LAB 2: Multiple regression, Non-linear regression, overfitting, and cross-validation

Due date : October 15, 2023, 11:59 PM

Your last name: Suri

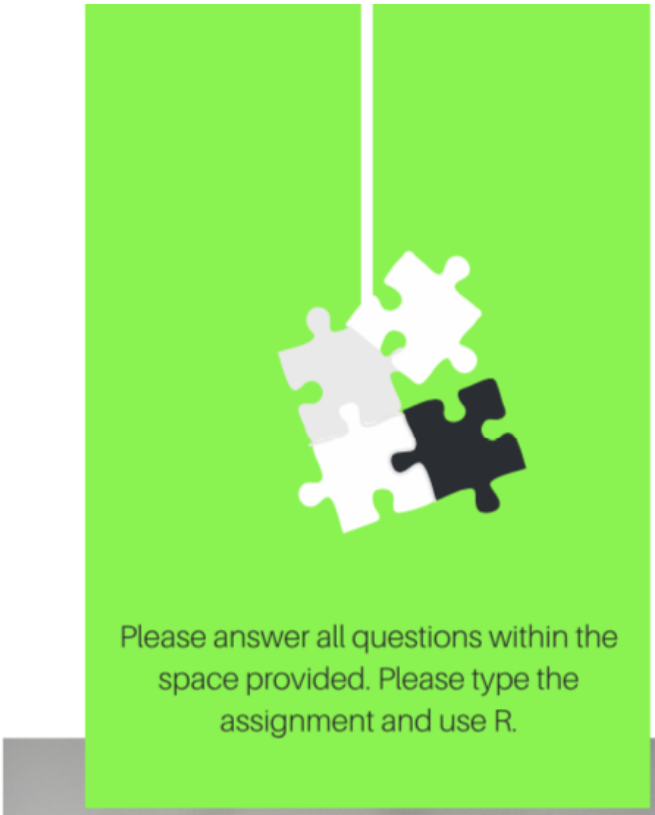
Your first name: Jatin

Your student ID: 261152263

Your last name: Agarwal

Your first name: Lakshya

Your student ID: 261149449



Lab 2: What makes a board game successful

Board games have become exponentially more popular (and better) over the past years. Beer and board games share a similar story: a few decades ago, we only had a few dozen popular board games (e.g., Monopoly, The Game of Life, Scrabble, Clue); similarly, we only had a few dozen popular beer brands (e.g., Budweiser, Heineken, Coors, Molson). But with the rise of the hipster culture, lots of things have changed—some for better and some for worse. To its credit, hipster culture has generated a renewed interest in "craft" and "artisan" objects. For instance, artisan beer has become extremely popular — and, as a result, most cities are now able to enjoy tons of small microbreweries that serve good and tasty beer. The same happened with board games: we are now able to enjoy thousands of alternative board games, some of which are super fun, creative, and unique.[1]

But not all board games are created equally: some are dreadfully lame. In this lab, we will study board game attributes to see what drives board game ratings. To this end, I have downloaded game attributes from boardgamegeek.com (the IMDb of board games). Our data includes the following variables:

- **Response Variable**

  - **avg_rating:** The average rating of the board game. This variable ranges between 0 and 100, and comes from boardgamegeek.com.

- **Predictors**

  - **ID**
    - **name:** The name of the game
    - **game_id:** The unique identifier of the game
  - **Quantitative**
    - **ranking:** The game's rank
    - **avg_timeplay:** On average, how long does it take to play one game
    - **min_timeplay:** minimum required time to play the game
    - **max_timeplay:** maximum required time to play the game
    - **year:** release year of the game
    - **num_votes:** number of people rating the game
    - **min_players:** minimum number of players that can play the game
    - **max_players:** maximum number of players that can play the game
    - **age:** recommended minimum age to play the game
    - **weight:** board game weight (in pounds)
  - **Categorical**
    - **designer:** who designed the game
    - **category:** board game category

[1] I personally recommend you play "Splendor." Such a cool board game.

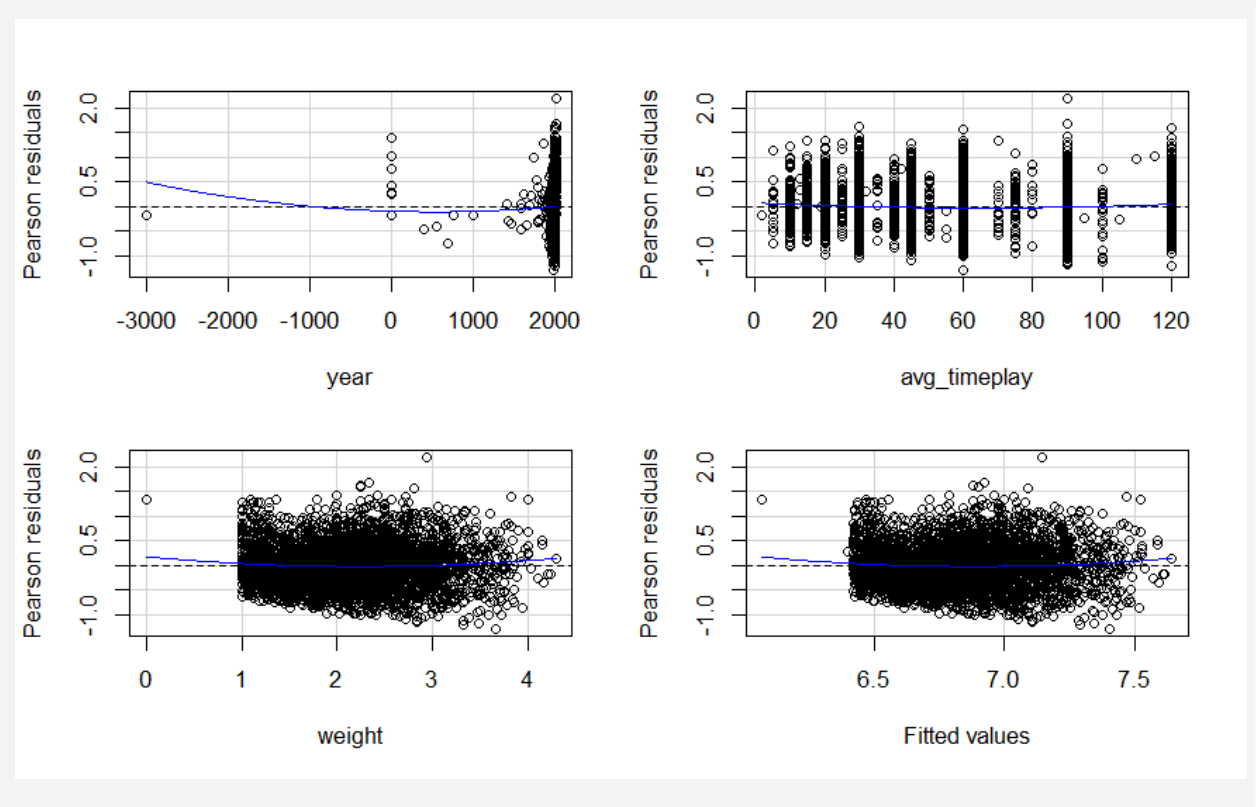
1. Model Issues: Non-linearity
(5 points - Lecture 4)

Run the following model:

$$avg\_rating = b_0 + b_1(year) + b_2(avg\_timeplay) + b_3(weight)$$

A. (2 points) We will assess (i) the linearity of each predictor and (ii) the linearity of the model as a whole. Use the residualPlots() function from the "car" package, and paste the residual plots (and the numerical linearity results) below.

