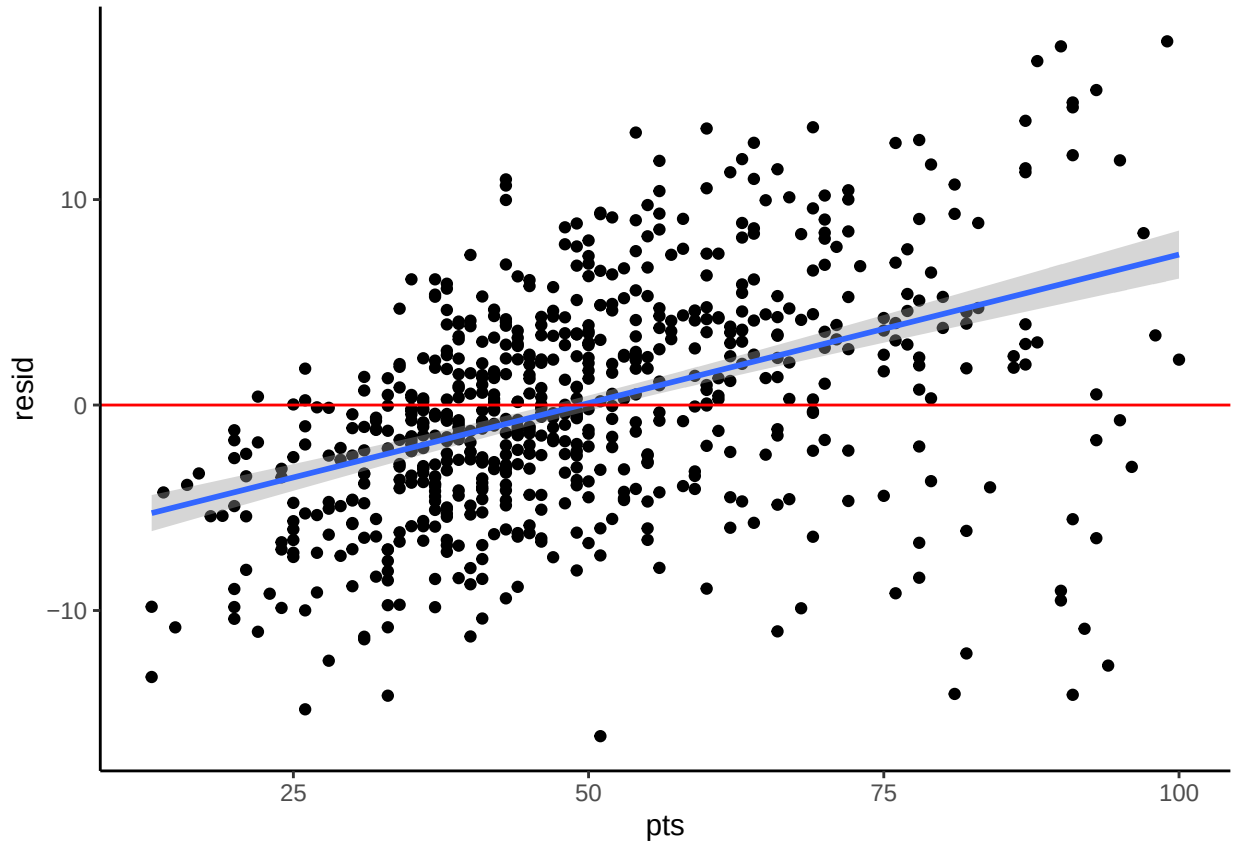



```
#Lasso Residuals

season_LASSO_MOD_output <- season_data_clean%>%
  bind_cols(predict(final_fit, new_data = season_data_clean))%>%
  mutate(resid = pts - .pred)

season_LASSO_MOD_output %>%
  rsq(truth = pts, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.896
```

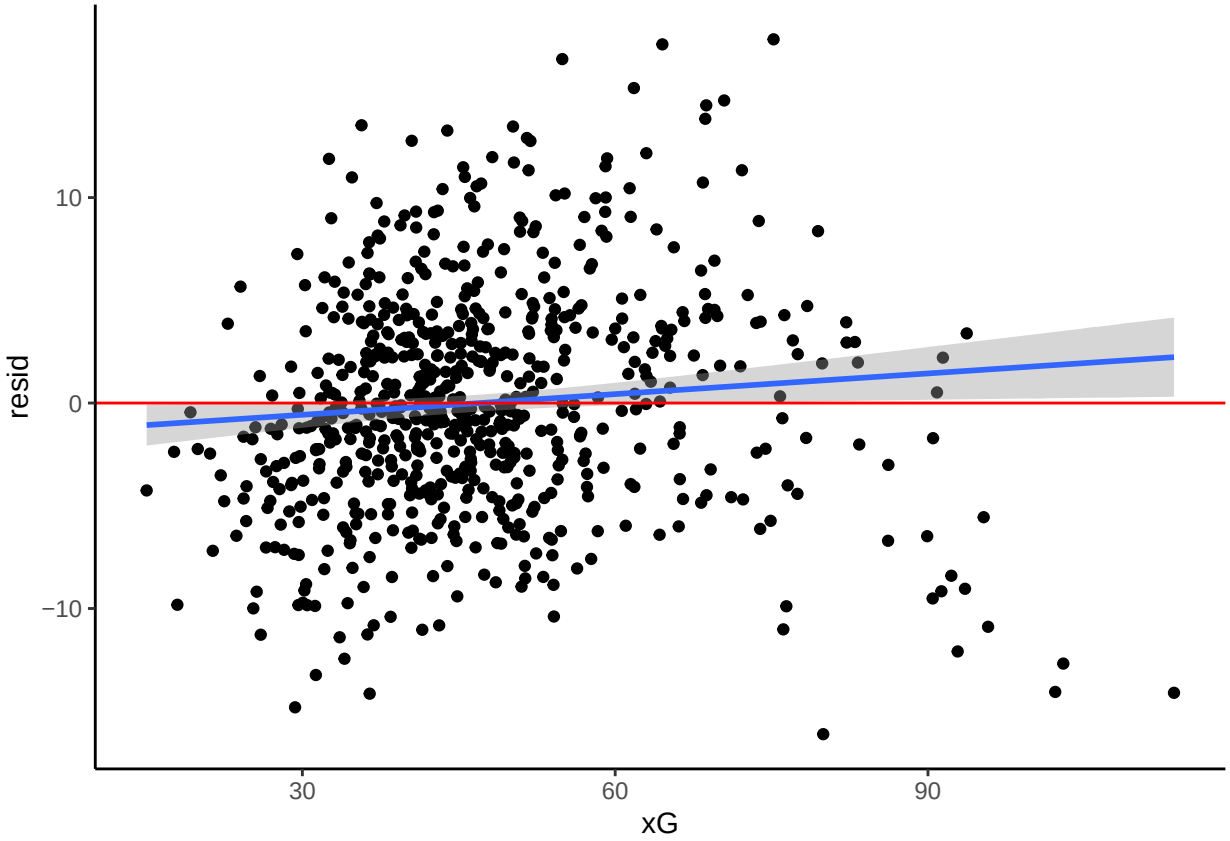
```
ggplot(season_LASSO_MOD_output, aes(x = pts, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = "red")+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



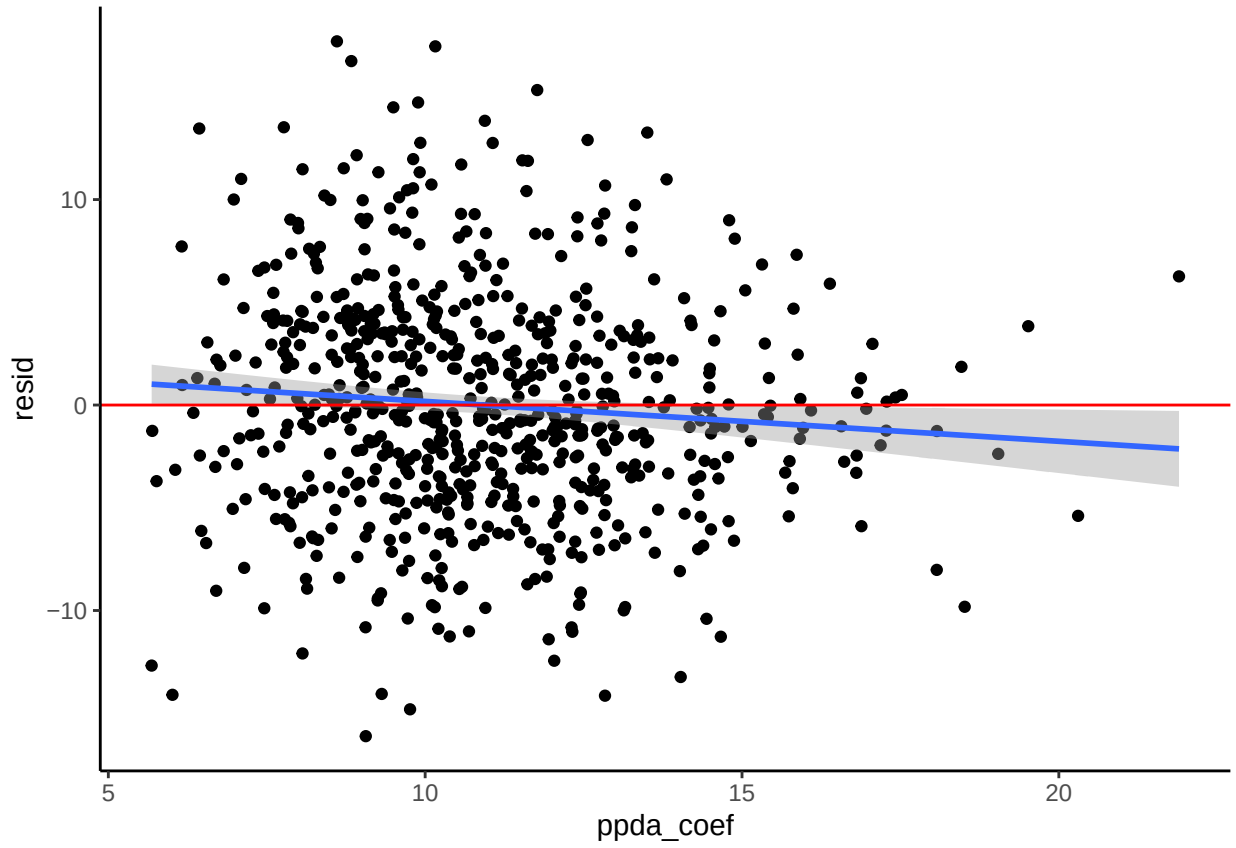

```
ggplot(season_LASSO_MOD_output, aes(x = xG, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = 'red')+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



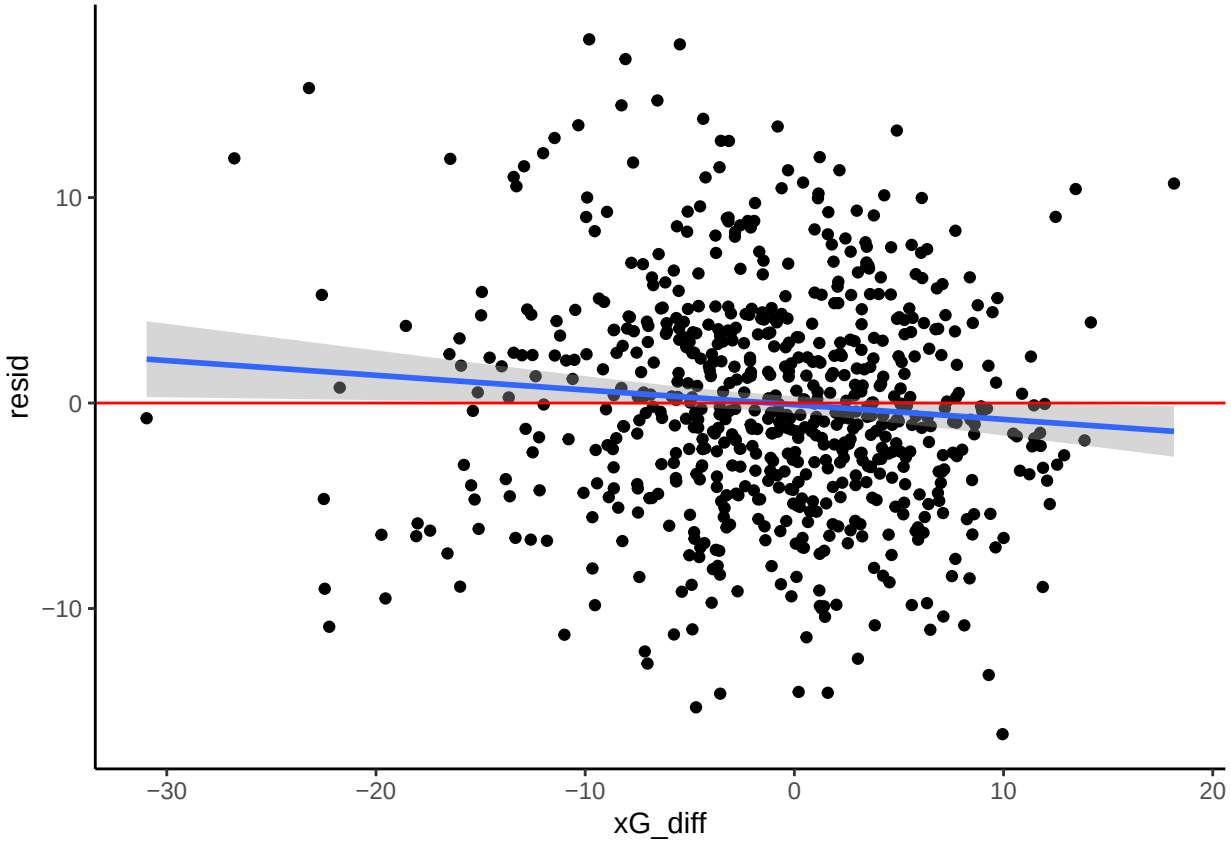

```
