Lab 2

Michael O'Neil

2025-01-04

Lab 1 | Regressions | Walkthrough

In this walkthrough, our goal will be to use a regression to predict the salary of an NHL player. (The NHL is the world's best professional ice hockey league.)

To predict the salaries, we will use a dataset containing various metrics about every player in the pre-cited league during the 2016-17 season.

Step 1 | Exploratory Data Analysis and Data Cleaning

The first step in any project is to load and explore the data. The exploration, commonly referred to as Exploratory Data Analysis (EDA), is essential as it helps us spot weaknesses in the data (duplicate rows, missing data, outliers, etc.) and can guide our decision in how to manipulate it before fitting a model (or while trying to improve it).

```
library(reader)
```

```
## Loading required package: NCmisc
```

```
##
## Attaching package: 'reader'
```

```
## The following objects are masked from 'package:NCmisc':
##
##     cat.path, get.ext, rmv.ext
```

```
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(ggplot2)
library(skimr)
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.4.2
```

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.4.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 4.4.2
```

```
nhl <- read.csv("C:/Users/15083/Downloads/nhl_2016_17.csv")
head(nhl)
```

```
##               name born height weight position games_played goals assists
## 1    Spencer Abbott 1988     69    170       LW            1     0       0
## 2 Justin Abdelkader 1987     74    218       LW           64     7      14
## 3      Pontus Aberg 1993     71    196       LW           15     1       1
## 4      Noel Acciari 1991     70    208        C           29     2       3
## 5    Kenny Agostino 1992     72    202       LW            7     1       2
## 6     Sebastian Aho 1997     71    172       RW           82    24      25
##   plus_minus penalty_minutes        salary
## 1          0               0   $575'000.00
## 2        -20              50 $5'500'000.00
## 3         -2               4   $842'500.00
## 4          3              16   $892'500.00
## 5          0               2   $625'000.00
## 6         -1              26   $925'000.00
```

```
dim(nhl)
```

```
## [1] 888  11
```

When working with a dataset, often the first challenge is understanding what each of the rows means. Thus, it is often advised to carefully read the data dictionary. In our case, the columns represent the following:

- `name` : A player's name
- `born` : A player's year of birth
- `height` : A player's height in inches
- `weight` : A player's weight in pounds
- `position` : A player's position
- `games_played` : The number of games a player played during the season
- `goals` : The number of goals scored by the player during the season
- `assists` : The number of assists recorded by the player during the season
- `plus_minus` : The number of times a player was on the ice when his team scored a goal minus the number of times he was on the ice when the opposing team scored a goal at even strength or short-handed
- `penalty_minutes` : The number of penalty minutes the player collected
- `salary` : A player's salary

Now that we understand our variables, we can start with the EDA.

Data Cleaning: Null and Missing Values

One thing to look at are the missing values, which one needs to decide how to handle on a case-by-case basis. This can involve:

- Deleting an entire column: If too many values are missing and the column is not critical
- Deleting rows with one, some, or all variables missing: If only a small pencentage of rows are affected on important variables
- Manually collect the missing data: Tedious and could introduce measurement deviations

- Input the mean, median, or an arbitrary number: Sometimes hard to justify and required expert knowledge

We notice that there are 14 players with missing values for their salary. Since 14/888 rows are not a lot, we will simply drop them.

```
# Look up the missing values for each column
colSums(is.na(nhl)) #let's check for null entries
```

```
##            name            born          height          weight        position
##               0               0               0               0               0
##    games_played           goals         assists      plus_minus penalty_minutes
##               0               0               0               0               0
##          salary
##               0
```

```
string_na_counts <- sapply(nhl, function(col) sum(col == "", na.rm = TRUE))
string_na_counts
```

```
##            name            born          height          weight        position
##               0               0               0               0               0
##    games_played           goals         assists      plus_minus penalty_minutes
##               0               0               0               0               0
##          salary
##              14
```

```
# Remove rows with missing values in the 'salary' column
nhl <- nhl[nhl$salary != "", ]

cat("Remaining number of rows:", nrow(nhl), "\n")
```

```
## Remaining number of rows: 874
```

```
head(nhl)
```

```
##                name born height weight position games_played goals assists
## 1    Spencer Abbott 1988     69    170       LW            1     0       0
## 2 Justin Abdelkader 1987     74    218       LW           64     7      14
## 3      Pontus Aberg 1993     71    196       LW           15     1       1
## 4      Noel Acciari 1991     70    208        C           29     2       3
## 5    Kenny Agostino 1992     72    202       LW            7     1       2
## 6     Sebastian Aho 1997     71    172       RW           82    24      25
##   plus_minus penalty_minutes        salary
## 1          0               0   $575'000.00
## 2        -20              50 $5'500'000.00
## 3         -2               4   $842'500.00
## 4          3              16   $892'500.00
## 5          0               2   $625'000.00
## 6         -1              26   $925'000.00
```

Data Cleaning: Converting the data type of a column (i.e. string to int; categorical to discrete; etc.)

Another thing to look at are the datatypes of the columns and check that they are in accordance with what we expect.

```
sapply(nhl, class)
```

```
##           name           born         height         weight       position
##    "character"      "integer"      "integer"      "integer"    "character"
##   games_played          goals        assists      plus_minus penalty_minutes
##      "integer"      "integer"      "integer"      "integer"      "integer"
##         salary
##    "character"
```

```
# Replace ' and $ stray characters in salary to get into correct data form
nhl$salary <- nhl$salary %>%
  gsub("\\$", "", .) %>%
  gsub("'", "", .) %>%
  as.numeric()

head(nhl)
```

```
##               name born height weight position games_played goals assists
## 1    Spencer Abbott 1988     69    170       LW            1     0       0
## 2 Justin Abdelkader 1987     74    218       LW           64     7      14
## 3      Pontus Aberg 1993     71    196       LW           15     1       1
## 4      Noel Acciari 1991     70    208        C           29     2       3
## 5    Kenny Agostino 1992     72    202       LW            7     1       2
## 6     Sebastian Aho 1997     71    172       RW           82    24      25
##   plus_minus penalty_minutes  salary
## 1          0               0  575000
## 2        -20              50 5500000
## 3         -2               4  842500
## 4          3              16  892500
## 5          0               2  625000
## 6         -1              26  925000
```

Data Exploration: Summary Statistics and Data Visualization

We can then turn our attention to the numerical variables. With `skim(df)`, we can easily get a lot of summary statistics for them. This is a way to get a good sense for the data and find potential outliers. However, plotting might be just as good to do this, and skim allows you to see the distribution of variables.

Note that we will use histograms in this example, which are usually the standard (especially when it comes to looking at the general distribution and spotting outliers), but scatterplots, boxplots, etc. are also powerful visualizations.