Residual plots



Numerical Non-Linearity results

(table displayed that includes Tukey test)

```
              Test stat Pr(>|Test stat|)
year             1.3980         0.162185
avg_timeplay     3.2107         0.001336 **
weight           2.8029         0.005092 **
Tukey test       2.8197         0.004807 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

B. (1 point) Based on the results above, which variables violate the linearity assumption? How did you reach this conclusion?

Your answer:
Variables `avg_timeplay` and `weight` are not linear, since their reported p-value in the Tukey test is less than 0.10.

C. (2 points) What does the Tukey test tell you about our model? Is linear regression a good fit for this data, overall? What statistical techniques do you think we would need to use to improve the statistical fit of this model?

Your answer:
The Tukey test tells us that the specified model is non-linear as it has an overall p-value of 0.004, which is lower than the 0.10 limit.

Linear regression does not seem like a good fit for this data. To improve the statistical fit of this model, we may need to use polynomial predictors to capture the true relationship.

2. Model Issues: Heteroskedasticity (5 points - Lecture 4)

A. (1 point) What does heteroskedasticity mean? What is the effect of heteroskedasticity on a linear model?

Your answer:
Heteroskedasticity means non-constant variance of the error term. If a model has heteroskedasticity, the standard errors are biased, leading to artificially high or low p-values, and incorrect significance tests.

Run the following model:

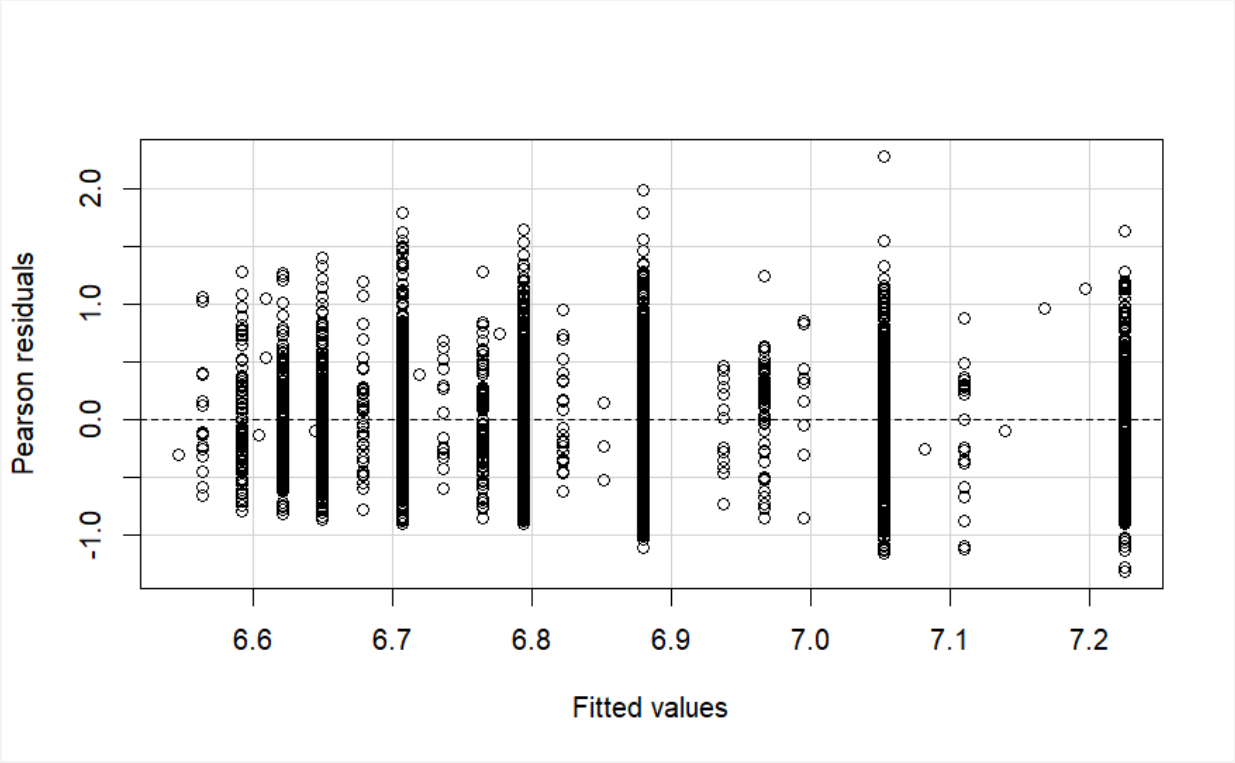
$$avg\_rating = b_0 + b_1(avg\_timeplay)$$

B. (0.5 point) Paste the regression output below.

```
Call:
lm(formula = avg_rating ~ avg_timeplay)

Residuals:
     Min       1Q   Median       3Q      Max
-1.31553 -0.35511 -0.04037  0.31530  2.27931

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.5342724  0.0164293  397.72   <2e-16 ***
avg_timeplay 0.0057565  0.0002678   21.49   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4904 on 3597 degrees of freedom
Multiple R-squared:  0.1138,    Adjusted R-squared:  0.1136
F-statistic: 461.9 on 1 and 3597 DF,  p-value: < 2.2e-16
```

C. (0.5 point) Do a visual funnel test. Paste the plot below. In one sentence, tell me if you find visual evidence of heteroskedasticity in the model.

Your plot:



Your answer:
Yes, there is heteroskedasticity as the residuals spread out as we move to the right on the X-axis

D. (1 point) Do a NCV test. Paste the results below. What can we conclude from this test?

Paste results here:

```
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 32.38041, Df = 1, p = 1.2676e-08
```

Your conclusion about the NCVTest:
p-value < 0.05 indicates that the model has heteroskedasticity.

E. (1 point) Run a heteroskedasticity-corrected model and paste the regression output below:

```
t test of coefficients:

              Estimate Std. Error t value  Pr(>|t|)
(Intercept)  6.53427237 0.01586292 411.921 < 2.2e-16 ***
avg_timeplay 0.00575647 0.00027626  20.837 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

F. (1 point) In your new model, what happened to (i) the coefficients and (ii) the t-value and p-values of the predictor?

