UFC Project

Jared De La Serna

March 12, 2024

Introduction:

UFC (Ultimate Fighting Championship) is a mixed martial arts (MMA) organization where athletes must be well-rounded in various martial arts disciplines. Competitors in the UFC come from various backgrounds such as Brazilian jiu-jitsu, boxing, wrestling, judo, karate, Muay Thai, and more. Being the best in the UFC often requires proficiency in both striking and grappling techniques, as well as excellent conditioning, perfect timing, and mental strength. This diverse skill set allows fighters to handle the wide range of combat scenarios they encounter in the octagon. UFC Championship is divided into 8 weight divisions, weight is the most crucial in the sport, you have to make a certain weight a day before the fight is one of the most important day before the fight, failure to meet the weight requirements you can be fine, titled will vacant, or the fight can be canceled.

We will explore this UFC dataset that has statistics of all the UFC fighters. I got the dataset from Kaggle, which data of 2,479 fighters, and it had statistics from every aspect of the sport.

The variables are: 1) SLpM - Significant Strikes Landed per Minute

2) Str. Acc. - Significant Striking Accuracy
3) SApM - Significant Strikes Absorbed per Minute
4) Str. Def. - Significant Strike Defence (the % of opponents strikes that did not land)
5) TD Avg. - Average Takedowns Landed per 15 minutes
6) TD Acc. - Takedown Accuracy
7) TD Def. - Takedown Defense (the % of opponents TD attempts that did not land)
8) Sub. Avg. - Average Submissions Attempted per 15 minutes

Method Description

Imputation For Missing Values

Research Question

How can we input values from fights that happened but the statistics of the fight were never recorded for reasons such as; the fight not being televised, UFC not posting the actual statistics for the fight, or data collection issues?

Descirption of the Method:

A technique used to handle missing data. The key idea is to create several different plausible imputations for the missing values, rather than filling them with a single estimate like the mean or median. This has

many purposes, it could be that you can either ignore the NA's valuesi in your dataset, or you can imput them based on predictive factors.

Motivation of Method:

Within the UFC there's been a long history of unrecorded fights within the entire history of the organization. This makes it difficult to track and understand how the sport has been involved over time. I really want to see what would happen if the UFC would release the true statistics of the fighter, because most of them are not televised or not enough importance in order to publish their statistics. Which I think is unfair to have the them as N/A.

Small Example

Since the imputation for missing values is complicated, I'll demonstrate the power of the method with a smaller dataset. The example was from "Imputation For Missing Values (HSUAR Chapter 16)" by Torsten Hothorn and Brian S. Everitt.

We create the dataset manually by imputing and creating all the variables. We will show the dataset. This dataset records lowest temperatures (in Fahrenheit) recorded in various months for cities in the US. Warning: there's some NA's.

```
temperature_data <- data.frame(
  City = c("Atlanta", "Baltimore", "Bismarck", "Boston", "Chicago", "Dallas", "Denver", "El Paso",
           "Honolulu", "Houston", "Juneau", "Los Angeles", "Miami", "Nashville", "New York",
           "Omaha", "Phoenix", "Portland", "Reno", "San Francisco", "Seattle", "Washington"),
  January = c(-8, -7, -44, -12, -27, 4, -25, -8, 53, 12, -22, 23, 30, -17, -6, -23, NA, -26, -16, 24, NA
  April = c(26, 20, -12, 16, 7, NA, -2, 23, 57, 31, 6, 39, 46, 23, 12, 5, 32, 8, NA, 31, 29, 24),
  July = c(53, NA, 35, 54, 40, 59, 43, 57, 67, 62, 36, 49, 69, 51, 52, 44, 61, 40, 33, 43, 43, 55),
  October = c(28, 25, 5, 28, 17, 29, 3, NA, NA, 33, 11, NA, 51, 31, 14, 13, 34, 15, 8, NA, 28, 29)
)

print(temperature_data)
```

```
##             City January April July October
## 1        Atlanta      -8    26   53      28
## 2      Baltimore      -7    20   NA      25
## 3       Bismarck     -44   -12   35       5
## 4         Boston     -12    16   54      28
## 5        Chicago     -27     7   40      17
## 6         Dallas       4    NA   59      29
## 7         Denver     -25    -2   43       3
## 8        El Paso      -8    23   57      NA
## 9       Honolulu      53    57   67      NA
## 10       Houston      12    31   62      33
## 11        Juneau     -22     6   36      11
## 12   Los Angeles      23    39   49      NA
## 13         Miami      30    46   69      51
## 14     Nashville     -17    23   51      31
## 15      New York      -6    12   52      14
## 16         Omaha     -23     5   44      13
## 17       Phoenix      NA    32   61      34
## 18      Portland     -26     8   40      15
## 19          Reno     -16    NA   33       8
## 20 San Francisco      24    31   43      NA
```

```
## 21       Seattle      NA    29   43      28
## 22    Washington      -5    24   55      29
```

We can see our dataset has some NA that might hurt the the credibility of dataset. In reality we have two options, either get rid of the NA's or we can use Imputation For Missing Values in order to replace our NA's with a value. You can either do both, but for the power of learning we will impute values for the missing values.

First step, we try to identify on our NA's in our dataset. This could be done with using the "sapply" function and doing the a sum() in order to count all the NA's.

```
sapply(temperature_data, function(x) sum(is.na(x)))
```

```
##    City January   April    July October
##       0       2       2       1       4
```

We see have multiple missing values in the months. Now we will try to imput values for October since it's the month that has the most missing values. Note you can do this for any variables.

First we create a dataset called "imp" which will contain all variables; City, January, April, July ,and October. The missing values in October will be replaced by the mean temperatures in October of the complete cases. We will talk about the package and library "mice" later on.

```
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
imp <- mice(temperature_data, method = "mean", m = 1, maxit = 1)
```

```
##
##  iter imp variable
##   1   1  January  April  July  October
```

```
## Warning: Number of logged events: 1
```

Now we can to compare our summary statistics of the IMP and our unedited dataset.

```
summary(temperature_data$October)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    3.00   13.25   26.50   22.33   29.00   51.00       4
```

```
with(imp, summary(October))
```

```
## call :
## with.mids(data = imp, expr = summary(October))
##
## call1 :
## mice(data = temperature_data, m = 1, method = "mean", maxit = 1)
##
## nmis :
## [1] 0 2 2 1 4
##
```

```
## analyses :
## [[1]]
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   14.25   22.33   22.33   28.75   51.00
```

We can see the that the values of the 1st Quartile, median, and 3rd Quartile has changed, but the mean has remained unchanged. Another useful fact is that we get to see how many missing values are in the dataset. Also usually, the mean won't change because the missing values that we had we replaced by the true sample mean. Which some cases when it comes to data cleaning that might be the right thing to, but in this problem we might have to do it differently.

