A Poisson Analysis of Pit Stops in Formula 1

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Andrew Estes

Dr. Tetyana Beregovska, Thesis Advisor

Department of Statistics

School of Science and Mathematics

2023

TRUMAN STATE UNIVERSITY
Kirksville, Missouri

TABLE OF CONTENTS

A Poisson Analysis of Pit Stops in Formula 1
An Abstract of the Thesis by
Andrew Estes

This paper attempts to analyze and predict when a pit stop would be made in a Formula1 race. A predictive model could improve efficiency within the race and optimize the race strategy. With lap-by-lap data, the distribution steered us to perform a Poisson analysis. We attempted several Poisson models, including but not limited to, Poisson Regression, Consul's Poisson, and a Bayesian Poisson Regression. There were issues with over-dispersion which we counteracted with additional models, such as Negative Binomial. None of the models worked. The errors included convergence not being reached for Poisson models as well as maximum tree depth being exceeded in the Bayesian model. It is our recommendation that if further analysis is done on the topic, that the researcher should look only at a specific course over a period of time rather than all courses. It would be ideal to incorporate the weather data which we were unable to do given the data layout.

Introduction

Formula 1, also referred to as Formula One and F1, cars are the fastest road-course racing cars in the world – often exceeding 200 mph seconds after braking to below 60 mph. There are ten teams, each with two drivers, competing for two concurrent awards – best driver and best constructor.

The driver with the most points at the end of a season is awarded the Drivers' Championship Award. Teams who have accumulated the most points from their drivers receive the Constructors' Championship Award. The point breakdown is seen in Table 1.

Table 1 – Point Breakdown

Position	Points Scored
1	25
2	18
3	15
4	12
5	10
6	8
7	6
8	4
9	2
10	1
11 thru 20	0

In 2021, Red Bull drivers finished 1st and 4th overall, with the two drivers scoring a combined 585 points. That same season, Mercedes drivers finished 2nd and 3rd overall with the two drivers scoring a combined 613 points. Red Bull had the top driver while Mercedes was deemed the top constructor.

Construction of the car plays a predominant role in Formula1 racing. Bell et al. (2016) used a multi-level (random coefficients) linear model to estimate whether driver skill or car construction most effected performance (as reflected by points scored). They found 86% of the variance in points scored came from the car construction and only 14% of the points came from driver skill. Kesteren and Bergkamp (2022) used a Bayesian Multilevel Beta regression method to distinguish driver skill from constructor advantage. Their findings also found an 86% variance coming from car construction with 14% of variance from driver skill.

Much of the disparity between cars' construction can be attributed to budget. Until 2021, there was no cost cap for car design and construction. Asher (2022) breaks down what the cost cap does and does not cover. It does cover all the parts on the car, transportation costs, most team personnel, garage equipment, etc. Notably, driver salaries, as well as the wages of the three highest staff members and travel costs, are not included in the salary cap. If a driver can make up 14% of the difference in results, a team with more spending power is at an immediate advantage.

Outside of the car construction capabilities and driver skill, success in F1 racing may be affected by other variables that have not been studied extensively. The number of races in a season, for example, may affect performance. The past five seasons utilized in this study have ranged from (17) to (22) races per season (a 29% variance), and the current 2023 season will have 23 races.