Your answer:
The coefficients remained the same for the predictor (`avg_timeplay`) in both the models.
The t-value of the predictor decreased slightly, and the p-value increased slightly.

3. Model Issues: Outliers
(4 points - Lecture 4)

Run the following model:

$$avg\_rating = b_0 + b_1(min\_players) + b_2(age) + b_2(num\_votes)$$

A. (1 point) Do a Bonferroni test to formally identify the outliers. Paste the Bonferroni test results below:

	rstudent <dbl>	unadjusted p-value <dbl>	Bonferroni p <dbl>
3124	4.630256	3.7814e-06	0.013609

1 row

B. (1 point) List the name of the games that are flagged as outliers by the Bonferroni test:

Your answer:
"Mythic Battles: Pantheon"

C. (2 points) Remove any outliers you identified, and re-run the regression without these outliers. Did you notice any changes in the model?

Your answer:
After removing the single outlier, we do not see any significant changes in the model.

The model previously had an adjusted R-squared of 0.1131, while the model with the outlier removed has an adjusted R-squared of 0.1127.

Similarly, the coefficients of the predictors did not change by much, while the standard errors went down slightly.

4. Model Issues: Collinearity
(6 points - Lecture 4)

A. (1 point) Define, in your own words, what collinearity is between two variables. Why is it problematic?

Your answer:
Collinearity occurs when one variable can be predicted using the other variable, i.e., they are linearly related.

It is problematic since including both the variables would lead to double counting that particular variable, leading to bad estimates for both the variables.

Suppose you want to run the following model:

$$avg\_rating = b_o + b_1(year) + b_2(age) + b_3(min\_timeplay) + b_4(max\_timeplay)$$

B. (1 point) Before running this model, you need to make sure that there is no collinearity between the predictors. Create the correlation matrix for the predictors.

	min_timeplay	max_timeplay	year	age
min_timeplay	1.00	0.88	0.03	0.30
max_timeplay	0.88	1.00	0.03	0.34
year	0.03	0.03	1.00	0.08
age	0.30	0.34	0.08	1.00

C. (1 point) Based on this correlation matrix, which pair(s) of predictors are collinear?

Your answer:

min_timeplay and **max_timeplay**, since the absolute value is > 0.80 in correlation matrix.

D. (1 point) Now, run a VIF test for the same model. Based on this test, which variables are collinear?

Your answer:

min_timeplay and **max_timeplay** are collinear, since the VIF > 4 for both these variables.

E. (2 points) Why do you think there is collinearity between these two variables? Would you have suspected it? Why? Explain your reasoning in one short paragraph.

Your answer:
The collinearity between the two variables is expected. In most cases, if a game requires a long time to play at minimum, it is likely that the maximum time required will also be long, and vice versa. Thus, it is reasonable to suspect collinearity between these two variables.

F. (Optional) Using the VIF test as a reference, get rid of the collinearity problem. Re-run the model (you may eliminate any of the collinear variables). Do you notice any change for any of the coefficients?

Your answer:
The model with collinear variables had a negative coefficient for `min_timeplay`, while the model with the `max_timeplay` removed has a positive coefficient for `min_timeplay`. The coefficients for the other predictors have also changed slightly, but the major change is for the variable that was linearly related to the dropped variable.

5. Presenting professional regression tables (3 points)

A. (1.5 points) As a business data analyst, your responsibility is to not only design and run predictive models, but to also present these results to managers, policymakers, executives, and board directors.

For this reason, the goal of this course goes beyond simply running regressions. We will also learn how to interpret and present great statistical reports—reports that are beautiful, sleek, well-written, and aesthetically pleasing.

We will now learn how to present regression tables. If you simply copy/paste the regression output from R (as you've done so far), your reports will look unimpressive and messy. Fortunately, there is a tool called Stargazer, which is ubiquitously used by statisticians. Stargazer takes your R output and formats it in a manner that is internationally accepted by the data-science community.

To create a table using Stargazer, follow the steps below:

- Step 1: Open the board_games dataset and run the following four linear regressions (call them *reg1*, *reg2*, *reg3*, *mreg*)

- reg1: avg_rating = b_o+ b_1(avg_timeplay)
- reg2: avg_rating = b_o+ b_2(min_players)
- reg3: avg_rating = b_o+b_3(max_players)
- mreg: avg_rating = b_o+ b_1(avg_timeplay)+ b_2(min_players)+b_3(max_players)

- Step 2: Run the following code:

```
>install.packages("stargazer")

>library(stargazer)

>stargazer(reg1, reg2, reg3, mreg, type="html")
```

- Step 3: You should get a long html code. Copy this code, and paste it on this link below:

https://www.w3schools.com/tryit/tryit.asp?filename=tryhtml_default

(Note: after opening the website link, you'll find some pre-written code. Delete this code, paste your own, and click *run*).

- Step 4: You should be getting a nice looking table with four regressions. Below, paste a screenshot of this table:

	Dependent variable:			
	avg_rating			
	(1)	(2)	(3)	(4)
avg_timeplay	0.006***			0.006***
	(0.0003)			(0.0003)
min_players		-0.138***		-0.124***
		(0.012)		(0.012)
max_players			-0.003**	0.0005
			(0.001)	(0.001)
Constant	6.534***	7.129***	6.855***	6.799***
	(0.016)	(0.027)	(0.011)	(0.030)
Observations	3,599	3,599	3,599	3,599
R^2	0.114	0.033	0.002	0.140
Adjusted R^2	0.114	0.033	0.001	0.139
Residual Std. Error	0.490 (df = 3597)	0.512 (df = 3597)	0.520 (df = 3597)	0.483 (df = 3595)
F Statistic	461.917*** (df = 1; 3597)	123.028*** (df = 1; 3597)	5.941** (df = 1; 3597)	195.058*** (df = 3; 3595)
Note:				*p<0.1; **p<0.05; ***p<0.01

As you can see, we have four different regressions, which are compiled in a single table. This is the standard format for presenting tables in the data science community. Each column represents a different regression: Column (1) represents *reg1*, Column (2) represents *reg2*, etc.

B. (0.5 point) In R, type *summary(reg2)* and compare the R output with your stargazer table (in Column 2). Now, tell me, what does the value -0.138 represent in this regression, and what does the number below it (0.012) represent?

Your answer:

-0.138 represents the coefficient value for the `min_rating` predictor. It is interpreted as the change in the `avg_rating` of the game for a unit increase in `min_players`, holding all other variables constant.
The number 0.012 represents the standard error of the coefficient estimate for the variable `min_players`. It measures the variability or uncertainty of the coefficient estimate.