Now we want to see if there's a change in standard deviation.

```
sd(temperature_data$October, na.rm = TRUE)
```

```
## [1] 12.3336
```

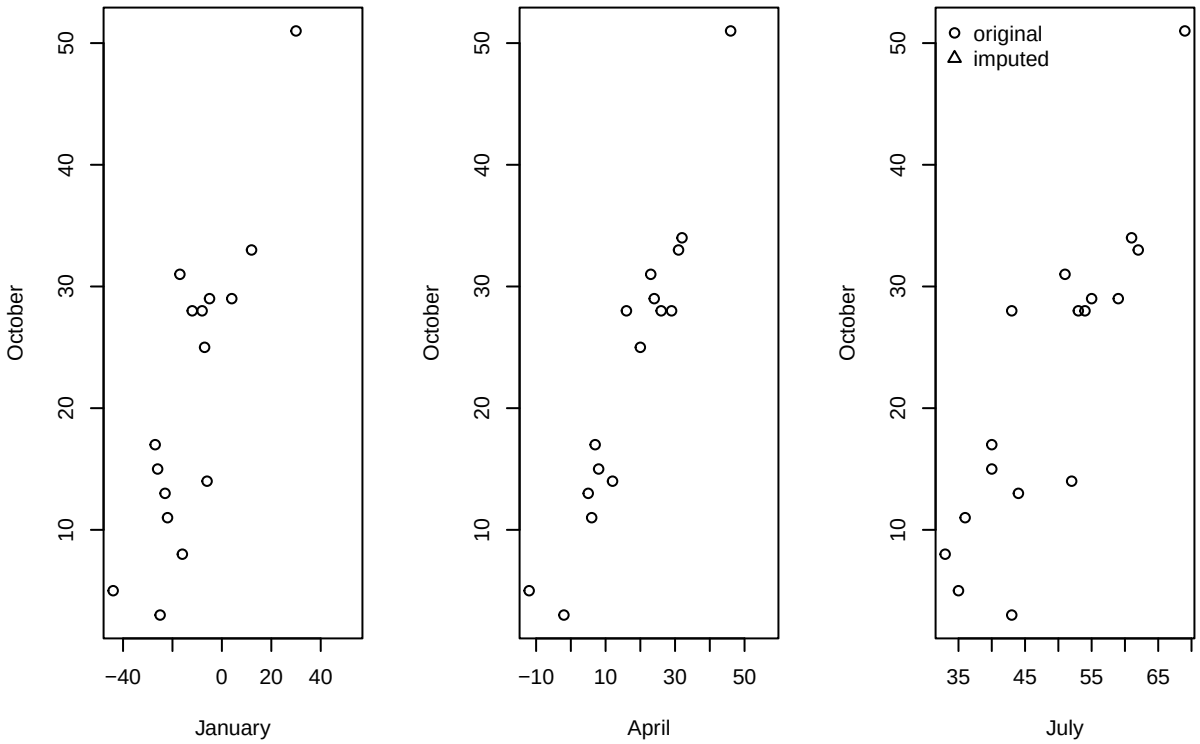
```
with(imp, sd(October))
```

```
## call :
## with.mids(data = imp, expr = sd(October))
##
## call1 :
## mice(data = temperature_data, m = 1, method = "mean", maxit = 1)
##
## nmis :
## [1] 0 2 2 1 4
##
## analyses :
## [[1]]
## [1] 11.09698
```

We can see that the standard deviation is 12.3336 just ignoring the NA's of our raw dataset. It's 11.09698 when we input the average mean and replace them with the NA's dataset. This means the values for October are clustered tightly around the mean more with the imputation of mean instead of having NA's. Which it might be a good method to replace our values, but we need to dive deeper into the analysis. What if we find a better way to do it?

First of all lets see if we can see a correlation between all our variables when comparing it to October.

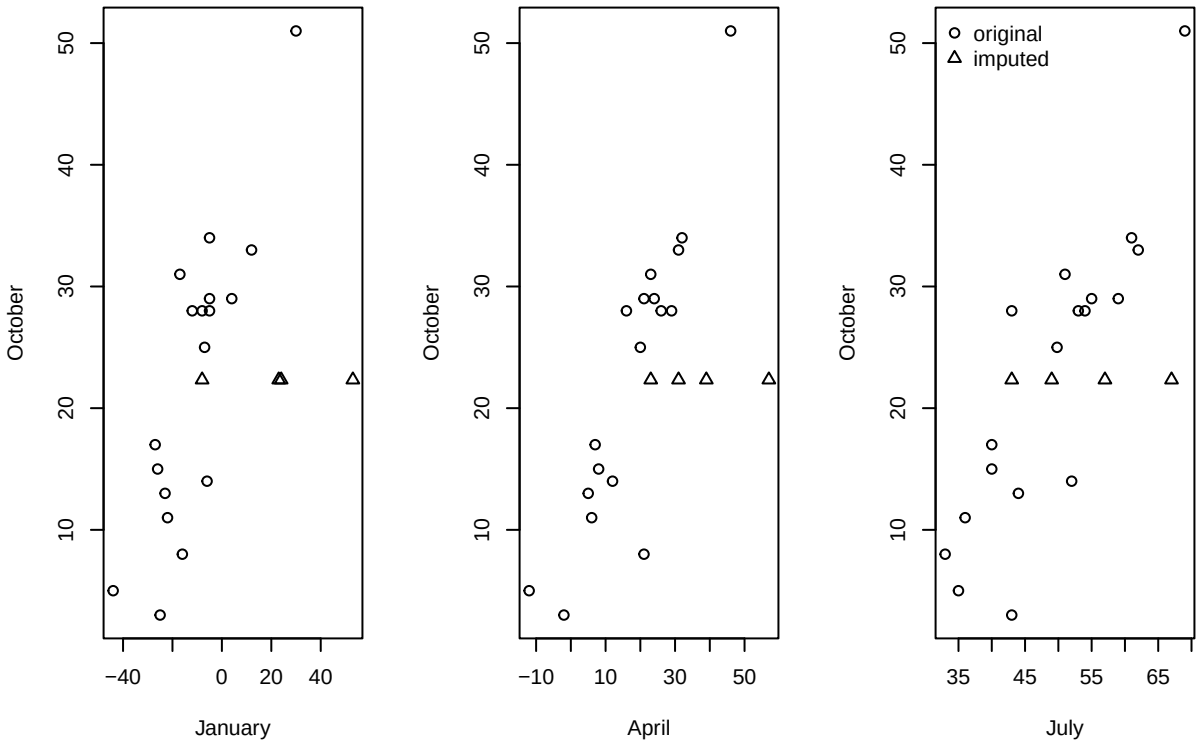
```
layout(matrix(1:3, nrow = 1))
plot(October ~ January, data = temperature_data)
plot(October ~ April, data = temperature_data)
plot(October ~ July, data = temperature_data)
legend("topleft", pch = 1:2, bty = "n",
legend = c("original", "imputed"))
```



We might see as the temperature increases in the months of January, April, and July the temperature in October seems to be going up. We see a positive correlation between the temperature of October compare to the temperature of other months.