Table 2 - Frequency of Courses Per Year							
Course #	***Course***	***2018***	***2019***	***2020***	***2021***	***2022***	**Sub-Total**
1	*Abu Dhabi*	0	1	1	1	1	**4**
2	*Australia*	1	1	0	0	1	**3**
3	*Austria*	1	1	2	2	1	**7**
4	*Azerbaijan*	1	1	0	1	1	**4**
5	*Bahrain*	1	1	2	1	1	**6**
6	*Belgium*	1	1	1	1	1	**5**
7	*Brazil*	1	1	0	1	1	**4**
8	*Canada*	1	1	0	0	1	**3**
9	*China*	1	1	0	0	0	**2**
10	*France*	1	1	0	1	1	**4**
11	*Germany*	1	1	1	0	0	**3**
12	*Great Britain*	1	1	2	1	1	**6**
13	*Hungary*	1	1	1	1	1	**5**
14	*Italy*	1	1	3	2	2	**9**
15	*Japan*	1	1	0	0	1	**3**
16	*Mexico*	1	1	0	1	1	**4**
17	*Monaco*	1	1	0	1	1	**4**
18	*Netherlands*	0	0	0	1	1	**2**
19	*Portugal*	0	0	1	1	0	**2**
20	*Qatar*	0	0	0	1	0	**1**
21	*Russia*	1	1	1	1	0	**4**
22	*Saudi Arabia*	0	0	0	1	1	**2**
23	*Singapore*	1	1	0	0	1	**3**
24	*Spain*	1	1	1	1	1	**5**
25	*Turkey*	0	0	1	1	0	**2**
26	*United Arab Emirates*	1	0	0	0	0	**1**
27	*United States*	1	1	0	1	2	**5**
	Sub-Total	**21**	**21**	**17**	**22**	**22**	**103**

Furthermore, the courses themselves may impact the car and driver performance. Not only does each course follow a unique pattern, but there are also two different road surfaces – track or street. A track course is a dedicated racing track. A street circuit is a collection of public roads that are closed off for the purposes of the F1 race. In 2022, five of the twenty-two (22.7%) courses were street courses. In 2023, 30.4% (seven of the 23) of the races will be on the street.

Finally, climate plays a role in racing conditions and overall performance. The wide geographic variance of the races and time of year for each race creates significant challenges for both the vehicles and the drivers. Track temperatures between the coldest and warmest race locations may vary by nearly 100 degrees Fahrenheit. Similarly, humidity varies between arid deserts with practically no humidity to wetter locales with 92.5% more moisture. Optimally, any F1 performance analysis would include these factors as they play a significant role in tire degradation and general pit stop strategy. Due to how the weather data is accumulated, it is unable to be merged with the other datasets we have; therefore, we acknowledge access to the data and its importance, but also the inability to properly assess its impact to our study.

Table 3 – Basic Climate Data

	Air Temperature (Fahrenheit)	Track Temperature (Fahrenheit)	Humidity (%)	Air Pressure (millibar)
Low	48	56.8	5	779.5
High	99	152.6	97.5	1023.5
Average	74.5	95.5	54.8	986.2
Standard Deviation	41.1	49.5	18.6	49.2

This study also does not have access to all the F1 data which might help the model's predictive power. Over the course of the race weekend, up to three terabytes of data are generated per the Mercedes-AMG Petronas Formula One Team (2022). This is accumulated through three practice sessions, a qualifying race (which determines starting order), and the final race. This data helps mechanical engineers adjust the vehicles to increase their performance capabilities.

While we do not have access to this data, we can obtain some information through the Ergast Developer API and the Formula 1 Official Data Stream. There is a python package called FastF1 used for accessing and analyzing the race results, schedules, timing data, and telemetry. It was created by a

German engineer (https://github.com/theOehrly/Fast-F1) and much of his work and documentation (https://theoehrly.github.io/Fast-F1/) has been used as the foundation of this paper.

The limited nature of historical data available restricts the time period for this study. Much of the timing data only dates back to 2018 as the information is not publicly available for prior years. As such, this study focused on the years beginning in 2018 and ending in 2022. This five-year period provided 103 races from at least 27 different courses. The uniqueness of courses can be obfuscated by country name as some countries can provide multiple courses such as the United States in 2023 with Miami, Las Vegas, and Austin providing courses. In the analysis, there is not a clear indicator of which track is being used at any given point in time.

Looking at these races, and acknowledging previous studies found car construction differences account for 86% of the race results, this study sought to examine another factor which might affect the outcome of a race. Once the race starts there is no ability to change the driver or make modifications to the engine. The only possible controllable change once the race starts is the decision to make a pit stop.

Drivers perform pit stops to change tires. Timing of the pit stop and the type of tires utilized on the car both play a significant role in racing. Deciding what tires to put on the car is an optimization problem that has not yet been solved. This study does not attempt to optimize the tire decision, but rather focuses on whether the need for a pit stop at all may be predicted using a Bayesian Equilibrium analytic model. The idea stemmed from Bruce Bueno de Mesquita's book The Predictioneer's Game (2010) in which he discusses his prediction of political policies individuals will institute.