```
skim(nhl)
```

Data summary

Name	nhl
Number of rows	874
Number of columns	11
_____	
Column type frequency:	
character	2
numeric	9
_____	
Group variables	None

Variable type: character

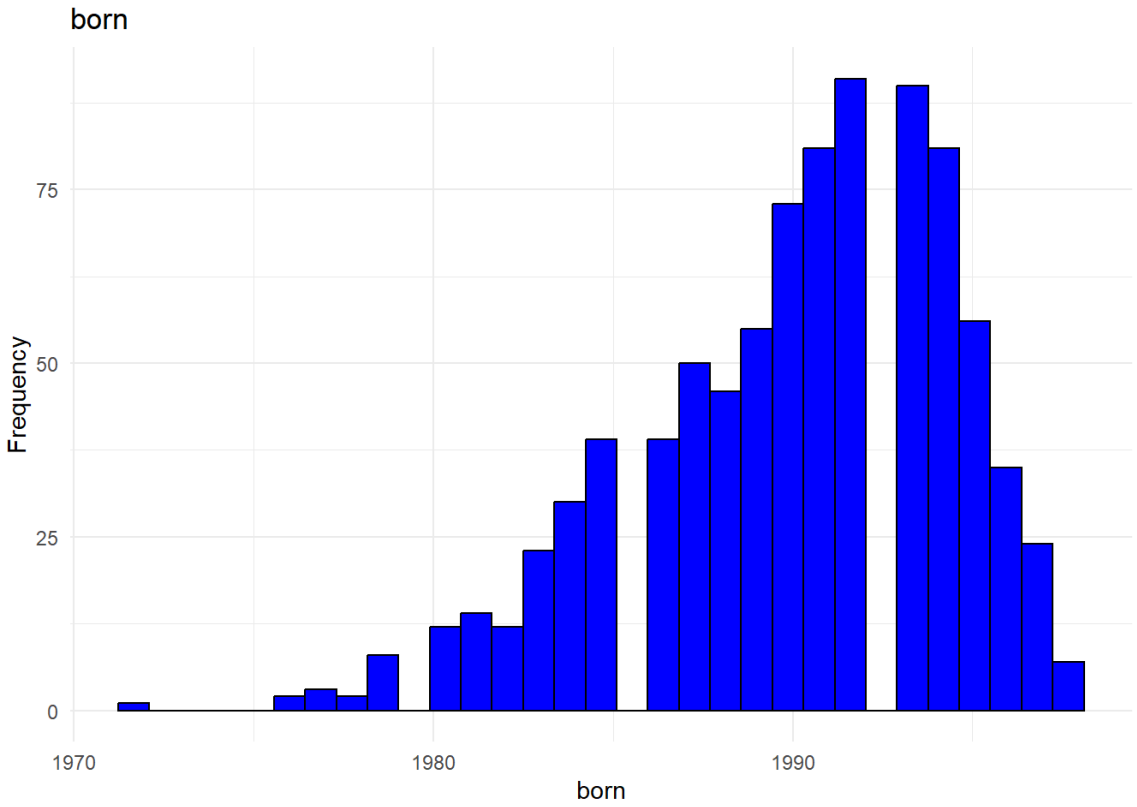
skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
name	0	1	8	24	0	874	0
position	0	1	1	2	0	4	0

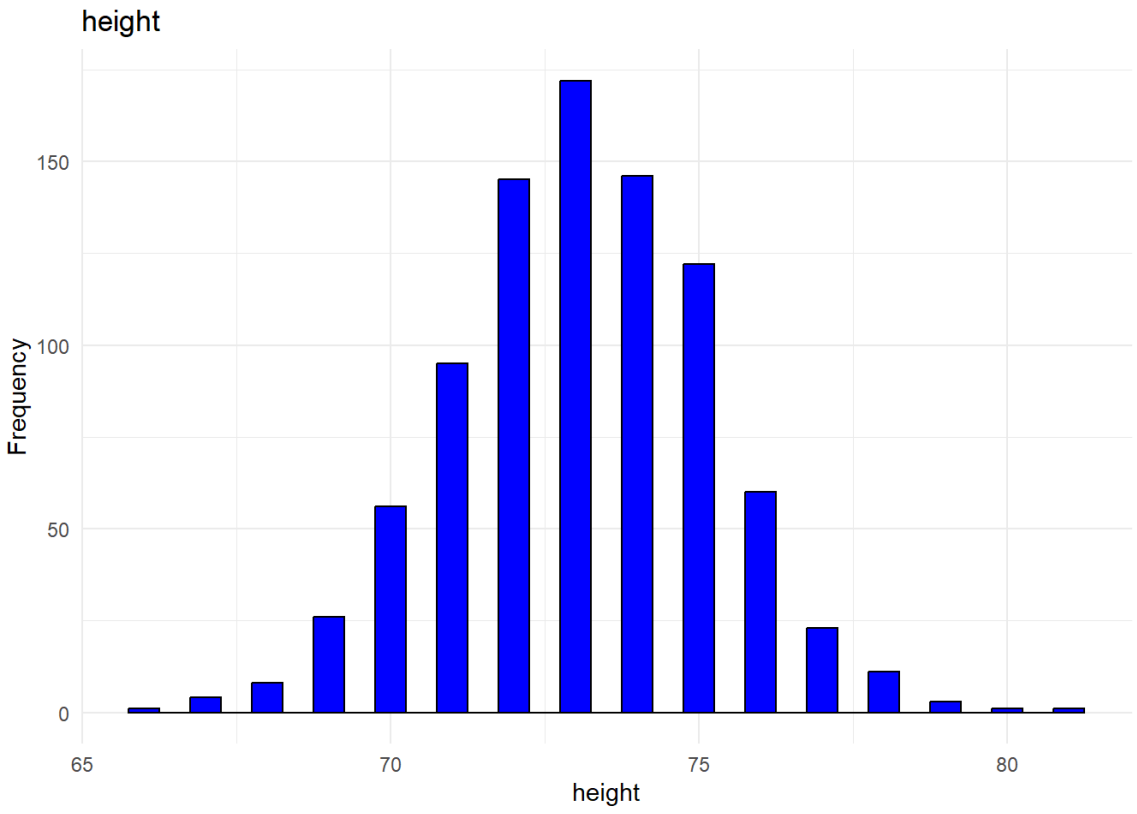
Variable type: numeric

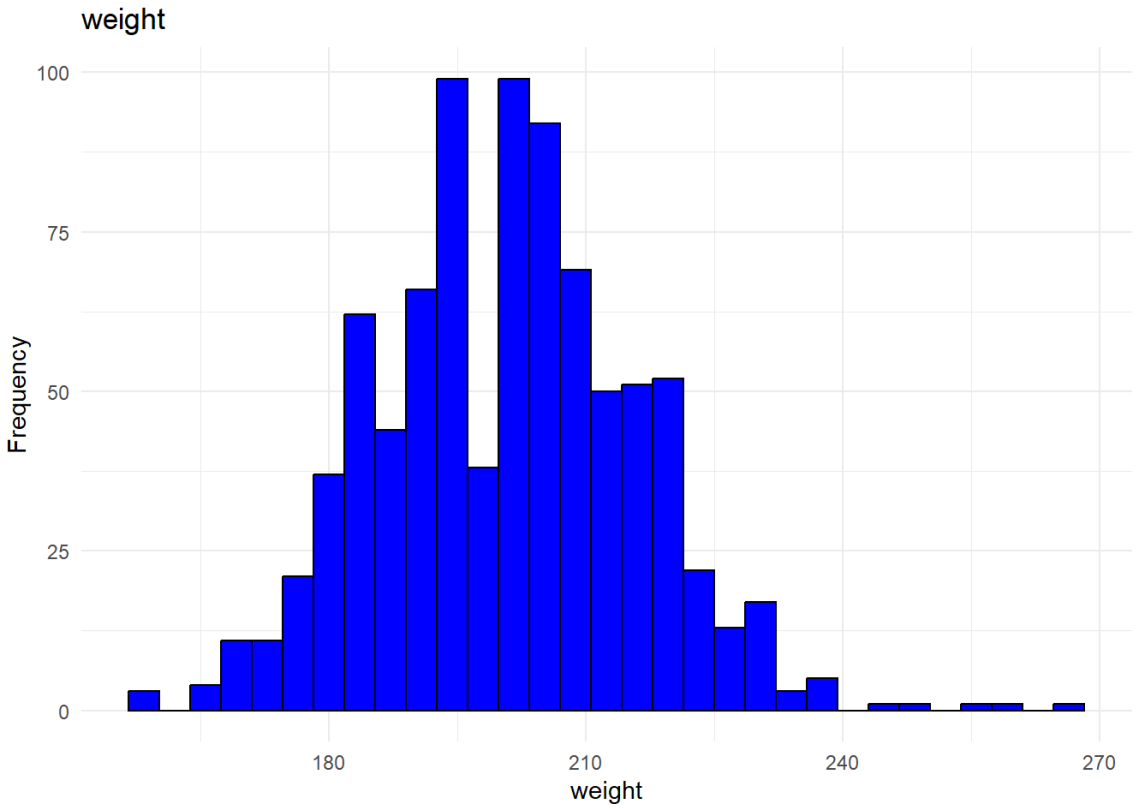
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
born	0	1	1990.05	4.40	1972	1987	1991	1993	1998	▁▁▃▇▇
height	0	1	73.08	2.11	66	72	73	75	81	▁▂▇▃▁
weight	0	1	200.84	15.06	157	190	200	210	265	▁▃▇▂▁
games_played	0	1	50.43	29.31	1	21	61	78	82	▆▁▁▁▇
goals	0	1	7.66	8.63	0	1	5	12	44	▇▃▂▁▁
assists	0	1	12.84	12.78	0	2	9	20	70	▇▃▂▁▁
plus_minus	0	1	-0.38	9.95	-34	-5	-1	4	34	▁▂▇▂▁
penalty_minutes	0	1	25.03	23.91	0	6	20	35	154	▇▃▂▁▁
salary	0	1	2325288.71	2298252.56	575000	742500	925000	3700000	14000000	▇▃▂▁▁

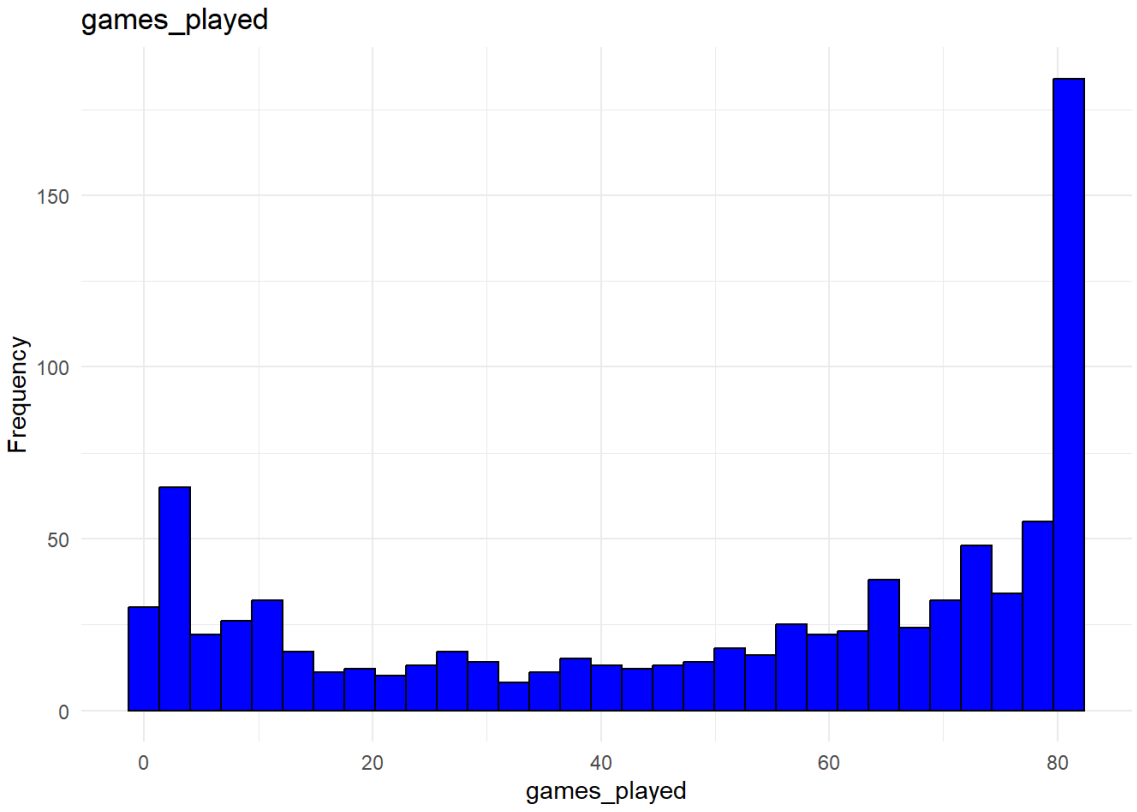
```
numeric_columns <- names(nhl)[sapply(nhl, is.numeric)]