Now we will show how those imputed values look like in a scatter plot, we expect to see all the mean values that are imputed to be in a horizontal line. We will compare the correlation between October to other months.

```
layout(matrix(1:3, nrow = 1))
plot(October ~ January, data = complete(imp),
pch = is.na(temperature_data$October) + 1)
plot(October ~ April, data = complete(imp),
pch = is.na(temperature_data$October) + 1)
plot(October ~ July, data = complete(imp),
pch = is.na(temperature_data$October) + 1)
legend("topleft", pch = 1:2, bty = "n",
legend = c("original", "imputed"))
```



As you can see all the imputed values are the mean of the temperate of October. As we can see this method of imputing value might not be the best approach to accurately input values based on many factors. We can see that we are suggesting that all the values that were imputed are the same in October no-matter what month we compare to such as January, April, or July. Now assuming that the months have correlation between each other, which sometimes they do when we look at the correlation between temperature and month. The imputed values don't follow the pattern that we see, so we might have to use a different method to impute values.

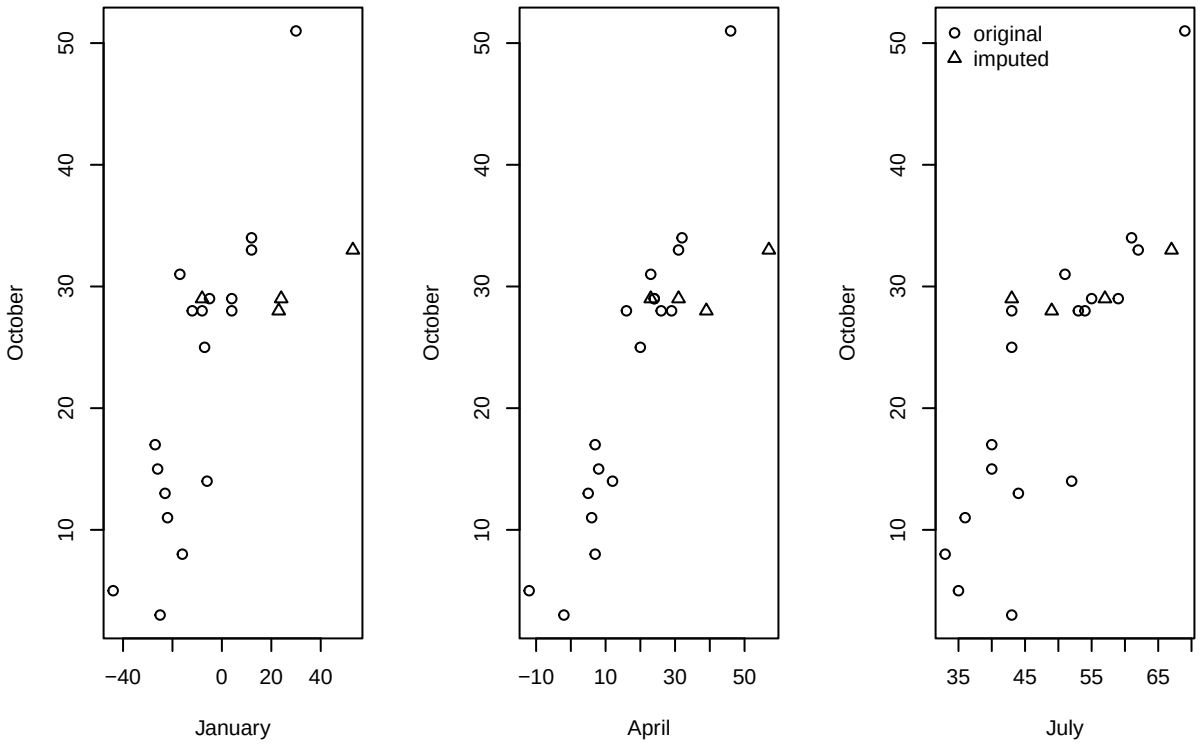
Now we will use "predictive mean matching", which is known as method = "pmm" which basically predicts values close to the missing value's prediction for imputation. This could be usually when we see a correlation between our variables, and the imputed value from the pmm will imput a value based on the pattern that they detec in the algorithm. We are doing 20 iterations, because our sample size is close to 20.

```
imp_ppm <- mice(temperature_data, m = 20, method = "pmm",
print = FALSE, seed = 1)
```

```
## Warning: Number of logged events: 1
```

We will graph the same plots but now we will the the "ppm" method in order to see how it works differently compared to "imp" method.

```
layout(matrix(1:3, nrow = 1))
plot(October ~ January, data = complete(imp_ppm),
pch = is.na(temperature_data$October) + 1)
plot(October ~ April, data = complete(imp_ppm),
pch = is.na(temperature_data$October) + 1)
plot(October ~ July, data = complete(imp_ppm),
pch = is.na(temperature_data$October) + 1)
legend("topleft", pch = 1:2, bty = "n",
legend = c("original", "imputed"))
```



We can see there are imputed values that follow the trends of the correlations of temperature between months, which can make us assume that we can use values for those values instead of NA's, but we just have to disclose that we put those missing values based on our "imp_ppm" method. We can say that when we use imp_ppm method we tend those values that were imputed follow the correlation that say before doing any method. This might be a better than method than predictive mean matching.

```
summary(unlist(with(imp_ppm, sd(October))$analyses))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.30   11.52   11.62   11.70   11.68   12.85
```

Right here it summarizes the standard deviations of the October variable across multiple imputed data sets in imp_ppm. As we can see have a minimum of 11.30 and a maximum of 12.85 even though our standard deviation might not be the lowest compared to imp method. It makes sense because imp was just imputing straight mean values which are totally inaccurate, these standard deviations have to vary because every value imputed is based on predictive mean matching.

Now we just will do some quick t.test in order to compare how each test vary.

```
with(temperature_data, t.test(October, mu = 20))
```

```
##
##  One Sample t-test
##
## data:  October
## t = 0.80264, df = 17, p-value = 0.4333
## alternative hypothesis: true mean is not equal to 20
## 95 percent confidence interval:
##  16.19998 28.46669
## sample estimates:
## mean of x
##  22.33333
```

```r
with(imp, t.test(October, mu = 20))$analyses[[1]]
```

```
##
##  One Sample t-test
##
## data:  October
## t = 0.98624, df = 21, p-value = 0.3352
## alternative hypothesis: true mean is not equal to 20
## 95 percent confidence interval:
##  17.41321 27.25346
## sample estimates:
## mean of x
##  22.33333
```

```r
fit <- with(imp_ppm, lm(I(October - 20) ~ 1))
summary(pool(fit))
```

```
##          term estimate std.error statistic       df   p.value
## 1 (Intercept) 3.738636  2.526899  1.479536 18.78248 0.1555748
```

We can see that when we test, in our case, the p-values are above 0.05, meaning there isn't strong evidence to reject the null hypothesis that the true mean temperature in October is 20°C. We can see how every model is different that they provide different p-test. Depending what your test and research consint we should be able to pick what method we think is more effective.