C. (0.5 point) Now, I'd like you to present the tables with proper variable names. It's certainly unprofessional to present the variables with the name "min_players" or "max_players." Stargazer allows you to rename variables. I would like you to rename, within the stargazer code, the variable names to "Board game rating," "Average timeplay," "Minimum number of players," "Maximum number of players." In other words, I'm not asking you to rename the variables in the dataset, but rather to rename in the stargazer code (Hint: This link might help you):

Paste your code:

Your code:

```
stargazer(reg1, reg2, reg3, mreg,
        title = "Regression Results",
        type = "html",
        align = TRUE,
        dep.var.labels = c("Board Game Rating"),
             covariate.labels = c("Average timeplay", "Minimum number of players",
"Maximum number of players"))
```

Below, paste the stargazer table with proper variable names:

Regression Results

	Dependent variable:			
	Board Game Rating			
	(1)	(2)	(3)	(4)
Average timeplay	0.006***			0.006***
	(0.0003)			(0.0003)
Minimum number of players		-0.138***		-0.124***
		(0.012)		(0.012)
Maximum number of players			-0.003**	0.0005
			(0.001)	(0.001)
Constant	6.534***	7.129***	6.855***	6.799***
	(0.016)	(0.027)	(0.011)	(0.030)
Observations	3,599	3,599	3,599	3,599
R^2	0.114	0.033	0.002	0.140
Adjusted R^2	0.114	0.033	0.001	0.139
Residual Std. Error	0.490 (df = 3597)	0.512 (df = 3597)	0.520 (df = 3597)	0.483 (df = 3595)
F Statistic	461.917*** (df = 1; 3597)	123.028*** (df = 1; 3597)	5.941** (df = 1; 3597)	195.058*** (df = 3; 3595)
Note:				*p<0.1; **p<0.05; ***p<0.01

D. (0.5 point) Now, present the stargazer table with the coefficients and standard errors rounded to two decimal places (Hint: This link might help you):

Paste your code:

Your code:

```
(reg1, reg2, reg3, mreg,
        title = "Regression Results",
        type = "html",
        align = TRUE,
        dep.var.labels = c("Board Game Rating"),
             covariate.labels = c("Average timeplay", "Minimum number of players",
"Maximum number of players"),
        digits = 2
)
```

Below, paste the stargazer table with results using only two decimal places:

Regression Results

	Dependent variable:			
	Board Game Rating			
	(1)	(2)	(3)	(4)
Average timeplay	0.01***			0.01***
	(0.0003)			(0.0003)
Minimum number of players		-0.14***		-0.12***
		(0.01)		(0.01)
Maximum number of players			-0.003**	0.0005
			(0.001)	(0.001)
Constant	6.53***	7.13***	6.86***	6.80***
	(0.02)	(0.03)	(0.01)	(0.03)
Observations	3,599	3,599	3,599	3,599
R^2	0.11	0.03	0.002	0.14
Adjusted R^2	0.11	0.03	0.001	0.14
Residual Std. Error	0.49 (df = 3597)	0.51 (df = 3597)	0.52 (df = 3597)	0.48 (df = 3595)
F Statistic	461.92*** (df = 1; 3597)	123.03*** (df = 1; 3597)	5.94** (df = 1; 3597)	195.06*** (df = 3; 3595)
Note:				*p<0.1; **p<0.05; ***p<0.01

If you look at any scientific statistical report, you'll find that scientists use this exact format. For example, look at page 10 of this report (from Harvard professors), or any report for that matter.

There you go—you've become a star….gazer!

6. Polynomial regression: Age
(5 points - Lecture 5)

Let's figure out the true functional relationship between *avg_rating* and *age*:

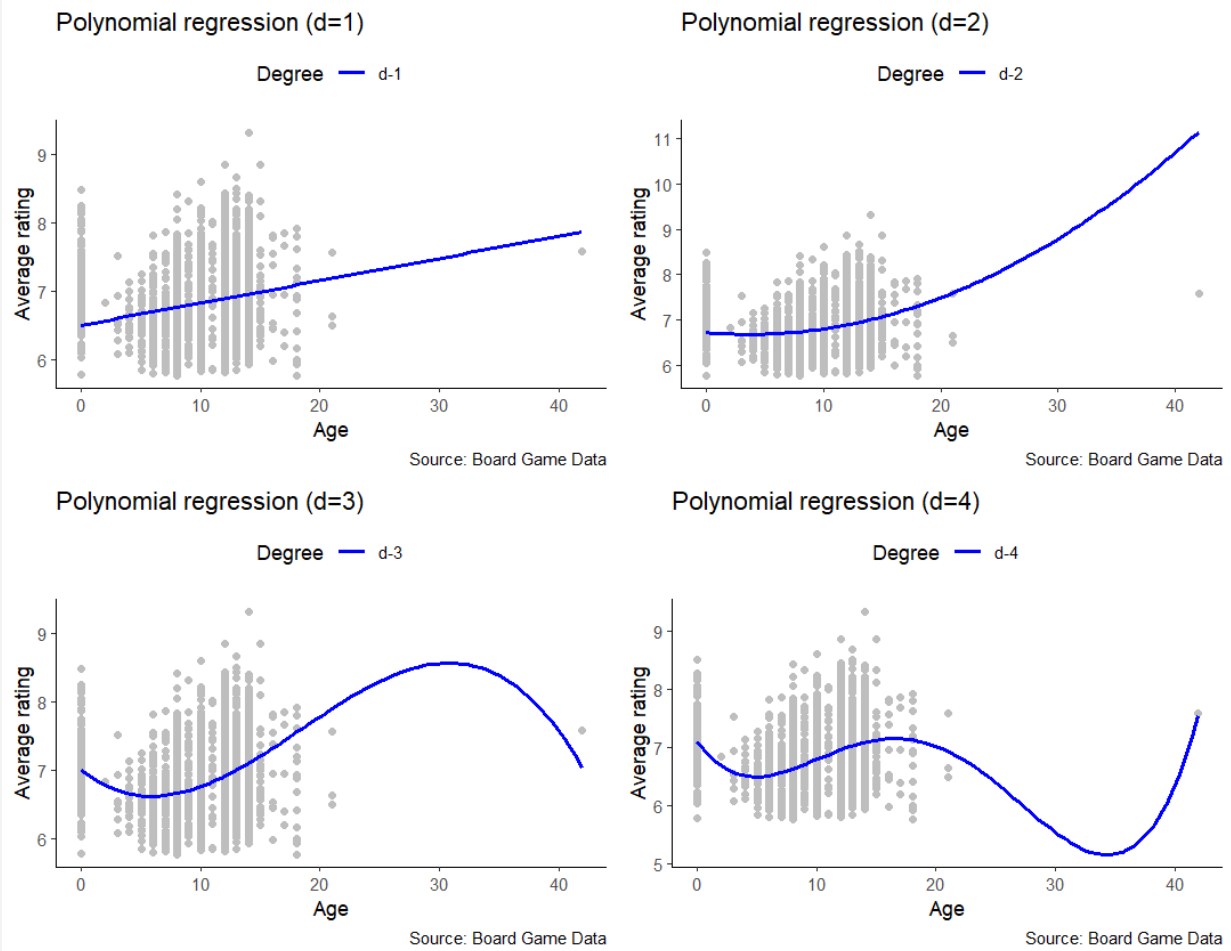
avg_rating = f(age)

A. (2 points) Try running four regressions: a linear (d=1), quadratic (d=2), cubic (d=3), and quartic (d=4). Paste the output of these four regressions below (using stargazer). This time, I want a table with five coefficient rows: one row for the intercept and one for each polynomial degree. Note: To achieve this, you should rename the coefficients in the table's output so that all polynomials of the same degree are in the same row.
    - For example, poly(age,3)2 and poly(age,4)2 should be renamed as age^2 and appear in the same row.
    - poly(age,3)3 and poly(age,4)3 should be renamed to age^3 and appear in the same row.