# Loop through numeric columns and create histograms
for (col in numeric_columns) {
  # Compute an appropriate binwidth dynamically
  data_range <- range(nhl[[col]], na.rm = TRUE)
  binwidth <- (data_range[2] - data_range[1]) / 30  # Aim for ~30 bins

  plot <- ggplot(nhl, aes_string(x = col)) +
    geom_histogram(
      binwidth = binwidth,
      fill = "blue",
      color = "black"
    ) +
    ggtitle(col) +
    xlab(col) +
    ylab("Frequency") +
    theme_minimal()

  print(plot)  # Print the plot inside the loop
}
```

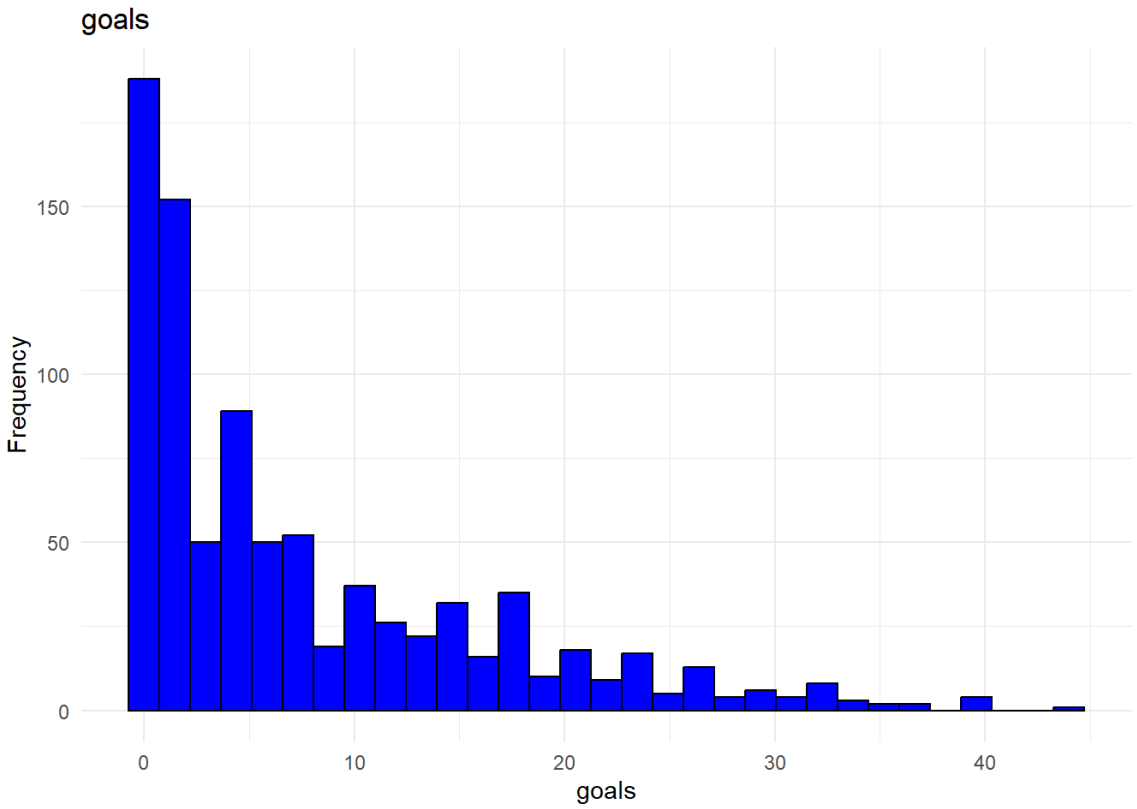
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

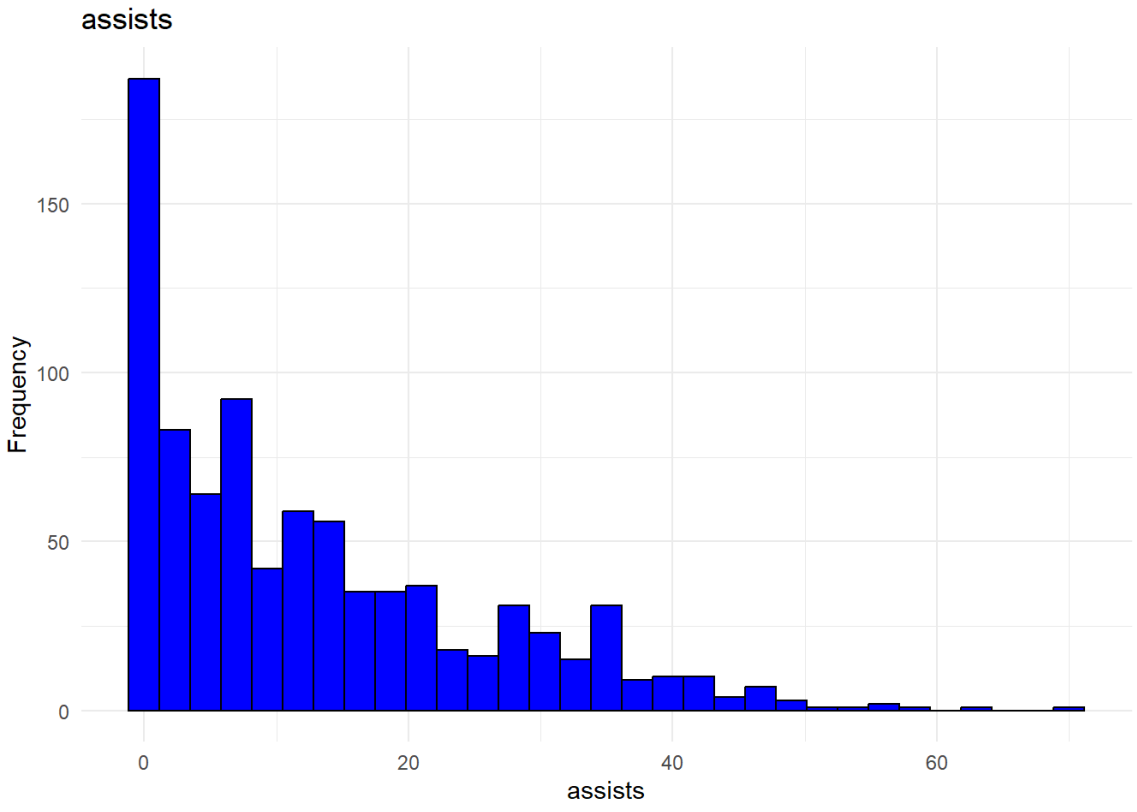

born
Frequency