Read and Load Data

To conduct Imputation For Missing Values in R, it is important we install and load the package and the library mice. The mice package implements a method to deal with missing data. The package creates multiple imputations for multivariate missing data.

```r
fighter_stats <- read_csv("fighter_stats (1).csv")
```

```
## Rows: 2479 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): name, stance
## dbl (14): wins, losses, height, weight, reach, age, SLpM, sig_str_acc, SApM,...
##
## 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.
```

Exploratory Data Analysis

We don't have a column for weight division, we decide to mutate in order to make columns, because every division has a certain fighting division, we made created a new column, also the weight was in kg so we coveted into lbs and made sure every fighter was into their right category.

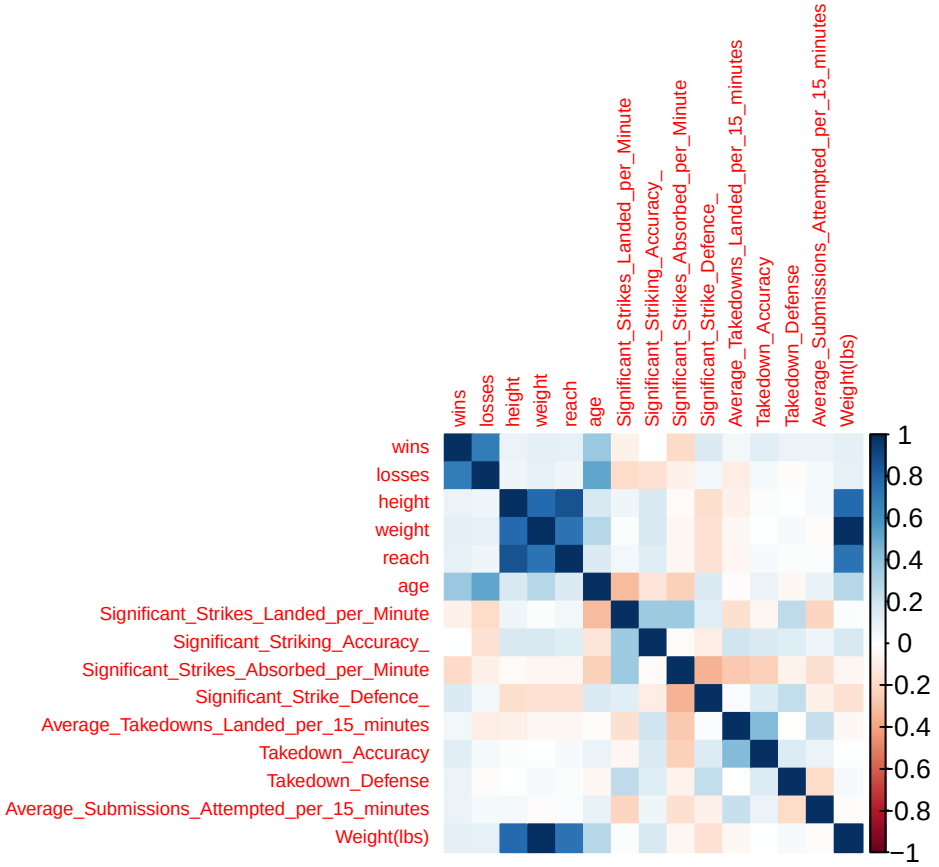
We want to rename our variables in order to better understanding of our variables. Names might get confusing.