The personnel involved with the pit stop is similarly difficult to factor into the study. Each team has twenty mechanics who partake in the pitstop. They can change four tires in two seconds The decision to make the pit stop falls on the "Race Strategy Engineer." That person uses data, simulation, and feedback to make the decision. The role, however, is not as clearly defined as it seems – some teams do not even identify who their main strategy engineer is and what their role is in the pre-race strategy plans and simulation optimization.

Image 1 – Pit Stop Crew In Action

https://www.caranddriver.com/news/a28567721/fastest-pit-stop-german-grand-prix/

In addition to determining when a pit stop will be made, the strategist decides on the type of tire to use. Unfortunately, even when a pit stop is called for, our data is not 100% accurate. There are six dry tire types, ranging in hardness from C0 (very hard) to C5 (very soft). We only get told if it is soft, medium, or hard. There are also intermediate tires and wet tires. Another consideration with compounds is the requirement that at least two different types of compounds are used in a race.

Tulabadin and Rudin (2014) performed a similar analysis on NASCAR races. They found that machine learning algorithms such as ridge regression, SVR, LASSO, and Random Forests are significantly predictive. Their question concerned tire optimization in NASCAR racing, comparing whether it was better to change all four tires or just two. In Formula1, all four tires must be changed at the same time, but there is variability as to the type of tire used as there are six variations (Pirelli).

Furthermore, an important random event, or shock, that can alter the face of the race is a flag. There are ten distinct flags which have their own meaning. A yellow flag, for example, can be the result of a minor car crash. In a yellow flag, all drivers must slow their speed and passing another car is not

allowed. As a result, yellow flags are an ideal time to make a pit stop for fresh tires. Pretorious (2022) has a chart that shows the breakdown of flags and their meaning:

Table 4 – Explanation of Various Flags	
Flag Color/Type	**Meaning**
Yellow	Hazard on track.
Green	Normal racing conditions apply.
Red	Session is suspended.
Blue	A faster car is approaching. Move aside (when a driver is being lapped).
Yellow & Red Stripes	Track is slippery.
Black With an Orange Circle	A driver has a mechanical issue and must return to the pits.
Black & White	Warning for unsportsmanlike behavior.
Black	Disqualification.
White	Slower moving vehicles ahead, or miscellaneous vehicles on the track.
Checkered	The session is finished (no new laps may be started).

This flag data can be found in the "Laps" dataframes under track status. Below is a breakdown of the variables on the Laps dataframe, and underneath that is the breakdown of the various TrackStatus definitions.

Table 5 – Description of Track Status and Flags	
Track Status	**Description**
1	Track clear
2	Yellow flag
3	Unknown at this time
4	Safety Car
5	Red Flag
6	Virtual Safety Car deployed
7	Virtual Safety Car ending

The reason we care about flags is they make a significant impact to the race. They often indicate one or more cars have crashed. In addition to a competitor (or multiple competitors) being out of the running, there is usually a mandatory slow-down of speed between 30% and 40% of normal race-pace. It usually takes teams around 20 seconds to fully complete a pit stop (2-3 seconds for the tire exchange and the rest for driving in/out of the pit stop lane). If every other driver is driving 40% slower, then there are significant time-savings available to drivers utilizing a pit stop while others are still on the course.

Since pit stops are one of the few things that can be changed during the race and seeing as strategists do care about when another team makes a pit stop, it seems to be a good challenge to try and predict when that pit stop will occur. Similar to how Chimka and Talafuse (2016) used a Poisson Regression to compare golf strokes hit before making it in the hole, we first wanted to look at the distribution of discrete data.