0
25
50
75
1970
1980
1990
born

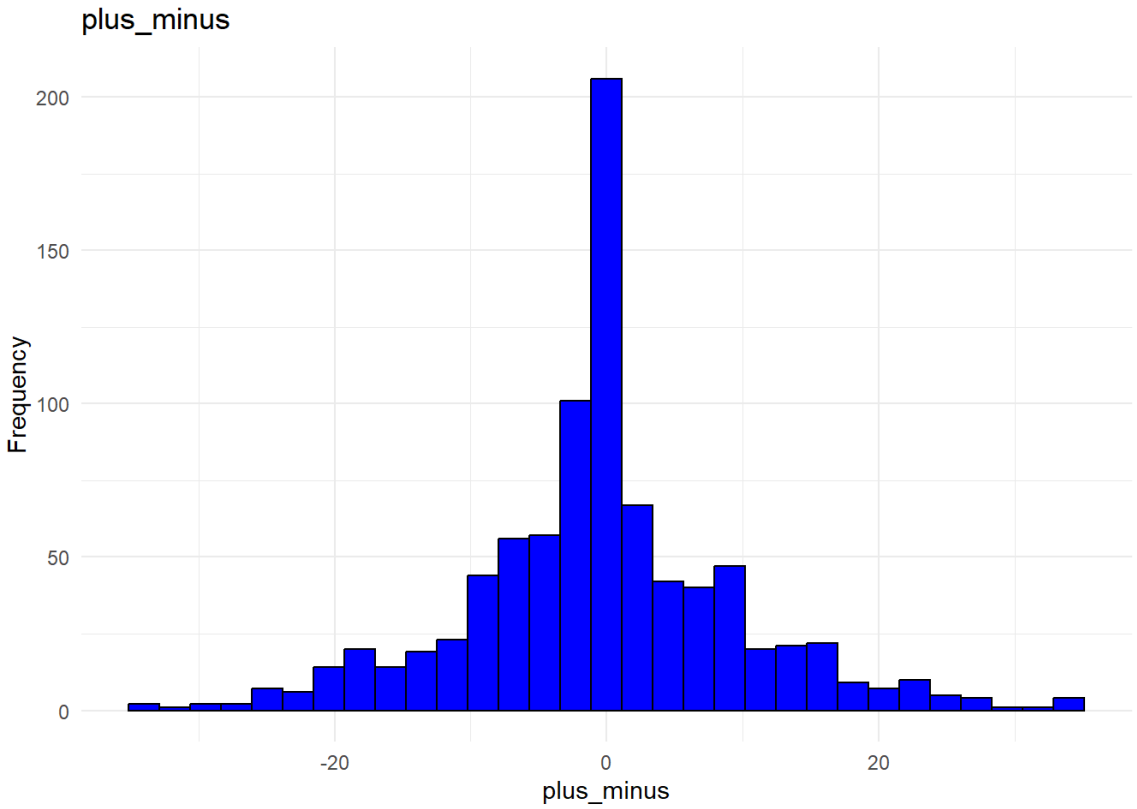

height
Frequency
0
50
100
150
65
70
75
80
height

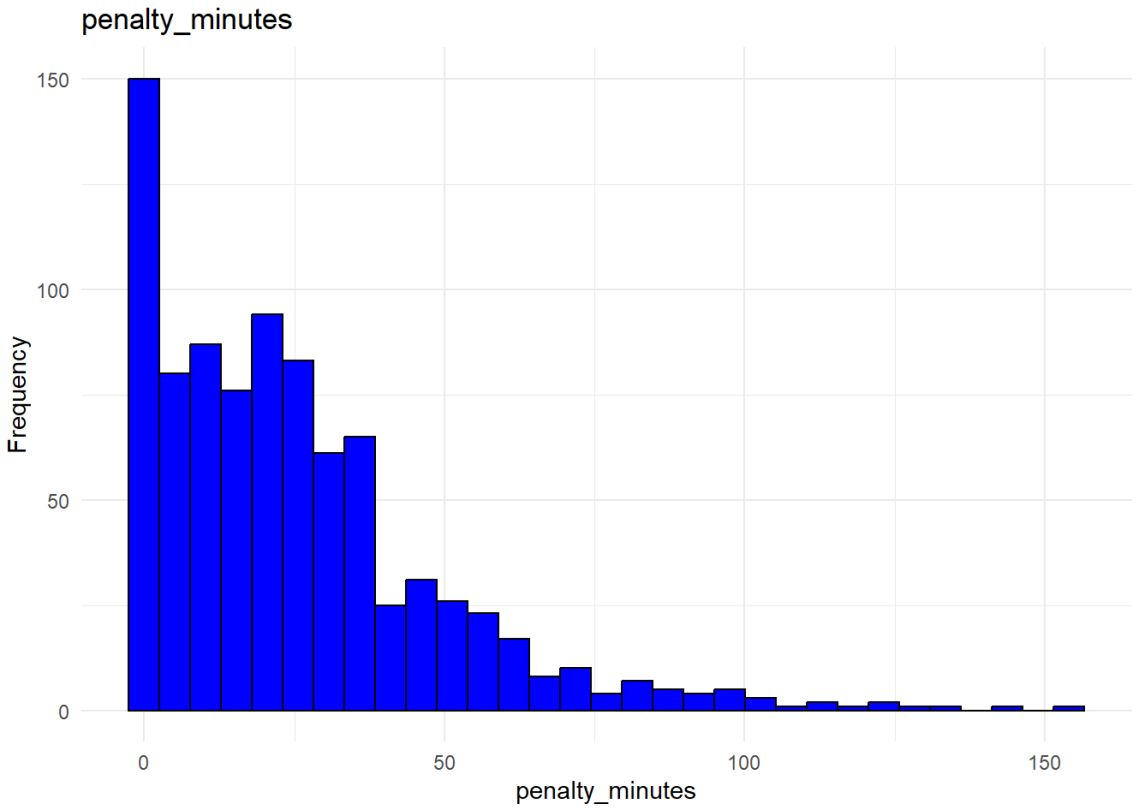

weight
Frequency
100
75
50
25
0
180
210
240
270
weight

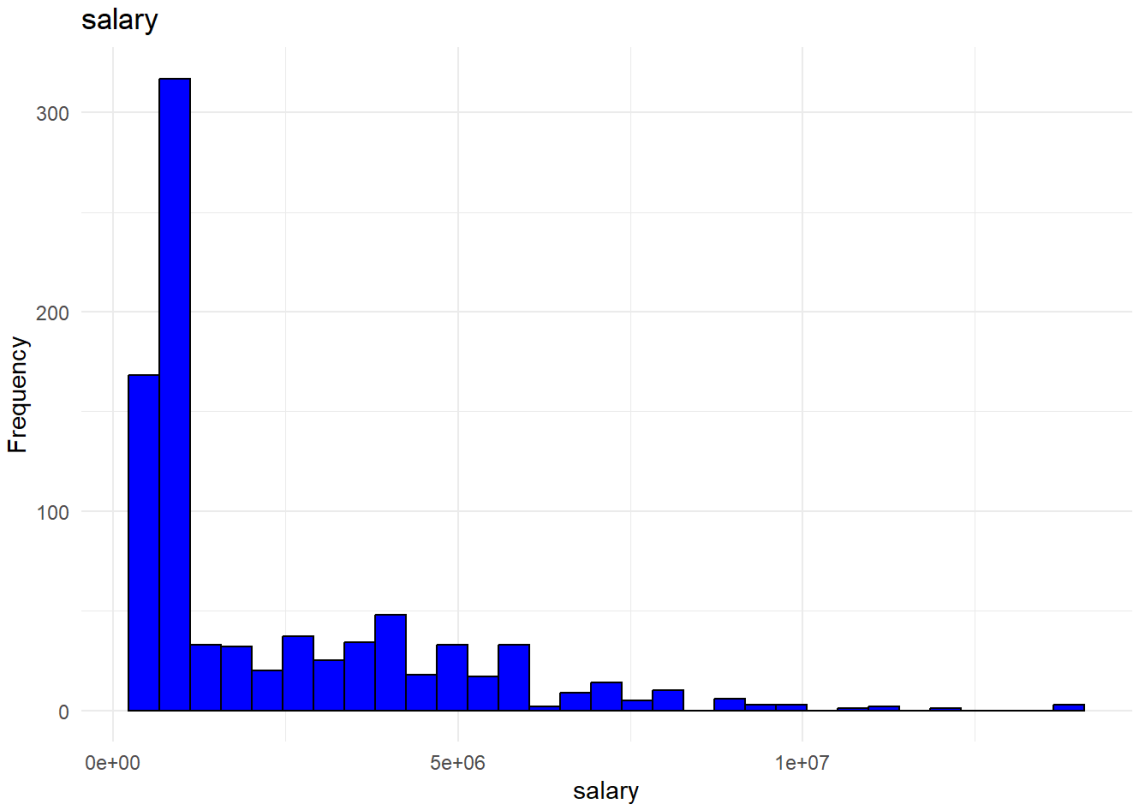

games_played
Frequency
150
100
50
0
0
20
40
60
80
games_played


goals
Frequency
150
100
50
0
0
10
20
30
40
goals


assists
Frequency
150
100
50
0
0
20
40
60
assists

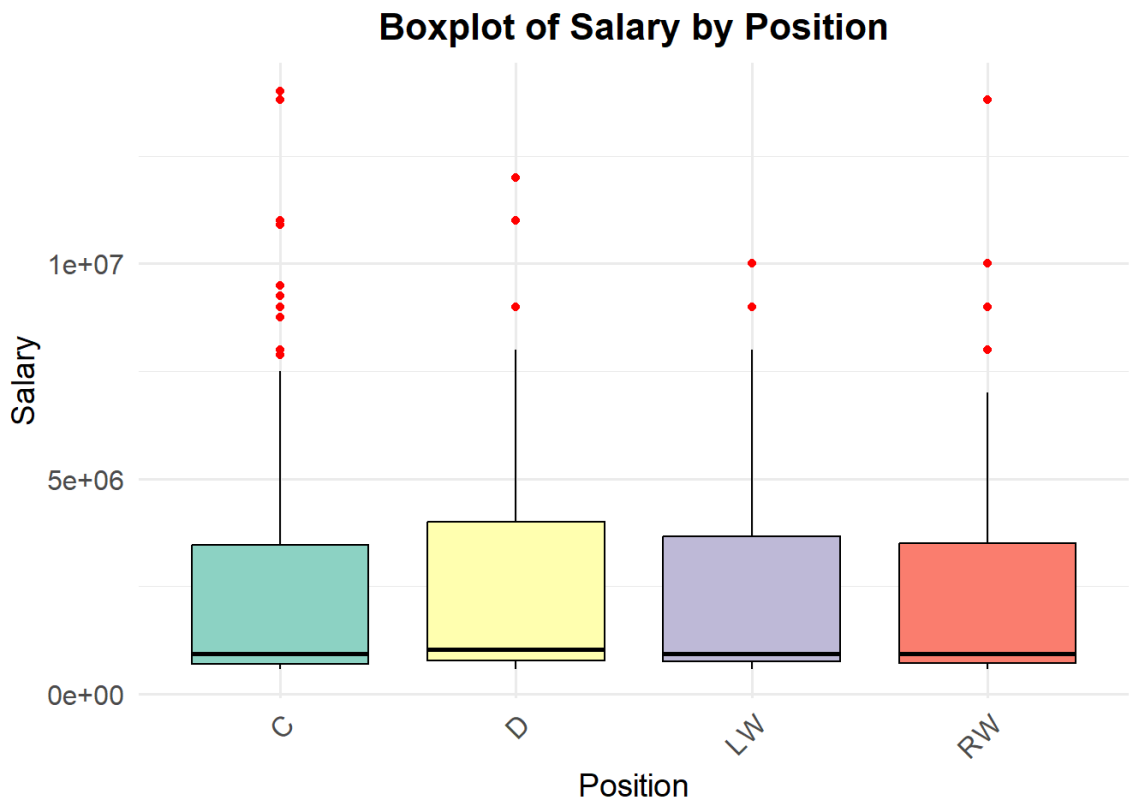

plus_minus
200
150
100
50
0
Frequency
-20
0
20
plus_minus


penalty_minutes
150
100
50
0
Frequency
0
50
100
150
penalty_minutes

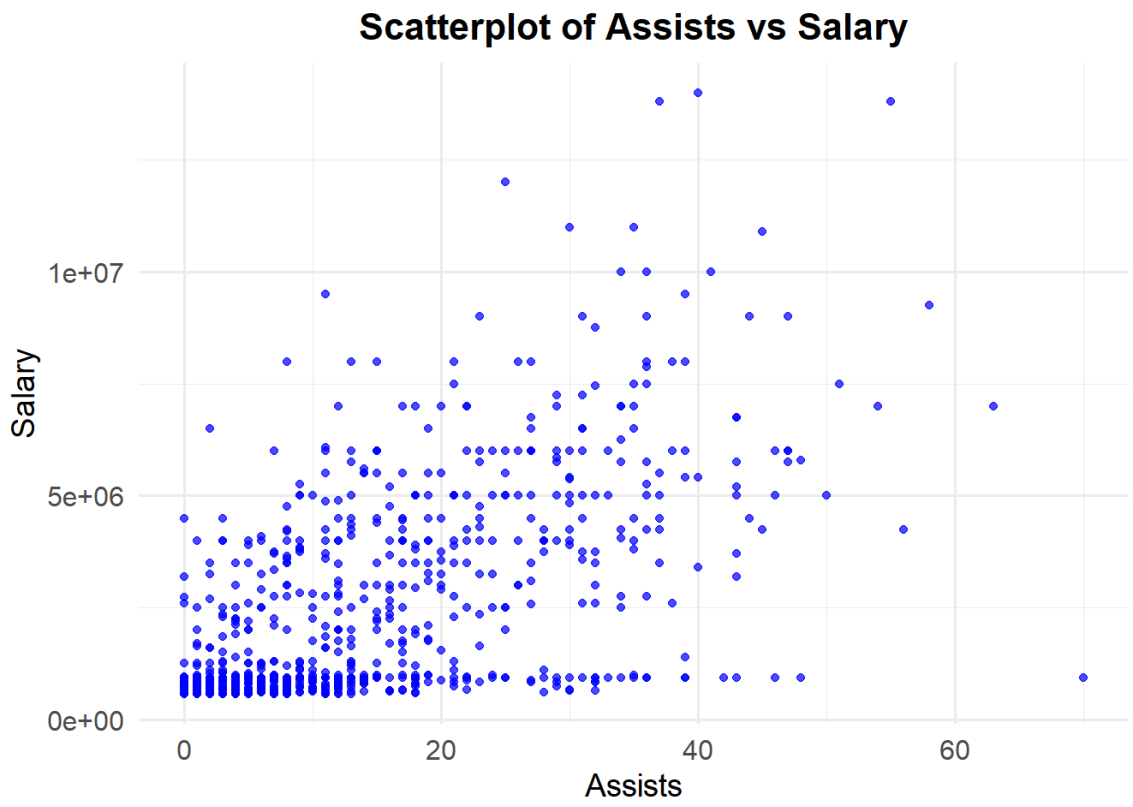