We want to see a comparison between Reach and Weight Class.

```
## Warning: Removed 645 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

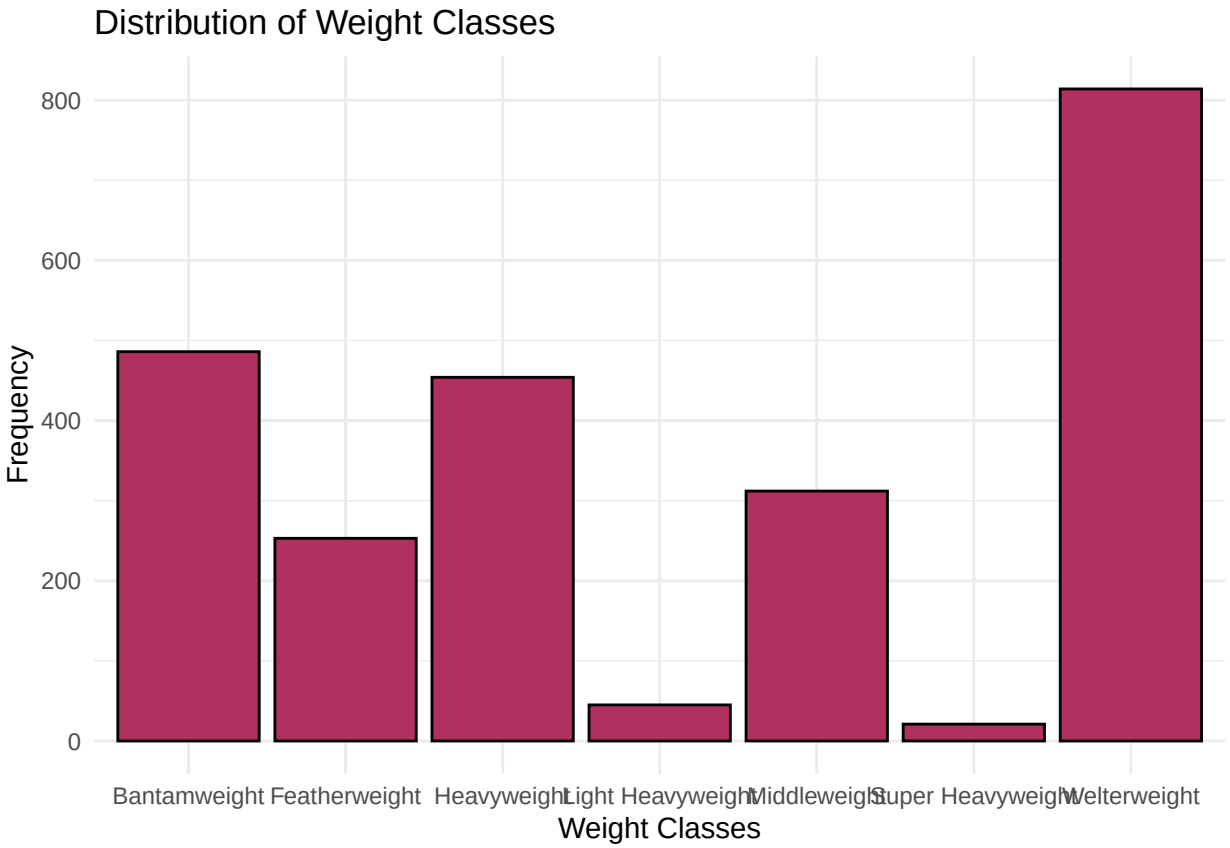
We can see a normal distributed data.We can see that even though we would think that there's a correlation between weight_class and significant strikes per minutes. We can say this because striking is one of the biggest attributes when it comes to fighting in the UFC. But we can see a correlation between weight classes and reach, and which make sense the more weight a fighter carries, they tend to be taller and being taller leads to a much longer reach.

A correlation matrix between all numeric variables.



the correlation plot shows a strong positive correlation between a fighter's weight and height. This indicates that taller fighters tend to be heavier, which is consistent. There is a positive correlation between offensive performance metrics such as significant strikes landed per minute and average takedowns landed per 15 minutes. This suggests that fighters who are more aggressive in striking are also more likely to attempt and succeed in takedowns, which can show an overall aggressive fighting style. The plot reveals correlations between defensive metrics such as strike defense and takedown defense. This implies that fighters who excel at defending against strikes also tend to be better at avoiding or countering takedowns. We can see more correlations.

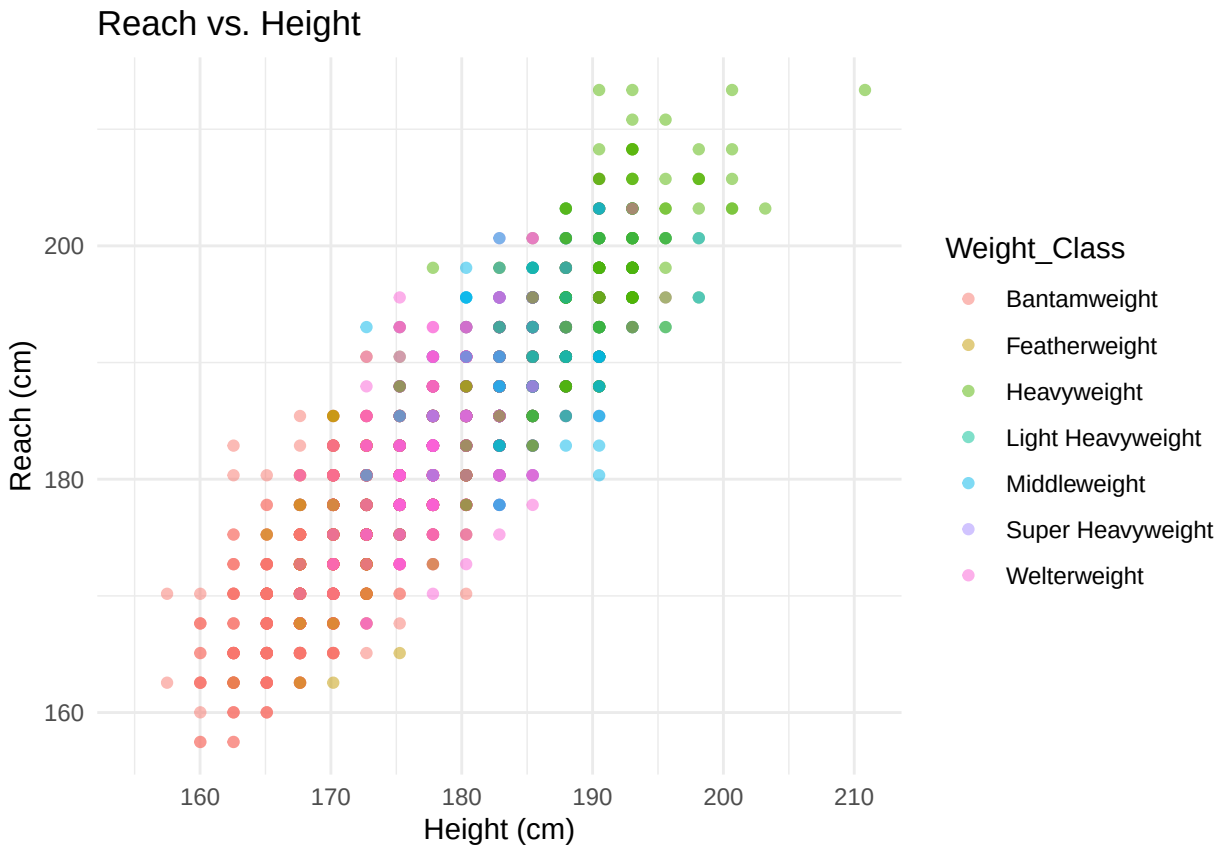
Distribution of Weight Classes



We can see the frequency across every weight division, the amount of fighters that there is the most popular weight division is welterweight and super heavyweight is the least popular weight decision in terms of number of fighters. The UFC divisions are mostly have the same watch popularity which is good for the buisness, but usually when we have light heavyweight and super heavy wight fights, since the amount of fighters in that division is so low, usually those fights are the most watched.

Reach vs Height

```
## Warning: Removed 645 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

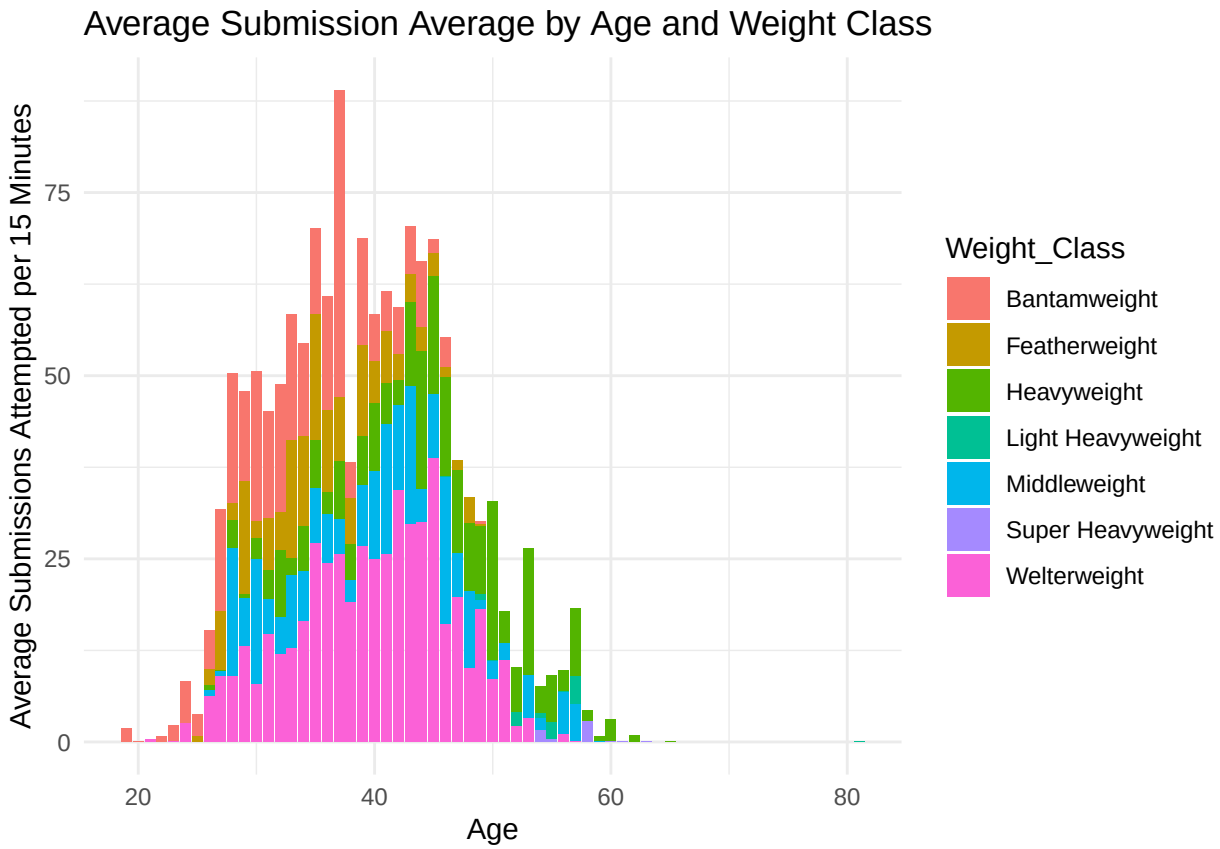


We can see that the strong positive relationship between height and reach. Which supports what our matrix plot shows.