Regression Results

	Dependent variable:			
	Board Game Rating			
	Linear (d=1)	Quadratic (d=2)	Cubic (d=3)	Quartic (d=4)
	(1)	(2)	(3)	(4)
Constant	6.51***	6.84***	6.84***	6.84***
	(0.03)	(0.01)	(0.01)	(0.01)
Age	0.03***	5.95***	5.95***	5.95***
	(0.003)	(0.51)	(0.49)	(0.49)
Age^2		4.48***	4.48***	4.48***
		(0.51)	(0.49)	(0.49)
Age^3			-6.33***	-6.33***
			(0.49)	(0.49)
Age^4				3.71***
				(0.49)
Observations	3,599	3,599	3,599	3,599
R^2	0.04	0.06	0.10	0.11
Adjusted R^2	0.04	0.06	0.10	0.11
Residual Std. Error	0.51 (df = 3597)	0.51 (df = 3596)	0.49 (df = 3595)	0.49 (df = 3594)
F Statistic	135.43*** (df = 1; 3597)	108.34*** (df = 2; 3596)	130.06*** (df = 3; 3595)	113.38*** (df = 4; 3594)
Note:				*p<0.1; **p<0.05; ***p<0.01

B. (2 points) Create a matrix of four scatterplots (two columns, two rows) with the corresponding polynomial fit in each graph, using GGPlot. Make sure that the fitted polynomials are in blue, the dots are in grey, and that there is a legend (indicating the degree of the polynomial). Paste the matrix of graphs below. Hint: par(mfrow) doesn't work with GGPlot, to create a matrix of plots. But you can use ggarrange(), following the commands from this website:



C. (0.5 point) Run an ANOVA test to determine the optimal polynomial model (between the linear, quadratic, cubic and quartic regressions). Post the results below:

```
Analysis of Variance Table

Model 1: avg_rating ~ poly(age, degree = 1)
Model 2: avg_rating ~ poly(age, degree = 2)
Model 3: avg_rating ~ poly(age, degree = 3)
Model 4: avg_rating ~ poly(age, degree = 4)
  Res.Df    RSS Df Sum of Sq       F    Pr(>F)
1   3597 940.69
2   3596 920.63  1    20.056  83.165 < 2.2e-16 ***
3   3595 880.54  1    40.097 166.264 < 2.2e-16 ***
4   3594 866.74  1    13.799  57.219 4.922e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

D. (0.5 point) Based on the above results, which model would you consider appropriate to describe this functional relationship ?

i) Linear (d=1) - No

ii) Quadratic (d=2) - No

iii) Cubic (d=3) - Yes

iv) Quartic (d=4) - No

7. Polynomial regression: avg_timeplay (5 points - Lecture 5)

Let's figure out the true functional relationship between *rating* and *avg_timeplay*:

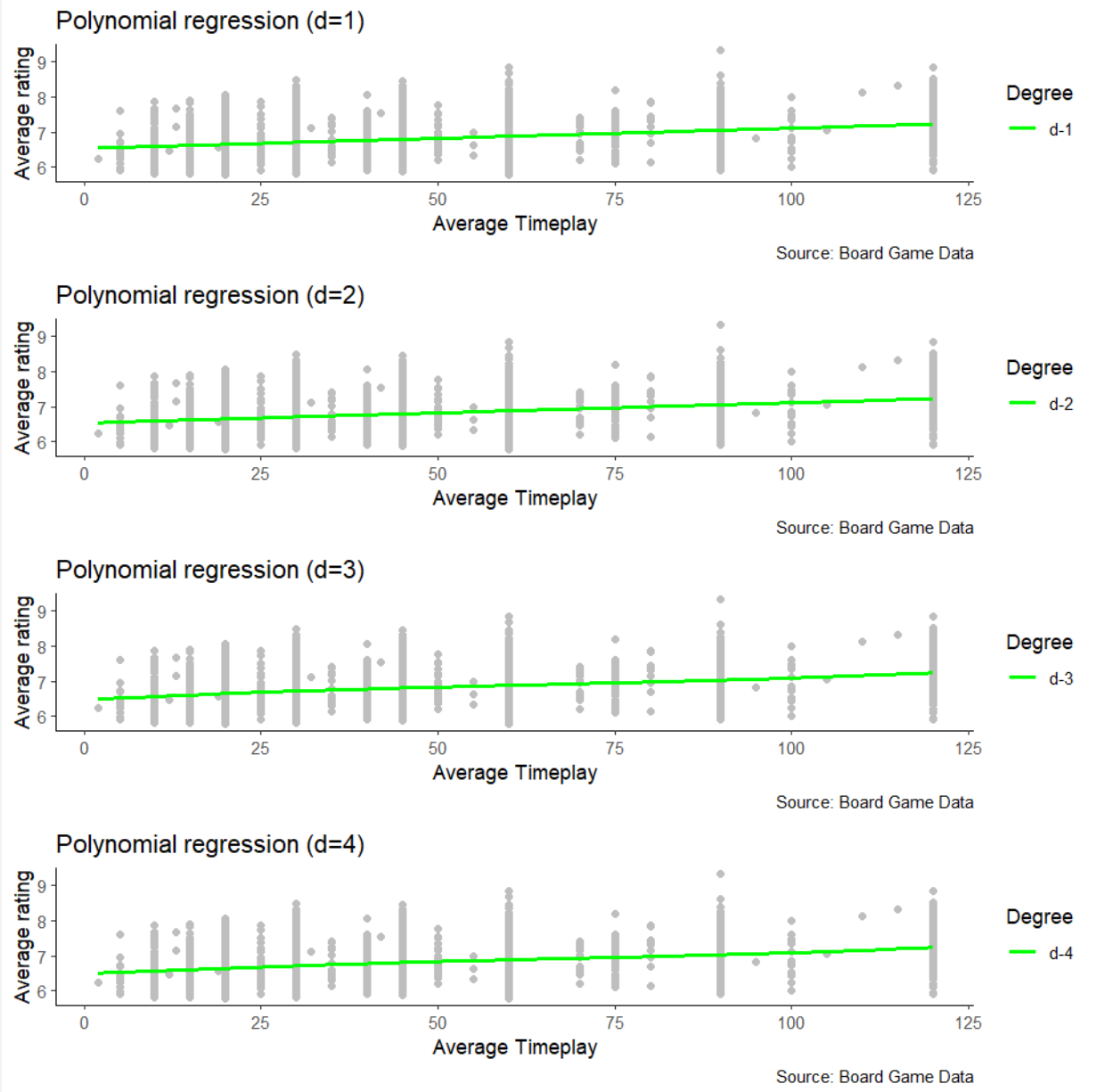
rating = f(avg_timeplay)

A. (2 points) Try running four regressions: a linear (d=1), quadratic (d=2), cubic (d=3), and quartic (d=4). Paste the output of these four regressions below (using stargazer). This time, I want a table with five coefficient rows: one row for the intercept and one for each polynomial degree. Note: To achieve this, you should rename the coefficients in the table's output so that all polynomials of the same degree are in the same row.
    - For example, poly(x,3)2 and poly(x,4)2 should be renamed as x^2 and appear in the same row.
    - poly(x,3)3 and poly(x,4)3 should be renamed to x^3 and appear in the same row .

Regression Results (avg_rating ~ avg_timeplay)

	Dependent variable:			
	Board Game Rating			
	Linear (d=1)	Quadratic (d=2)	Cubic (d=3)	Quartic (d=4)
	(1)	(2)	(3)	(4)
Constant	6.53***	6.84***	6.84***	6.84***
	(0.02)	(0.01)	(0.01)	(0.01)
x	0.01***	10.54***	10.54***	10.54***
	(0.0003)	(0.49)	(0.49)	(0.49)
x^2		-0.02	-0.02	-0.02
		(0.49)	(0.49)	(0.49)
x^3			0.81	0.81
			(0.49)	(0.49)
x^4				0.19
				(0.49)
Observations	3,599	3,599	3,599	3,599
R^2	0.11	0.11	0.11	0.11
Adjusted R^2	0.11	0.11	0.11	0.11
Residual Std. Error	0.49 (df = 3597)	0.49 (df = 3596)	0.49 (df = 3595)	0.49 (df = 3594)
F Statistic	461.92*** (df = 1; 3597)	230.90*** (df = 2; 3596)	154.91*** (df = 3; 3595)	116.19*** (df = 4; 3594)
Note:				*p<0.1; **p<0.05; ***p<0.01

B. (2 points) Create a matrix of four scatterplots (one column, four rows) with the corresponding polynomial fit in each graph, using GGPlot. Make sure that the fitted polynomials are in green, the dots are in grey, and that there is a legend (indicating the degree). Paste the matrix of graphs below:



C. (0.5 point) Run an ANOVA test to determine the optimal polynomial model (between the linear, quadratic, cubic and quartic regressions). Post the results below:

```
Analysis of Variance Table

Model 1: avg_rating ~ poly(avg_timeplay, degree = 1)
Model 2: avg_rating ~ poly(avg_timeplay, degree = 2)
Model 3: avg_rating ~ poly(avg_timeplay, degree = 3)
Model 4: avg_rating ~ poly(avg_timeplay, degree = 4)
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1   3597 865.02
2   3596 865.02  1   0.00062 0.0026 0.9596
3   3595 864.37  1   0.65040 2.7044 0.1002
4   3594 864.33  1   0.03784 0.1573 0.6916
```

D. (0.5 point) Based on the above results, which model would you consider most appropriate to describe this functional relationship ?

i) Linear (d=1) - Yes

ii) Quadratic (d=2) - No

iii) Cubic (d=3) - No

iv) Quartic (d=4) - No

8. Multiple polynomial regression (2 points - Lecture 5)

Consider the following relationship:

avg_rating = f(age, avg_timeplay)

A. (1 point) Run the regression using the best polynomial relationship you found for *age* (in Question #6) and *avg_timeplay* (in Question #7) , and combine them in a multiple polynomial regression. Please write the multiple polynomial regression equation to be estimated (in the form of avg_rating=b_o+b_1.....)

Your regression equation:

$$avg\_rating = b_0 + b_1 * avg\_timeplay + b_2 * age + b_3 * age^2 + b_4 * age^3$$

B. (1 point) Run the multiple polynomial regression, and paste the results of this regression below:

Regression Results

	Dependent variable:
	Board Game Rating
Average Timeplay	0.005***
	(0.0003)
Age	3.14***
	(0.51)
Age^2	4.07***
	(0.48)
Age^3	-4.61***
	(0.49)
Constant	6.60***
	(0.02)
Observations	3,599
R^2	0.16
Adjusted R^2	0.16
Residual Std. Error	0.48 (df = 3594)
F Statistic	168.11*** (df = 4; 3594)
Note:	*p<0.1; **p<0.05; ***p<0.01

9. Spline regression
(5 points - Lecture 5)

A. (1 point) What is the difference between a polynomial regression and spline regression? Why would we want to use splines as opposed to polynomials?

Your answer:

Polynomial regression involves creating new features to fit curves rather than straight lines, allowing for flexibility, but forces the function to behave in the same manner throughout the x-axis.

Spline regression, on the other hand, provides the ability to create a continuous function by connecting linear or polynomial segments through knots.

B. (2 points) Let's focus on the following relationship :

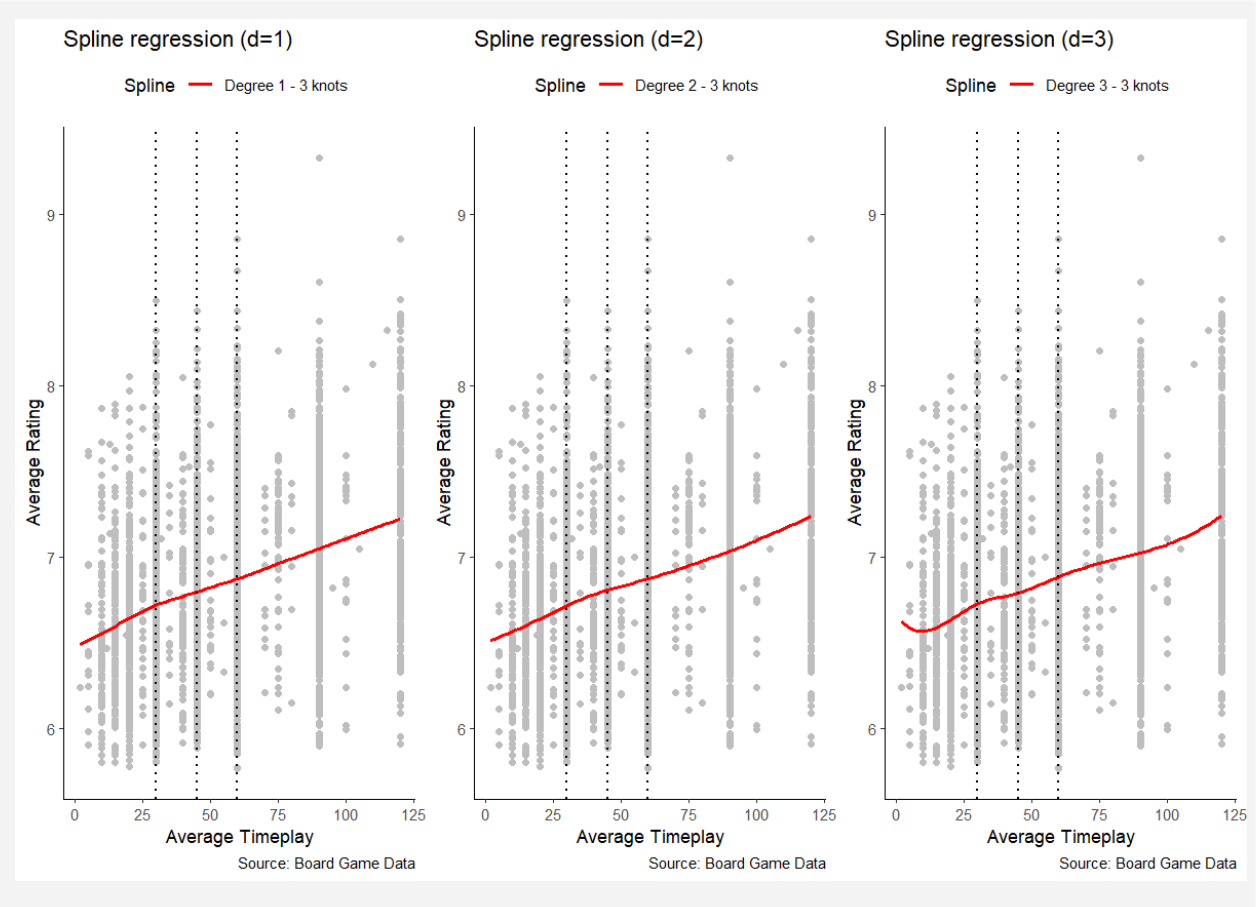
avg_rating = f(avg_timeplay)

For the above relationship, run three spline regressions with three knots each: (i) a linear spline; (ii) a quadratic spline; and (iii) a cubic spline. Space the knots uniformly across the quantiles of the data, using the method we learned in class. Please paste the three regressions output below (using stargazer)