Figure 1 – Lap Number When Pit Stop Made

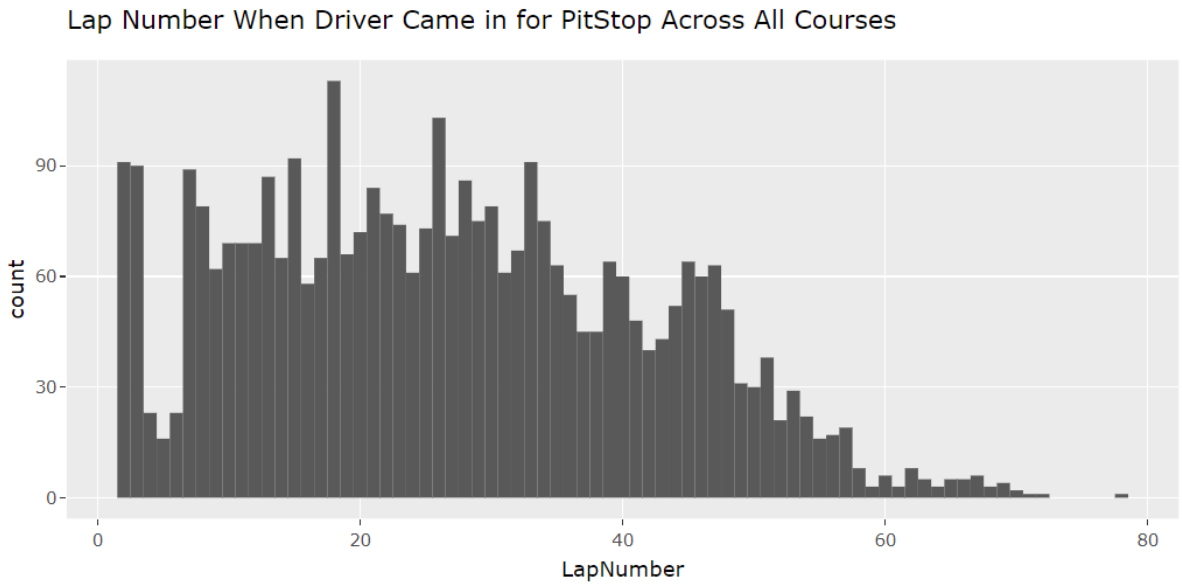
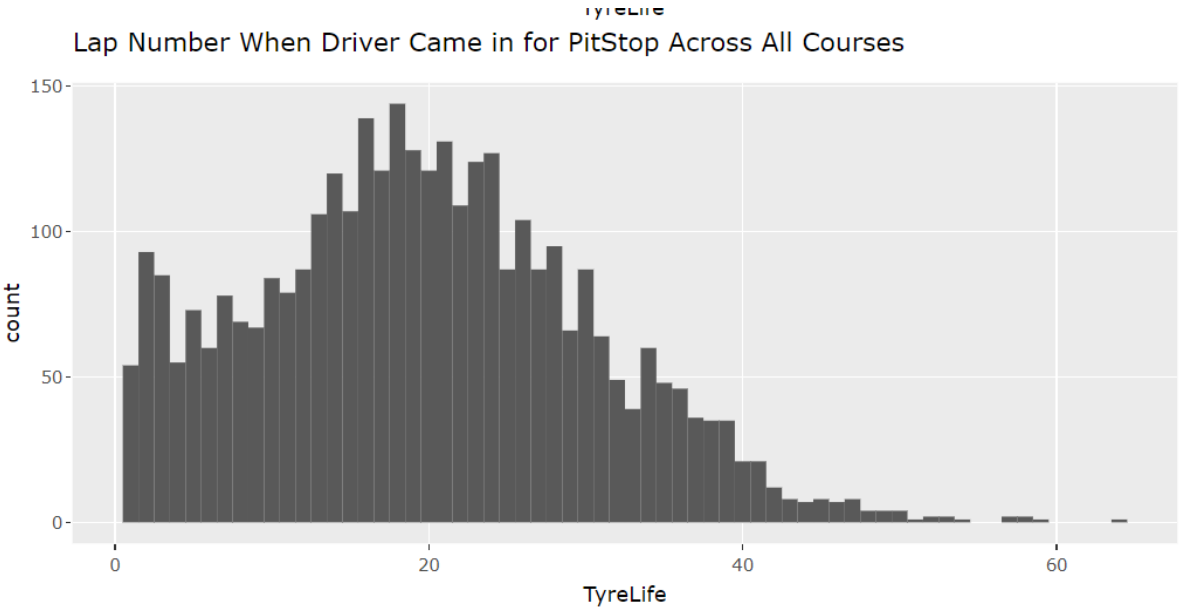