Average Submission Average by Age and Weight Class

```
ggplot(fighter_stats, aes(x = age, y = Average_Submissions_Attempted_per_15_minutes, fill = Weight_Class
  geom_col() +
  labs(title = "Average Submission Average by Age and Weight Class",
       x = "Age",
       y = "Average Submissions Attempted per 15 Minutes") +  # Corrected label description
  theme_minimal()
```

```
## Warning: Removed 160 rows containing missing values or values outside the scale range
## (`geom_col()`).
```



We see that ages are evenly distributed through ages and how we can see the how in sporting events there's a term where is prime which can be consider when its around 25-35 who they are people who have the highest average reach making them strongest and most aggressive.We can see a pattern between all these 3 variables, we can see how age and the average submission are correlated and how all the weight classes tend to follow the same correlation.

Data Science Method: Imputation For Missing Values

We noticed that our dataset had many missing variables. This is mainly because we UFC started they were really counting significant strikes or televised certain fights so their statistics become unknown. We want those numbers as NA in order to impute the missing values based on other variables.

Replace all the zeroes in your dataset with NA (Not Available)

```
fighter_stats[fighter_stats == 0] <- NA
```

Counting how many NA's we have in our dataset using a sapply() and seeing how many NA have per variable.

```
na_count <- sapply(fighter_stats, function(x) sum(is.na(x)))
print(na_count)
```

```
##                                    name
##                                       0
##                                    wins
##                                      53
##                                  losses
##                                      43
##                                  height
```

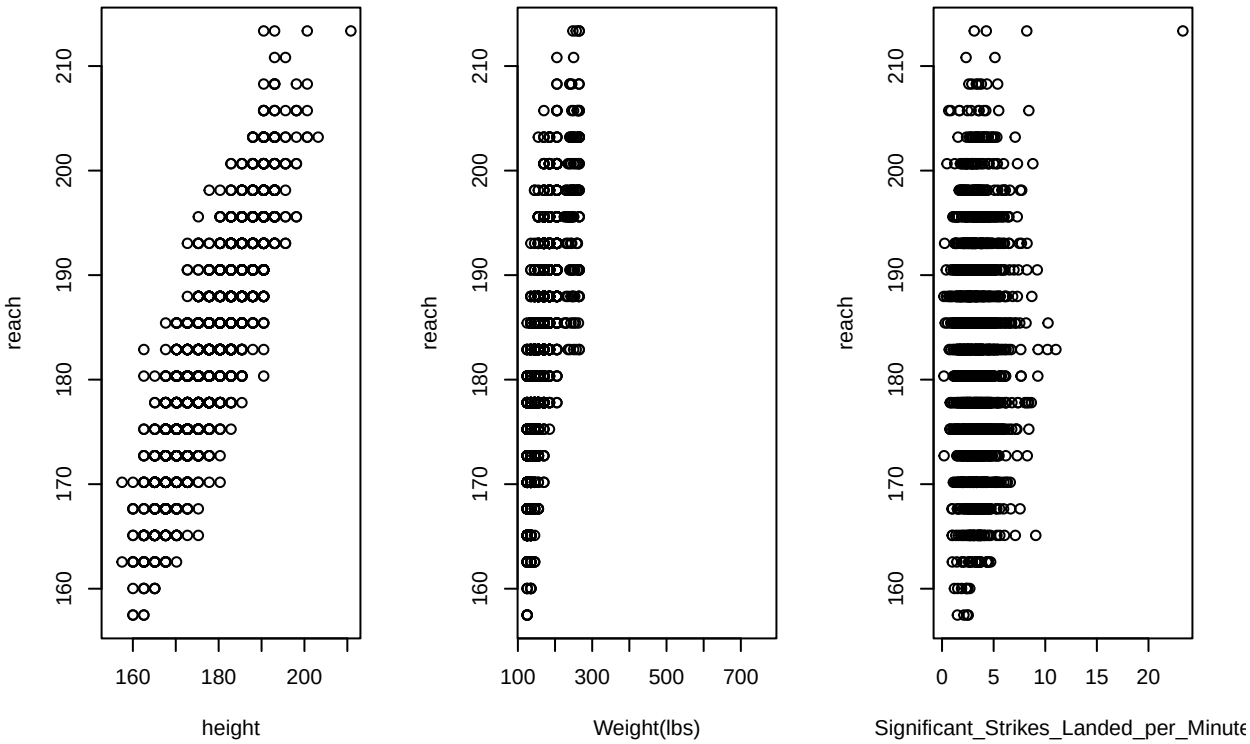
```
##                                          0
##                                     weight
##                                          0
##                                      reach
##                                        645
##                                     stance
##                                         75
##                                        age
##                                        160
##        Significant_Strikes_Landed_per_Minute
##                                        181
##               Significant_Striking_Accuracy_
##                                        181
##      Significant_Strikes_Absorbed_per_Minute
##                                        166
##                  Significant_Strike_Defence_
##                                        168
##      Average_Takedowns_Landed_per_15_minutes
##                                        602
##                             Takedown_Accuracy
##                                        602
##                              Takedown_Defense
##                                        434
## Average_Submissions_Attempted_per_15_minutes
##                                        993
##                                  Weight(lbs)
##                                          0
##                                 Weight_Class
##                                          0
```

The variables I want to impute are reach because we don't have any missing values for height so predicting each UFC fighter can be relatively easy. I also want to input values for reach because reach is something that I feel can have strong correlation with striking and height. The data about reach of a fighter shouldn't be NA because every fighter has a reach no matter how tall or short they are. 645 missing values for reach is concerning.

Exploratory data analysis for the 'reach' variable:

```
library(mice)

layout(matrix(1:3, nrow = 1))
plot(reach ~ height, data = fighter_stats)
plot(reach ~ `Weight(lbs)`, data = fighter_stats)
plot(reach ~ Significant_Strikes_Landed_per_Minute, data = fighter_stats)
```



The scatter plots display the relationships between fighters' reach and their height, weight, and significant strikes landed per minute. We can see that there is a positive correlation between height and reach, as well as between weight and reach, suggesting that taller and heavier fighters generally have a longer reach. But there is not to be a clear relationship between reach and the rate of significant strikes landed per minute.