ggplot(season_LASSO_MOD_output, aes(x = ppda_coef, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = 'red')+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(season_LASSO_MOD_output, aes(x = xG_diff, y = resid))+
  geom_point()+
  geom_smooth(method = 'lm')+
  geom_hline(yintercept = 0, color = 'red')+
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



e.

2. **Summarize investigations**

    - Decide on an overall best model based on your investigations so far. To do this, make clear your analysis goals. Predictive accuracy? Interpretability? A combination of both? We would use the lasso model with a penalty of 0.4893901. This model included the variables of xG, xGA, xG_diff, npXGA and ppda_coef. This makes sense because teams that score more goals and create better chances to score are more likely to be better. The r-squared was 0.896, which was marginally higher than the OLS method where all predictors were used. We found that OLS was slightly more accurate in predicting pts, with a MAE of 4.3 as opposed to the LASSO MAE of 4.5. LASSO is more interpretable because it uses fewer variables and eliminates the variables that contribute little explanation in the model. LASSO is also much less likely to be overfit because it uses a more general strategy for model creation.

3. **Societal impact**

    - Are there any harms that may come from your analyses and/or how the data were collected?
    - What cautions do you want to keep in mind when communicating your work?

One major harm that we identified in our analysis is the fact that the majority of our indicators are offensive statistics. Many teams play a defensive style, with the goal of not conceding goals. These teams score less goals, but also give up less goals. Because of the lack of offensive production, our model may underpredict the amount of points the defensive teams score. We also need to be careful and realize we are doing analysis on the top 5 European leagues. There are many different styles of play and many different leagues around the world. So what we see in these leagues may not occur in the many other leagues. When communicating

our work we need to keep in mind the fact that many people do not know much about soccer, especially European soccer. We need to be able to find away that everyone will understand what we are trying to present.