Figure 2 – Age of Tyre When Pit Stop Made



Solving for Bayesian Equilibrium requires the use of Python and R to extract, manipulate, and visualize the data. We have global variables, local variables, and random shocks. The Bayesian statistical model appeared to be the best approach for this study because, as Santos et al (2018) noted, it incorporates prior beliefs, probabilistic estimates, and can update likelihood lap-after-lap likelihood.

Methods

Initially the FastF1 package was accessed via JupyterLab. After utilizing JupyterLab via Anaconda for the initial setup, we transferred the workstation to Spyder IDE via Anaconda. Python is where the majority of the programming and analytical work is done. Visualization was done in Python and R. With R, we used Shiny for exploratory data analysis.

Unfortunately, there was no cumulative dataframe for each year. Instead, each race has its own group of dataframes (one for results, one for the lap-by-lap information, one for weather, etc.). This is due to how the data gets collected into the FastF1 package.

Our first step was to put each race into a singular dataframe for the year. The first thing we had to do was to merge the “laps” dataframe with the “results” dataframe for each race. The “laps” dataframe had 27 columns and between 60 and 1533 rows.

The “results” dataframe has 17 columns and 20 rows. In the results dataframe, we had to rename “Abbreviation” into “Driver” to complete the merge as there were no commonly named columns. The results dataframe had multiple duplicate variables as the laps dataframe, so we only grabbed 'Driver', 'GridPosition', 'Position', 'Points'. This helped keep the merged dataframe to a manageable 31 columns. An issue with Results dataframe was a programming error that prevented us from viewing the dataframe in the windowpane like normal. We had to write the Results output to CSV and view it in Excel to understand what the data looked like.

After merging the laps and results dataframe, we created a new column called ‘TotalTime’ that calculated the total time a driver had driven. This let us figure out how long it took each driver to reach a certain lap in a race and therefore we were able to sort the race by their total time. The issue we ran across was drivers who did not complete the race, or who were lapped, became intermingled in the final results. The only way to account for those drivers would be to remove them from the dataframe – which would reduce the total number of rows from 110,124 to 3,415, a 97% loss of data.

With our goal of predicting pit stops, we determined that knowing the positioning of each driver at the end of each lap was important. We used the Saudi Arabia race in 2021 as our test data set. We

transformed all the timing data into date-time type in seconds and performed column-wise addition to find the total lap time.

The second step was to calculate the pit stop time. The pit stop location on all racecourses is at the start/finish line. This means that whenever a pit stop happens, the “pit stop in” time happens on lap “i” and the “pit stop out” time happens on lap “i + 1”. While this is not intuitive, this is accurate because of the location of the pit stop on the track is right beside the start/finish line on the course. In order to determine the true pit stop time, we used the ‘shift’ function in python to perform the diagonal subtraction of “pit in time” from “pit out time” with a lower bound of 0.

There were quite a number of issues that populated with pit stops. To begin with, every driver’s first lap included a pit stop exit time. This obviously is not true to the race as the drivers do not begin in the pit stop lane. This issue was overcome with the initial setup of the pit stop calculation by forcing the pit stop time to be 0.

The second issue was the timing associated with the first pit stop. The first record showed a pit out time of 24 minutes 6 seconds and 51 milliseconds. It obviously did not take any driver 24 minutes to be in the pit stop area before the first lap of the race was initiated.

One way to combat the timing was to look at sector times. There are three sectors in a race, and if we sum all three sectors, we should get the total time of the lap per driver. Sector time of a lap per driver is based upon their actual timing in that sector. However, whenever there was a pit stop out time, there was no sector 1 time.