Linear model summary for 'reach':

```
summary(lm(reach ~ height + `Weight(lbs)` + Significant_Strikes_Landed_per_Minute, data = fighter_stats)
```

```
##
## Call:
## lm(formula = reach ~ height + `Weight(lbs)` + Significant_Strikes_Landed_per_Minute,
##     data = fighter_stats)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3113  -3.5200  -0.0851   3.2867  15.2781
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         18.279367   3.230902   5.658 1.79e-08 ***
## height                               0.877011   0.021749  40.323  < 2e-16 ***
## `Weight(lbs)`                        0.049550   0.005408   9.162  < 2e-16 ***
## Significant_Strikes_Landed_per_Minute  0.023720   0.074958   0.316    0.752
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.756 on 1736 degrees of freedom
##   (645 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.7667, Adjusted R-squared:  0.7663
## F-statistic:  1901 on 3 and 1736 DF,  p-value: < 2.2e-16
```

We can see that p value for intercept, height, and weight, is significant, but its not significant with Significant Strikes Landed per Minute which totally correlates with our scatter plots.

Now as we did in the small example before we are going to do "imp" again. Which is single imputation using mean values:

```
imp <- mice(fighter_stats, method = "mean", m = 1, maxit = 1)
```

```
##
##  iter imp variable
##   1   1  wins  losses  reach  age  Significant_Strikes_Landed_per_Minute  Significant_Striking_Accura
```

```
## Warning: Number of logged events: 4
```

```
with(imp, summary(reach))
```

```
## call :
## with.mids(data = imp, expr = summary(reach))
##
## call1 :
## mice(data = fighter_stats, m = 1, method = "mean", maxit = 1)
##
## nmis :
##  [1]   0  53  43   0   0 645  75 160 181 181 166 168 602 602 434 993   0   0
##
## analyses :
## [[1]]
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   157.5   177.8   183.0   183.0   188.0   213.4
```

```
with(imp, sd(reach))
```

```
## call :
## with.mids(data = imp, expr = sd(reach))
##
## call1 :
## mice(data = fighter_stats, m = 1, method = "mean", maxit = 1)
##
## nmis :
##  [1]   0  53  43   0   0 645  75 160 181 181 166 168 602 602 434 993   0   0
##
## analyses :
## [[1]]
## [1] 8.402345
```

```
with(imp, cor(height, reach))
```

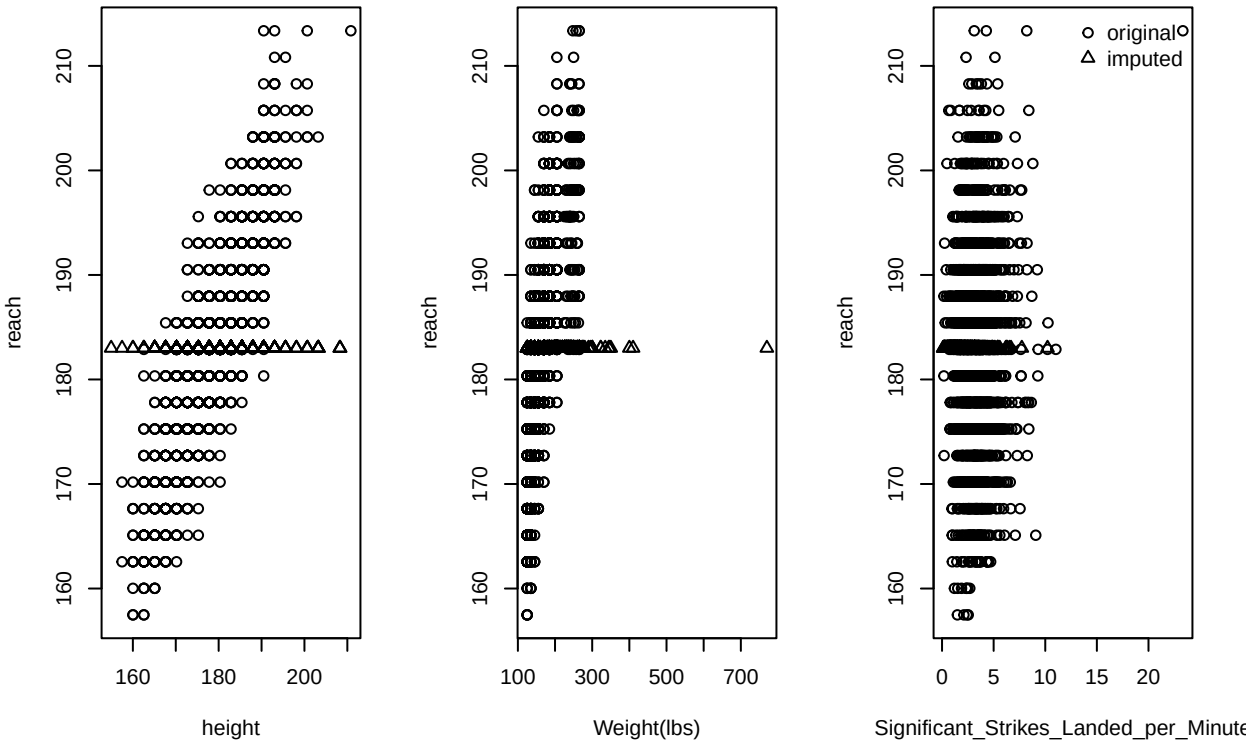
```
## call :
## with.mids(data = imp, expr = cor(height, reach))
##
## call1 :
## mice(data = fighter_stats, m = 1, method = "mean", maxit = 1)
##
## nmis :
##  [1]   0  53  43   0   0 645  75 160 181 181 166 168 602 602 434 993   0   0
```

```
##
## analyses :
## [[1]]
## [1] 0.7448594
```

First of all we can see a strong correlation between height and reach which is around .74485, which makes sense. We can see a standard deviation of 8.402345 which is significantly high. More research is needed. Now let see our mean imputed values and see how does it look in a plot.

Visualizing imputed data:

```
layout(matrix(1:3, nrow = 1))
plot(reach ~ height, data = complete(imp),
pch = is.na(fighter_stats$reach) + 1)
plot(reach ~ `Weight(lbs)`, data = complete(imp),
pch = is.na(fighter_stats$reach) + 1)
plot(reach ~ `Significant_Strikes_Landed_per_Minute`, data = complete(imp),
pch = is.na(fighter_stats$reach) + 1)
legend("topright", pch = 1:2, bty = "n",
legend = c("original", "imputed"))
```



We see that the values that were imputed are the means of reach, which it's nice visualization as we can we see where the missing values are really located. But in reality, it just a horizontal line of imputed value which makes sense because its the average of reach imputed across height,weight, and significant strikes landed per minute. But we need to do predictive mean matching in order to determine which one is better.