Spline Regression Results (avg_rating ~ avg_timeplay)

	Dependent variable:		
	Board Game Rating		
	d=1	d=2	d=3
	(1)	(2)	(3)
bs(avg_timeplay, knots = knots, degree = 1)1	0.23***		
	(0.05)		
bs(avg_timeplay, knots = knots, degree = 1)2	0.31***		
	(0.05)		
bs(avg_timeplay, knots = knots, degree = 1)3	0.38***		
	(0.05)		
bs(avg_timeplay, knots = knots, degree = 1)4	0.73***		
	(0.05)		
bs(avg_timeplay, knots = knots, degree = 2)1		0.09	
		(0.14)	
bs(avg_timeplay, knots = knots, degree = 2)2		0.26***	
		(0.08)	
bs(avg_timeplay, knots = knots, degree = 2)3		0.32***	
		(0.10)	
bs(avg_timeplay, knots = knots, degree = 2)4		0.50***	
		(0.10)	
bs(avg_timeplay, knots = knots, degree = 2)5		0.72***	
		(0.10)	
bs(avg_timeplay, knots = knots, degree = 3)1			-0.15
			(0.21)
bs(avg_timeplay, knots = knots, degree = 3)2			0.13
			(0.13)
bs(avg_timeplay, knots = knots, degree = 3)3			0.15
			(0.15)
bs(avg_timeplay, knots = knots, degree = 3)4			0.39***
			(0.15)
bs(avg_timeplay, knots = knots, degree = 3)5			0.38**
			(0.16)
bs(avg_timeplay, knots = knots, degree = 3)6			0.62***
			(0.14)
Constant	6.49***	6.51***	6.63***
	(0.04)	(0.09)	(0.14)
Observations	3,599	3,599	3,599
R^2	0.11	0.11	0.11
Adjusted R^2	0.11	0.11	0.11
Residual Std. Error	0.49 (df = 3594)	0.49 (df = 3593)	0.49 (df = 3592)
F Statistic	115.98*** (df = 4; 3594)	92.84*** (df = 5; 3593)	77.77*** (df = 6; 3592)
Note:			*p<0.1; **p<0.05; ***p<0.01

C. (2 points) Create a matrix of three distinct scatterplots, using GGPlot. In each graph, put each regression fitted splines, with (i) splines in red and (ii) vertical dashed lines where the knots are located; and (iii) a legend indicating the degree of the spline, and the number of knots.



10. Validation set test
(4 points - Lecture 6)

A. (3 points) Consider the following statistical relationship:

avg_rating = f(weight)

Perform a validation set test for different polynomial models (d=1…,10). Repeat this test 30 times for each model, and paste the average of the 30 tests. You should write down the 10 average MSEs. Note: you need to automate these 30 tests using loops (write the code below):

Your R code:

```
validation_mse <- function(degree, num_iter) {
  fit_dn <-
    lm(formula = if (degree == 1)
      avg_rating ~ weight
      else
        (avg_rating ~ poly(weight, degree))
      ,
      data = train_set)

  actual = test_set$avg_rating

  mses = rep(NA, num_iter)

  for (i in 1:num_iter) {
    prediction = predict(fit_dn, test_set)
    squared_error = (actual - prediction) ^ 2
    mses[i] = mean(squared_error)
  }

  mean(mses)
}

mse_degree = rep(NA, 10)

for (i in 1:10) {
  mse_degree[i] = validation_mse(i, 30)
  print(glue("d={i}: {format(mse_degree[i], digits=3)}"))
}