As mentioned above, other types of visualizations are also very powerful for EDA. ggplot2 can help with boxplots, scatterplots, and pairplots with examples below.

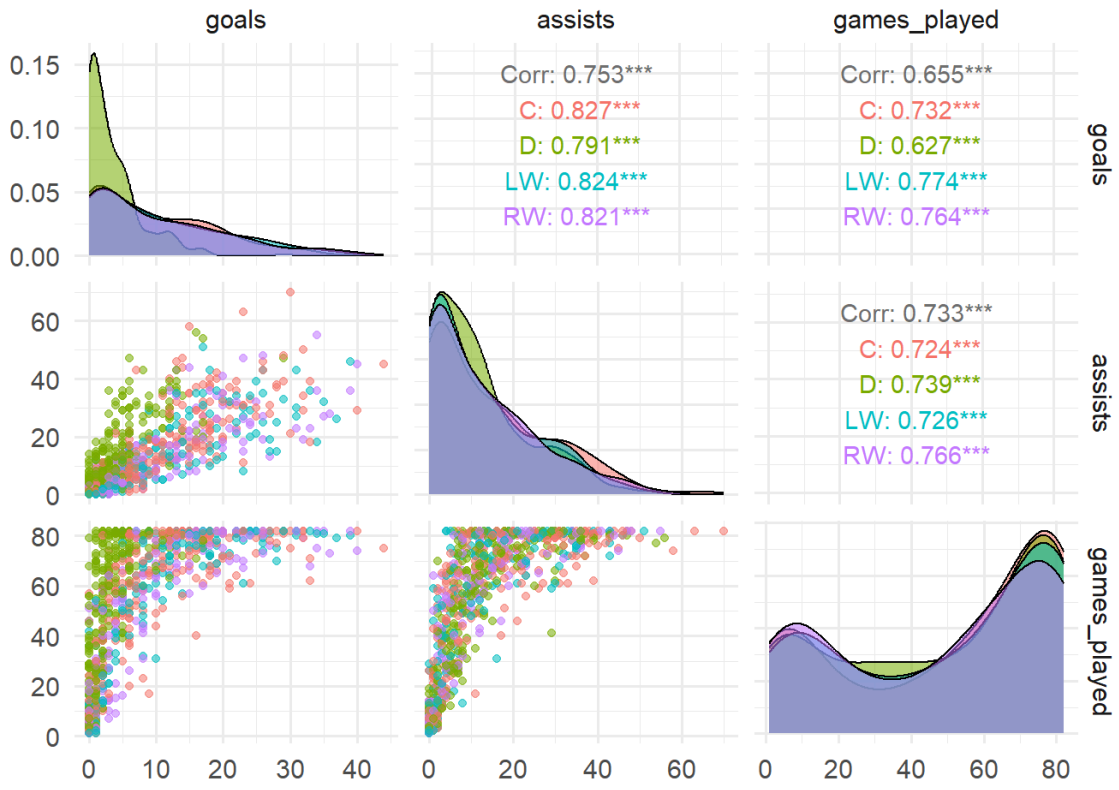
```
ggplot(nhl, aes(x = position, y = salary, fill = position)) +
  geom_boxplot(color = "black", outlier.color = "red", outlier.shape = 16) +
  ggtitle("Boxplot of Salary by Position") +
  xlab("Position") +
  ylab("Salary") +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    axis.text.x = element_text(size = 12, angle = 45, hjust = 1),
    axis.text.y = element_text(size = 12),
    legend.position = "none"
  ) +
  scale_fill_brewer(palette = "Set3")
```



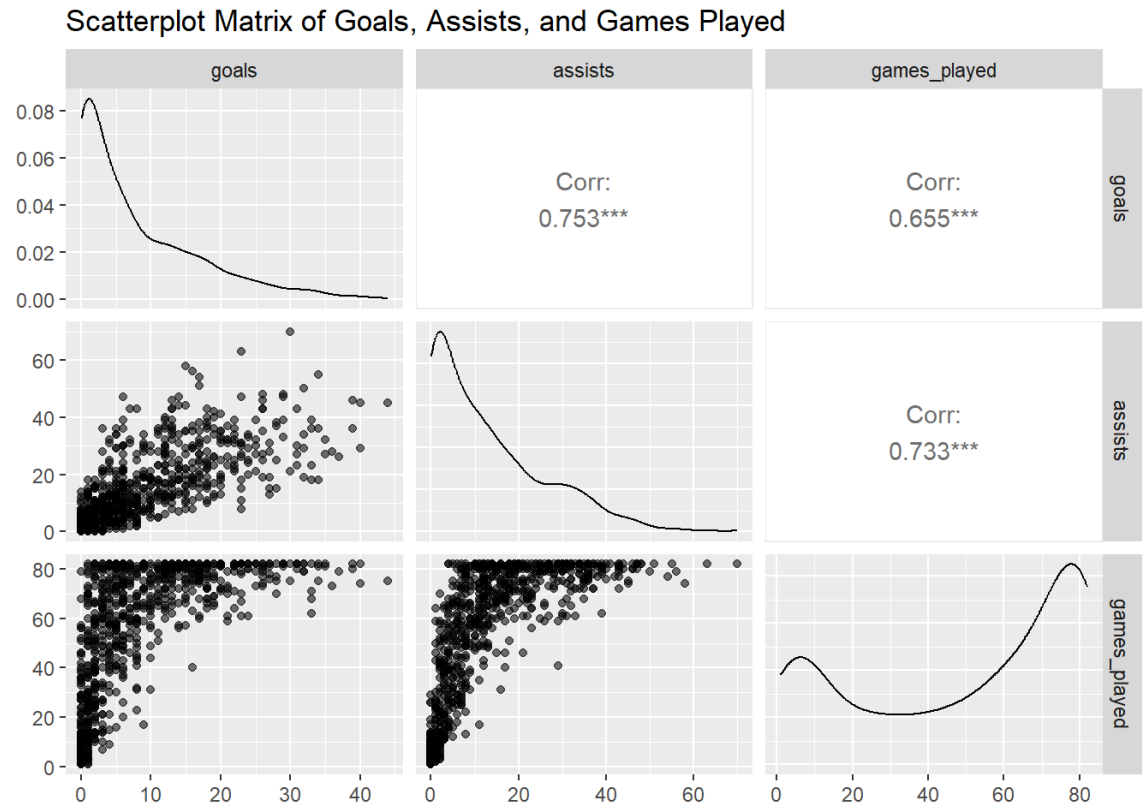