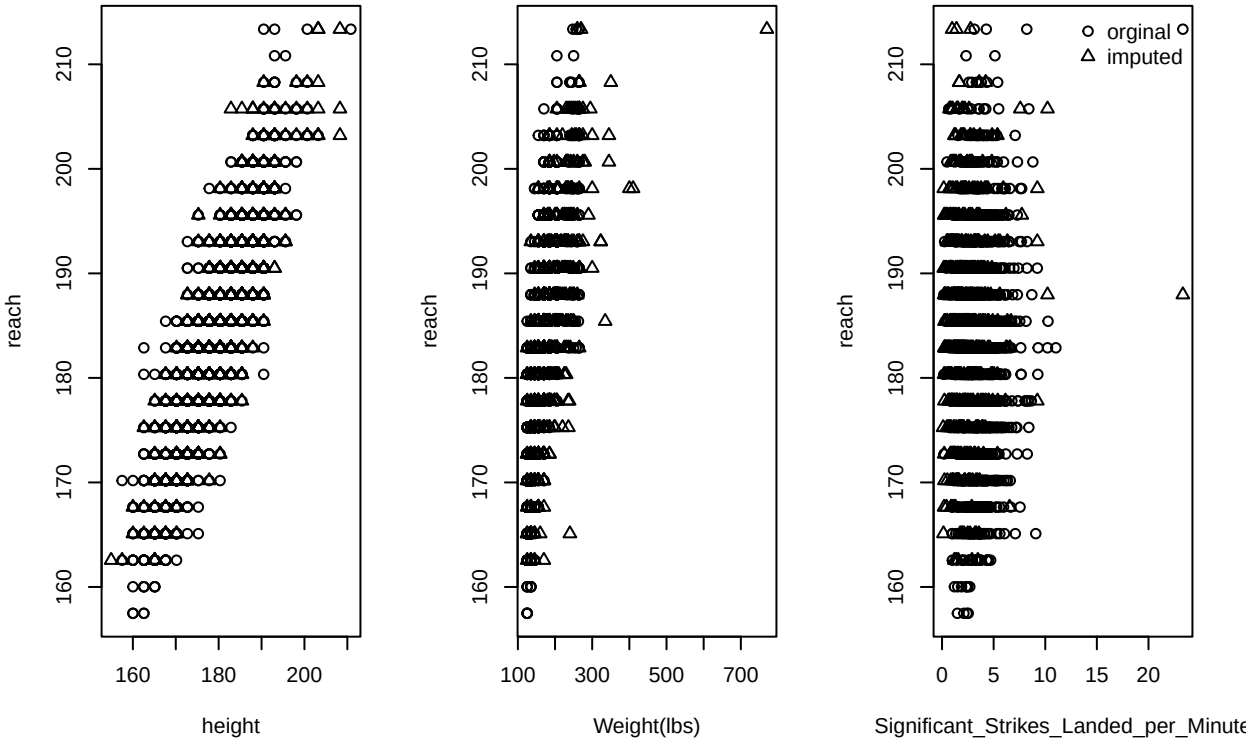
Now we use Multiple Imputation using predictive mean matching, we do 50 iterations, compared to 20 in the example.

```
imp_ppm <- mice(fighter_stats, m = 50, method = "pmm",
print = FALSE, seed = 1)
```

```
## Warning: Number of logged events: 4
```

Visualizing results from multiple imputation:

```
layout(matrix(1:3,nrow = 1))
plot(reach ~ height, data= complete(imp_ppm),
pch = is.na(fighter_stats$reach) + 1)
plot(reach ~ `Weight(lbs)`, data= complete(imp_ppm),
pch = is.na(fighter_stats$reach) + 1)
plot(reach ~ `Significant_Strikes_Landed_per_Minute`, data = complete(imp_ppm),
pch = is.na(fighter_stats$reach) + 1)
legend("topright", pch = 1:2, bty = "n",
       legend = c("orginal", "imputed"))
```



As we can see the values that were imputed mostly follow the pattern that it does when its two variables, which can help us predict in a way what would happen if the actual statistics were imputed. Since UFC is a really unpredictable sport, this could give us a sense of what would a fighter would have been if we didn't know the actual statistics. Since weights and height have a strong correlation with reach we have to use this model better.

Summary statistics of multiple imputation results:

```
summary(unlist(with(imp_ppm, sd(reach))$analyses))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.686   9.764   9.804   9.799   9.830   9.918
```

```
summary(unlist(with(imp_ppm,
    cor(height, reach))$analyses))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##  0.8597  0.8636  0.8647  0.8652  0.8671  0.8706
```

```
summary(unlist(with(imp_ppm,
    cor(`Weight(lbs)`, reach))$analyses))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6727  0.6850  0.6896  0.6886  0.6933  0.7019
```

```
sd(fighter_stats$reach, na.rm = TRUE)
```

```
## [1] 9.837933
```

We can see that our standard deviation might be higher than the actual standard deviation, but that is fine because as mentioned earlier, UFC is one of the most unpredictable sports, because there are so many factors that play into fight day such as adrenaline, timing, and training. The purpose of mice is not to improve the standard deviation of our dataset but instead to impute missing values making sure our dataset doesn't include NA. Indeed, adding predicted values will usually increase the standard deviation but that's not always the case. Now we will do a T test in order to see how testing with a certain mean value.

Now we test our models without imp, with imp_pp, and with imp_ppm. We should see a difference in the p-value.

```
t.test(fighter_stats$reach, mu =183)
```

```
##
##  One Sample t-test
##
## data:  fighter_stats$reach
## t = 0.011112, df = 1739, p-value = 0.9911
## alternative hypothesis: true mean is not equal to 183
## 95 percent confidence interval:
##  182.5400 183.4652
## sample estimates:
## mean of x
##  183.0026
```

```
with(imp, t.test(reach, mu = 183))$analyses[[1]]
```

```
##
##  One Sample t-test
##
## data:  reach
## t = 0.015232, df = 2384, p-value = 0.9878
## alternative hypothesis: true mean is not equal to 183
## 95 percent confidence interval:
##  182.6652 183.3400
## sample estimates:
## mean of x
##  183.0026
```

```
with(imp_ppm, t.test(reach, mu = 183))$analyses[[1]]
```

```
##
##  One Sample t-test
##
## data:  reach
## t = 3.294, df = 2384, p-value = 0.001002
## alternative hypothesis: true mean is not equal to 183
## 95 percent confidence interval:
```

```
##  183.2665 184.0505
## sample estimates:
## mean of x
##  183.6585
```

Imp_ppm likely provided a more accurate and varied estimation of the missing data. Multiple imputation generally produces more reliable estimates because it accounts for the uncertainty in the imputation process by creating several different imputations and pooling the results. We can also conclude that models can perform more than others, there might be instances where a model with just the mean of the variable imputed might be different, and there might be instances where the predictive mean matching might be better. It all depends on the context of the dataset. As we can our p-value was less than 0.05 for imp_ppm, while in the other hand normally and imp were relative the same, and way bigger than 0.05. Which this why for imputing values for reach, I would use imp_ppm.

Conclusion

I believe if you have missing values, and you have to impute them and the variable that has missing values is not correlated to your dataset I will use imp, but if you have strong correlation between all your variables across I would use imp_ppm. It all to the researcher discretion.

Bibliography

@misc{everitt2013hsaur, title={HSAUR: a handbook of statistical analyses using R. R package}, author={Everitt, BS and Hothorn, T}, year={2013} }

‘‘‘