which.min(mse_degree)
min(mse_degree)
```

Paste the average MSEs of the 30 tests for each model:

Average MSEs (3 decimal points):

d=1: 0.204
d=2: 0.204
d=3: 0.202
d=4: 0.202
d=5: 0.202
d=6: 0.203
d=7: 0.203
d=8: 0.203
d=9: 0.203
d=10: 0.206

B. (1 point) Based on your results, which polynomial function would you use?

Your answer:
Based on the results above, a polynomial function of degree 3 would be used as it has the lowest MSE.

11. LOOCV test
(7 points - Lecture 6)

Suppose we want to test the following statistical relationship:

avg_rating = f(weight)

A (1 point) In a given trial of a LOOCV test (in our dataset), how many sets are created, and what are their sizes?

Your answer:
For the given dataset, in a LOOCV test, a single test would have 3598 training observations and 1 test observation.

B. (1 point) In our dataset, how many tests would a LOOCV perform?

Your answer:
3599 tests would be conducted

C. (1 points) Perform a LOOCV for the simple regression model (d=1). Put a timer before you run the test. What was the result of this test?

Your answer:
MSE for LOOCV (d=1): 0.210
Time taken: ~17 seconds

D. (3 points) Now, run a LOOCV for the ten models (d=1,…10). Note: This test will take between a few minutes.

Average MSEs (3 decimal points):

d=1: 0.21
d=2: 0.21
d=3: 0.209
d=4: 0.209
d=5: 0.209
d=6: 0.208
d=7: 0.208
d=8: 0.882
d=9: 0.598

d=10: 9.79

E. (1 points) Based on your results, which polynomial function would you use?

Your answer:
Based on the above results, a polynomial function of degree 6 would be used.

12. K-fold cross-validation
(5 points - Lecture 6)

A. (1 point) Suppose I have a dataset with 30 observations=(Y_1,X_1), (Y_2,X_2),…,(Y_{30},X_{30}). If I have five folds, how many tests would I perform?

Your answer:
With 5 folds, we will perform 5 tests.
Each test would have a training size of 24 observations and a test size of 6 observations

We want to perform a K-fold cross validation for the statistical relationship:

avg_rating = f(num_votes)

B. (3 points) Perform a 5-fold test for the ten models (d=1, d=2,..., d=10). Make sure the test is automated. Paste the code and the results below:

Your code:

```
k_fold_mse <- function(degree, k) {
  k_fold_fit_dn <- glm(formula = if (degree == 1)
    avg_rating ~ num_votes
    else
      (avg_rating ~ poly(num_votes, degree))
    ,
    data = data)

  mse_k_fold_fit_dn <- cv.glm(data = data, k_fold_fit_dn, K = k)$delta[1]

  return (mse_k_fold_fit_dn)
}

mse_k_fold_list <- rep(NA, 10)

for (i in 1:10) {
  mse_k_fold_list[i] = k_fold_mse(i, 5)
  print(glue("d={i}: {format(mse_k_fold_list[i], digits = 3)}"))
}
```

```
which.min(mse_k_fold_list)
min(mse_k_fold_list)
```

Average MSEs (3 decimal points):

d=1: 0.259
d=2: 0.256
d=3: 0.256
d=4: 0.256
d=5: 0.255
d=6: 0.256
d=7: 0.257
d=8: 0.598
d=9: 0.279
d=10: 0.261

C. (1 point) Based on your results, which polynomial function would you use?

Your answer:
Based on the above results, a polynomial function of degree 5 would be used.

13. Cross-validation in a multiple spline model (5 points - Lecture 6)

Consider the following relationship:

avg_rating = f(age, year, num_votes, avg_timeplay)

We want to have the model with the best out-of-sample performance (aka. predictive performance). Each spline will have three knots, at the 25th, 50th, and 75th percentile of the distribution of each predictor:

$$avg\_rating = f_{spline(\#\ of\ knots=3, degree=a)}(age) + f_{spline(\#\ of\ knots=3, degree=b)}(year)$$

$$+ f_{spline(\#\ of\ knots=3, degree=c)}(num\_votes) + f_{spline(\#\ of\ knots=3, degree=d)}(avg\_timeplay)$$

We want to determine if we should use

i) a linear spline (degree=1)
ii) a quadratic spline (degree=2); or
iii) a cubic spline (degree=3); or
iv) a quartic spline (degree=4); or
v) a quintic spline (degree-5)

In other words, you will need to find the optimal combination of (a,b,c,d). Note, there are hundreds of combinations possible.

To find this optimal combination, you will run a K-fold validation set (with K=20). In other words, you will need a 20-fold test for each combination of (a,b,c,d), and then determine which combination gives you the lowest MSE. Hint: You will need to program multiple loops.

(Depending on how efficient your code is, the calculations should take between 5 to 30 minutes on a regular computer.)

A. (3 points) Paste your code below:

Your code:

```
knots_age <- quantile(age, probs = c(0.25, 0.50, 0.75))
knots_year <- quantile(year, probs = c(0.25, 0.50, 0.75))
knots_num_votes <- quantile(num_votes, probs = c(0.25, 0.50, 0.75))
knots_avg_timeplay <- quantile(avg_timeplay, probs = c(0.25, 0.50, 0.75))

k_fold_spline <- function(a, b, c, d, k) {
  fit_model <- glm(
    formula =
      avg_rating ~
      bs(age, knots = knots_age, degree = a) +
      bs(year, knots = knots_year, degree = b) +
      bs(num_votes, knots = knots_num_votes, degree = c) +
      bs(avg_timeplay, knots = knots_avg_timeplay, degree = d)
  )

  mse <- cv.glm(data=data, fit_model, K=k)$delta[1]

  return (mse)
}

mse_spline_df = data.frame(
  a = rep(NA, 5 ^ 4),
  b = rep(NA, 5 ^ 4),
  c = rep(NA, 5 ^ 4),
  d = rep(NA, 5 ^ 4),
  mse = rep(NA, 5 ^ 4)
)


for (a in 1:5) {
  for (b in 1:5) {
    for (c in 1:5) {
      for (d in 1:5) {
        print(glue("Running 20-fold CV for a,b,c,d = {a},{b},{c},{d}"))

        mse_spline_df[nrow(mse_spline_df) + 1, ] <-
          c(a, b, c, d, k_fold_spline(a, b, c, d, 20))
      }
    }
  }
}

mse_spline_df[which.min(mse_spline_df$mse),]
```

Note: Two years ago, for a similar question, someone shared her/his code with a FRIEND, in good faith—this friend, in turn, shared it with a group of friends, and so on. Long story short, about 70% of the class ended up with the exact code (and you can imagine the rest of the story). While you're encouraged to collaborate, do not pass along codes as a "favour" to your classmates—no matter how much you trust them—because these codes will probably spread out. I would like to avoid this situation from ever happening, by warning you ahead of time. This is not an easy question, but it's a fantastic one to train to become a good coder. Just do your best and take it as a challenge to train your coding skills. I am generous when giving part marks for good attempts. But if anyone asks you for the code as a favour, just tell them: "*Stargazers never give up their codes*"

B) (2 points) Which model has the best out-of-sample performance?

Your answer:

- a: 1
- b: 2
- c: 4
- d: 5

Your MSE for this optimal combination:

0.151

Note: This is how data scientists test models in real life. But they typically have super computers running this type of test for very large statistical models, looking for millions of possible combinations to continually improve their predictive capabilities! In fact, during the midterm you'll have to predict a real life event, and you'll be doing much of this to find the best prediction.

- Instructions: Please save in colour as a PDF and submit by the beginning of class, via mycourses. If you don't submit as a PDF, you'll be subject to a 2-point deduction
- Please attach your code when you submit Lab 2 (as a separate file). It will help us see where you made mistakes
- Due date: October 15 at 11:59 pm (Montreal time).