```
ggplot(nhl, aes(x = assists, y = salary)) +
  geom_point(color = "blue", alpha = 0.7) +
  ggtitle("Scatterplot of Assists vs Salary") +
  xlab("Assists") +
  ylab("Salary") +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
    axis.text = element_text(size = 12),
    axis.title = element_text(size = 14)
  )
```




```
library(GGally)
#GGally is a helpful extension of ggplot primarily for the ggpairs function to stack scatterplots here, https://gg
obi.github.io/ggally/
ggpairs(
  nhl,
  columns = c("goals", "assists", "games_played"),  # Variables to include
  aes(color = position, alpha = 0.7)               # Group by 'position' with transparency
) +
  theme_minimal(base_size = 14)
```

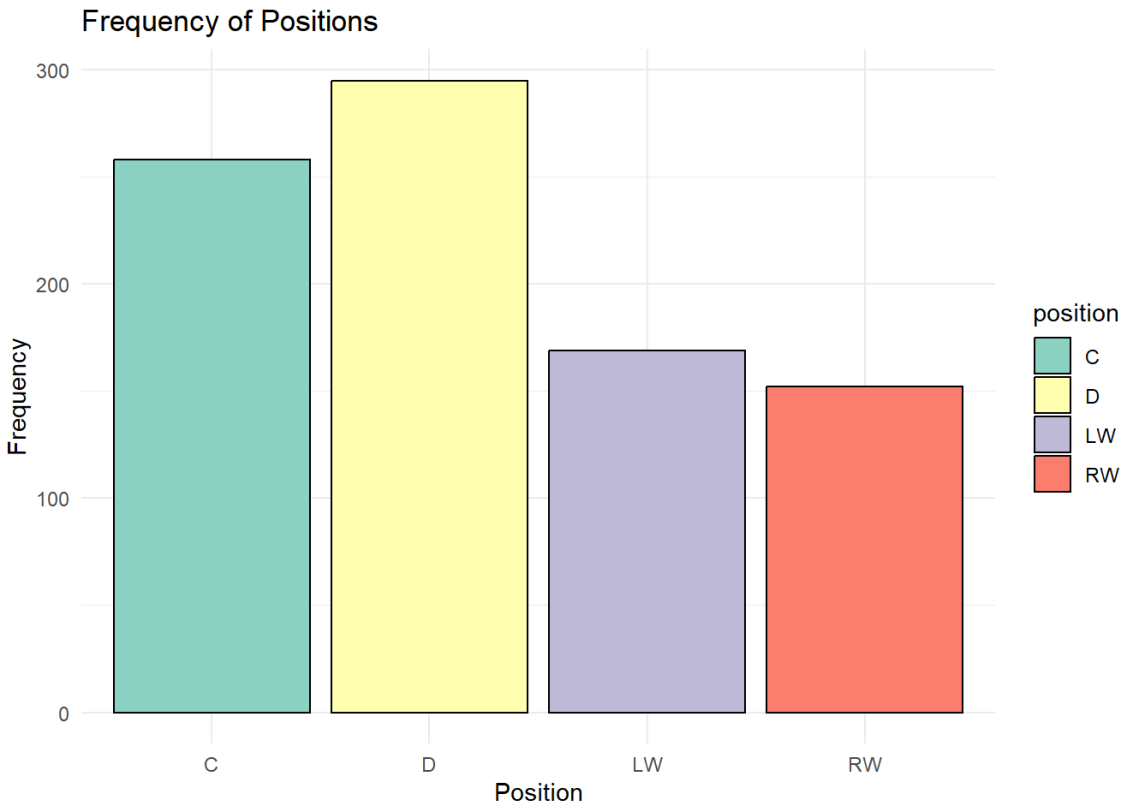