Sector time per session, while similarly named, represents different value than sector time. This variable had additional timing issues - for example, sector 2 time in the Brazil 2021 race in the first observation was 29 seconds and 642 milliseconds. However, the sector time per session variable showed the sector 2 time of 1 hour 2 minutes 43 seconds and 773 milliseconds. The difference is the “session” which has nothing at all to do with the start of the race. Sector time per session was measured from the start of television coverage, which frequently began prior to the actual start of the race.

We then created a cumulative time per lap by driver. We want to know how much time has elapsed for each driver at the end of each lap to determine their positioning. Pandas has a built-in cumulative summing function that was used in conjunction with the *groupby* method.

We sorted that output by time. We are in the position to know the results of the race. Based off the cumulative time, the ending order should result with the point structure: 26, 18, 15, 12, 10, 8, 6, 4, 2, 1, 0, 0. However, our output was well off: 12, 26, 15, 18, 10, 2, 1, 4, 0, 6, 8, 0 (only 3rd and 7th are accurate)

The issue was mainly with the pit stop times. Due to red flags and potentially other unknown features, the length of pit stop times became inaccurate as the race was re-started and drivers in first place spent longer in the pit stop than others.

We then removed pit stops and just sorted by race time, which resulted in closer to accurate results: 18, 26, 12, 15, 10, 8, 6, 4, 1, 2, 0, 0. There are three positions that are mistaken, and they are all off by exactly one spot (1st vs 2nd, 3rd vs 4th, and 8th vs 9th).

Another attempt was made by removing all the rows with red flags. This change resulted in a final sorting of 12, 26, 15, 18, 10, 2, 1, 4, 0, 6, 8, 0 (seven of the 12 positions are incorrect).

The timing data is a such a known issue within the data that there is an entire column dedicated to determining if the start lap time and end lap time are synced up correctly.

Since our other attempts left us worse off than the original calculation, we returned to the original calculation with the caveat that sometimes non-finishers are included in the final lap's data. It was not efficient to continue attempting to manipulate the data to obtain the most accurate results. With the original calculation, we would have to determine the number of non-finishing or lapped drivers who were interspersed in the final results and remove them from the dataframe. This was not feasible to calculate as it was a manual adjustment for 103 races, which would require a comparison of 2060 dataframe rows to real-world results. Furthermore, doing so would not actually solve the problem – it would focus on the top finishers, but the point of the study was to predict pit stops for all drivers at any given lap at any given

racecourse. We created a “gap” column to indicate how far behind or ahead a driver was compared to the next closest driver.

After arranging the data into an imperfect but acceptable format, we performed a Poisson regression on the data. We wanted to predict when a pit stop will be made so we pared the dataframe down into just those laps involving a pit stop. We also removed duplicate and unnecessary column values. This reduced the columns to 16 and rows to 3415. The total time variable was changed into a total seconds rather than a dtype(‘<M8[ns]’) type.

We were left with five string variables. To get them in the proper form, we had to transform them into categories and then codify them before conducting statistical analysis. The first thing we did was aggregate the pit stop only data by laps, calculating the 'mean', 'min', 'max', 'count'. We had to transform the count into a float. We were then able to visualize the number of pit stops both from a total number and from a normalized viewpoint. We then looked specifically at the tire life when the pit stop was made, using the same parameters in the aggregated calculations for pit stops by lap.

Our next step was to perform statistics. Sci-kit Learning provides a Python framework for calculating the Poisson regression. It led to a value error in our data using Lap Number as the Y-variable and TyreLife as the Predictor-variable due to array dimensionality. We could reshape the array, but we found another package called *statModels* which allows for Poisson Regression. Around half of the data was able to be summarized without further transformation. For FreshTyres, we replaced the values with dummy variables. The remaining data that provided issues were all strings (Team, Driver, DriverNumber, Compound, and TrackStatus) that were transformed into categories and then codified.