```
#pairwise scatterplots
ggpairs(
  nhl,
  columns = c("goals", "assists", "games_played"),
  mapping = aes(alpha = 0.2),  # Transparency for points
  title = "Scatterplot Matrix of Goals, Assists, and Games Played"
)
```



Note: GGally and ggplot2 are designed to work togeather and have similar syntax.

```
#plot frequencies of categorical data, position

ggplot(nhl, aes(x = position, fill = position)) +
  geom_bar(color = "black") +
  ggtitle("Frequency of Positions") +
  xlab("Position") +
  ylab("Frequency") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set3")
```



Data Cleaning: Converting Categorical Variables to Dummies

It is often necessary to convert the categorical variables to numerical values so that the model can interpret them. In other words, some packages can only perform analyses on numbers, so you might need to convert string categories with numerical representations.

Today we will make indicator variables of the `position` column.

```
nhl_dummies <- as.data.frame(model.matrix(~ position - 1, data = nhl))
colnames(nhl_dummies) <- sub("^position", "position_", colnames(nhl_dummies))
nhl <- cbind(nhl, nhl_dummies)

head(nhl)
```

```
##               name born height weight position games_played goals assists
## 1    Spencer Abbott 1988     69    170       LW            1     0       0
## 2 Justin Abdelkader 1987     74    218       LW           64     7      14
## 3      Pontus Aberg 1993     71    196       LW           15     1       1
## 4      Noel Acciari 1991     70    208        C           29     2       3
## 5    Kenny Agostino 1992     72    202       LW            7     1       2
## 6     Sebastian Aho 1997     71    172       RW           82    24      25
##   plus_minus penalty_minutes  salary position_C position_D position_LW
## 1          0               0  575000          0          0           1
## 2        -20              50 5500000          0          0           1
## 3         -2               4  842500          0          0           1
## 4          3              16  892500          1          0           0
## 5          0               2  625000          0          0           1
## 6         -1              26  925000          0          0           0
##   position_RW
## 1           0
## 2           0
## 3           0
## 4           0
## 5           0
## 6           1
```

It is important to note that when we generate indicaor variables we have to watch out for multicollinearity, which is why we use the drop_first=True argument (this removes one of the outcomes to avoid multicollinearity). We will learn more about this in future labs and assignments, as well as get more practice creating indicator variables and performing one-hot encoding in future weeks.

Data Cleaning: Removing Duplicate Rows

Another thing you should always check is that you don't have any duplicate rows. For example, in our hockey dataset, each row should represent a unique hockey player. We can check for duplicate rows by ensuring none of the names of the `name` column occur more than once.

```
#many ways to check for uniqueness

#make a table of the most repeated
most_occurrences <- max(table(nhl$name))
cat("Most occurrences of the same name:", most_occurrences, "\n")
```

```
## Most occurrences of the same name: 1
```

```
#check uniqueness of the name entry
is_unique <- nrow(nhl) == length(unique(nhl$name))
cat("Are all names unique?", is_unique, "\n")
```

```
## Are all names unique? TRUE
```

```
nhl <- unique(nhl)
# or in dplyr nhl <- nhl %>% distinct()
```

Step 2 | Create/fit the regression

StatsModels

Running regressions is likely a very familiar concept for everyone. This can easily be done in python with the `StatsModels` library.

Let's imagine that we want to predict `salary` with `goals` and a constant: $salary_i = \beta_0 + \beta_1 \cdot goals_i + \varepsilon_i$
We want to view the output which displays the statistical metrics of the model.

```
regression1 <- lm(salary ~ goals, data = nhl)
summary(regression1)
```

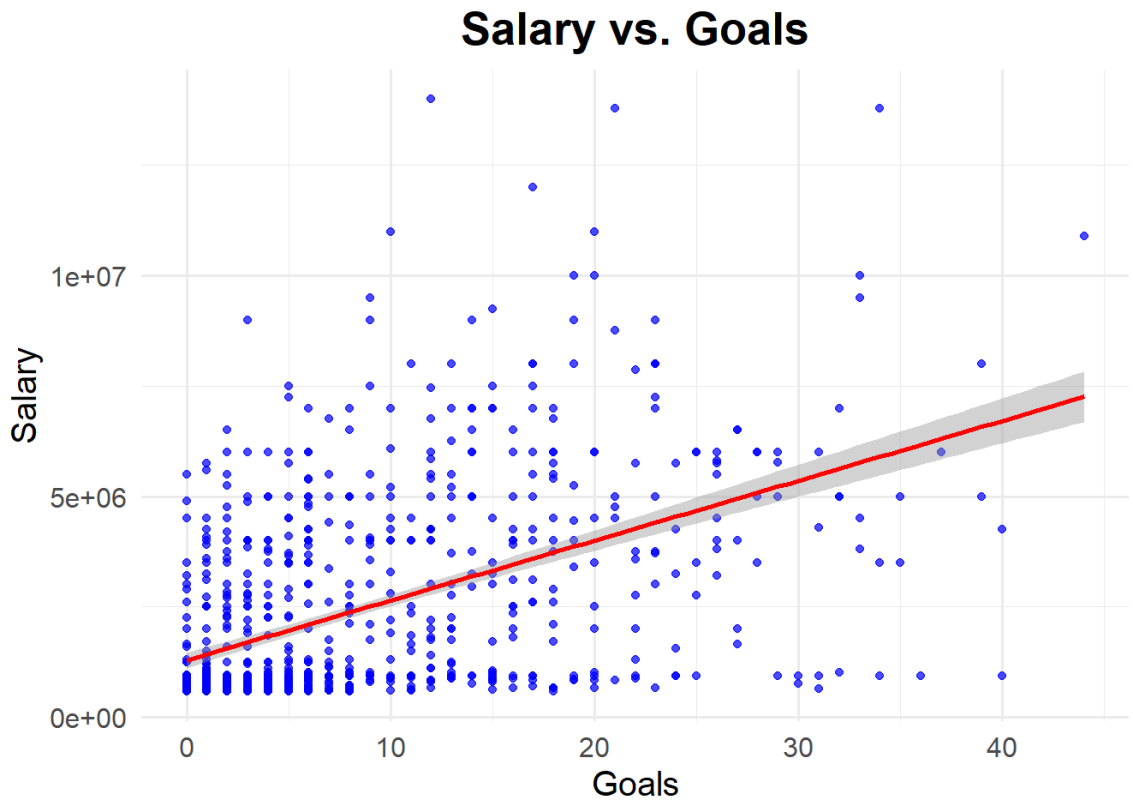
```
##
## Call:
## lm(formula = salary ~ goals, data = nhl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5788053  -986532  -609530   899633 11085630
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1286363      89484   14.38   <2e-16 ***
## goals         135667       7757   17.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1979000 on 872 degrees of freedom
## Multiple R-squared:  0.2597, Adjusted R-squared:  0.2589
## F-statistic: 305.9 on 1 and 872 DF,  p-value: < 2.2e-16
```

According to our dataset, we can interpret our parameters. The intercept tells us that a player who scores no goals has an expected salary of $1,286,363. We can interpret the coefficient on goals as meaning each additional goal is associated with a $135,667 increase in predicted salary.

We can easily visualize this regression as well.

```
ggplot(nhl, aes(x = goals, y = salary)) +
  geom_point(color = "blue", alpha = 0.7) +  # Scatterplot points
  geom_smooth(method = "lm", color = "red", se = TRUE) +  # Regression line with confidence interval
  ggtitle("Salary vs. Goals") +
  xlab("Goals") +
  ylab("Salary") +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 20, face = "bold"),
    axis.title = element_text(size = 15),
    axis.text = element_text(size = 12)
  )
```

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



Adding more variables is also quite straightforward. For instance, if we were to run the following regression:
$salary_i = \beta_0 + \beta_1 \cdot goals_i + \beta_2 \cdot games\_played_i + \beta_3 \cdot position\_RW_i + \varepsilon_i$

```
regression2 <- lm(salary ~ goals + games_played + position_RW, data = nhl)

summary(regression2)
```

```
##
## Call:
## lm(formula = salary ~ goals + games_played + position_RW, data = nhl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5030520 -1250108  -170702   672124 10704842
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    622547     136114   4.574 5.49e-06 ***
## goals           90541      10125   8.942  < 2e-16 ***
## games_played    20870       2963   7.043 3.81e-12 ***
## position_RW   -247216     174244  -1.419    0.156
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1921000 on 870 degrees of freedom
## Multiple R-squared:  0.304,  Adjusted R-squared:  0.3016
## F-statistic: 126.7 on 3 and 870 DF,  p-value: < 2.2e-16
```

Step 3 | Evaluate the Model's Statistical Metrics

To evaluate our regression models we commonly use MSE and R^2. We can extract R^2 from the regression output generated in base R as well as in `stats`, which includes logistic regression and glm.

As an example, let's create a regression using goals, games plyed, penalty minutes, and our position dummies are our predictors.

```
regression3 <- lm(salary ~ goals + games_played + penalty_minutes + position_D + position_LW + position_RW, data =
nhl)
summary(regression3)
```

```
##
## Call:
## lm(formula = salary ~ goals + games_played + penalty_minutes +
##     position_D + position_LW + position_RW, data = nhl)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5343203 -1162055  -225140   645509 11025639
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       382630     163440   2.341   0.0195 *
## goals             116080      11022  10.532  < 2e-16 ***
## games_played       14614       3516   4.157 3.55e-05 ***
## penalty_minutes     3148       3349   0.940   0.3476
## position_D        790926     173785   4.551 6.10e-06 ***
## position_LW      -119596     188164  -0.636   0.5252
## position_RW       -33290     194347  -0.171   0.8640
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1892000 on 867 degrees of freedom
## Multiple R-squared:  0.3271, Adjusted R-squared:  0.3224
## F-statistic: 70.23 on 6 and 867 DF,  p-value: < 2.2e-16
```

```
#Print R^2
regression3_summary <- summary(regression3)
rsquared <- regression3_summary$r.squared
cat("R^2:", rsquared, "\n")
```

```
## R^2: 0.3270534
```

```
#generate predicted Y to get MSE
pred_y <- predict(regression3)

y <- nhl$salary

#Calc MSE
mse <- mean((y - pred_y)^2)
cat("The MSE of our model is:", mse, "\n")
```

```
## The MSE of our model is: 3.550413e+12
```

Typically, you would experiment with different regressions and evaluate each model's performance to determine which might be the best model (i.e. minimum MSE or greatest R^2) out of these options.

Step 4 | Applying our Regression Model

Finally, after selecting a model, we can infer what a "simulated" player's salary might be.

Let's say we would like to predict what a right winger would expect to make if he scored 12 goals in 55 games and had 6 penalty minutes. Following from our knowledge of regressions, we simply "plug in" these values manually for our predictors to calculate the predicted salary of this type of player.

```
player_data <- data.frame(
  goals = 12,
  games_played = 55,
  penalty_minutes = 6,
  position_D = 0,
  position_LW = 0,
  position_RW = 1
)

# view player data
print(player_data)
```

```
##   goals games_played penalty_minutes position_D position_LW position_RW
## 1    12           55               6          0           0           1
```

```
#predict salary
predicted_salary <- scales::comma(predict(regression3, newdata = player_data), accuracy = 0.01)

cat("We would expect a right winger player with 12 goals in 55 games and 6 penalty minutes to have a salary of", p
redicted_salary, "\n")
```

```
## We would expect a right winger player with 12 goals in 55 games and 6 penalty minutes to have a salary of 2,56
4,931.35
```