Looking solely at the Saudi Arabia 2021 race data for only the pit stops, the average lap number a pit stop was made was 14.833. The variance was 46.567 (over three times as much). This result violates the Poisson assumption of an equal mean-variance value, or equally dispersed values. To counteract the overdispersion, we started building Consul’s Generalized Poisson regression model (https://www.jstor.org/stable/1267389), known as GP-1. Some additional errors occurred running this regression. For the “Compound”, “TrackStatus”, “Gap”, “Cumulative”, “CumulativeLap”,

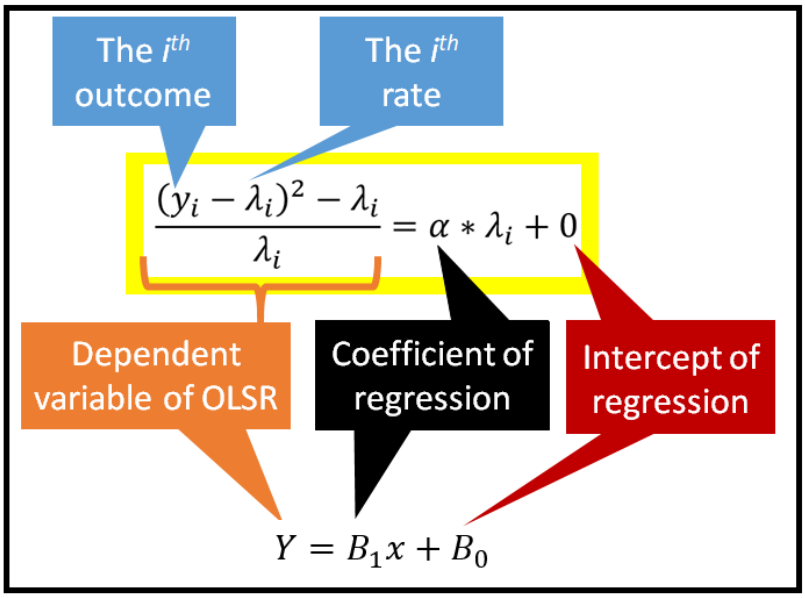
“CumulativePit”, and “PitTime” factors there was an error where the maximum likelihood optimization failed to converge. The factor “PitInTime” had a divide by zero error. The factor “PitOutTime” had a linear algebra singular matrix error.

The next method attempted was Famoye's Restricted Generalized Poisson regression model, known as GP-2 (https://www.tandfonline.com/doi/abs/10.1080/03610929308831089). We tested half of the variables, including those which did not have an issue with the normal Poisson model or Consul’s model, and all resulted in a convergence error.

There were only two options left: a Quasi-Poisson process or a Negative Binomial regression. With the underlying criterion of mean-variance equality being violated, we started with building a Negative Binomial regression analysis. There are two common Negative Binomial formulas. In Joseph Hilbe’s Negative Binomial Regression book (https://www.cambridge.org/core/books/negative-binomial-regression/12D6281A46B9A980DC6021080C9419E7), he states that "NB2 is the standard form of Negative Binomial used to estimate data that are Poisson-over dispersed and is the form of the model which most statisticians understand by Negative Binomial. NB2 is typically the first model we turn to when we discover that a Poisson model is over dispersed." The stat models package has an entire method for the NB2 model: *class statsmodels.genmod.families.family.NegativeBinomial(link=None, alpha=1.0)*

The default value of alpha is 1 which is not necessarily correct. There is a way to estimate the alpha. In Cameron and Trivedi’s book, they propose calculating alpha by using a technique they call auxiliary OLS regression without a constant (https://faculty.econ.ucdavis.edu/faculty/cameron/racd2/).

Image 2 – Formula to Find Alpha in Negative Binomial Regression Model



"We processed the Negative Binomial algorithm using a training and test set. Set up the X and y matrices for the training and testing data sets#Add the λ vector as a new column called 'BB_LAMBDA' to the dataframe of the training data set#add a derived column called 'AUX_OLS_DEP' to the pandas DataFrame. This new column will store the values of the dependent variable of the OLS regression#use patsy to form the model specification for the OLSR#Configure and fit the OLSR model."

The result for the alpha was a -0.06. The actual alpha should be between 0 and 1. We did take the absolute value in and plugged it into a t-test online calculator to determine the significance. It was significant. It also led to a decent predictive result as shown in Table 13. It cannot be trusted, however, due to the manual adjustment of the alpha value.

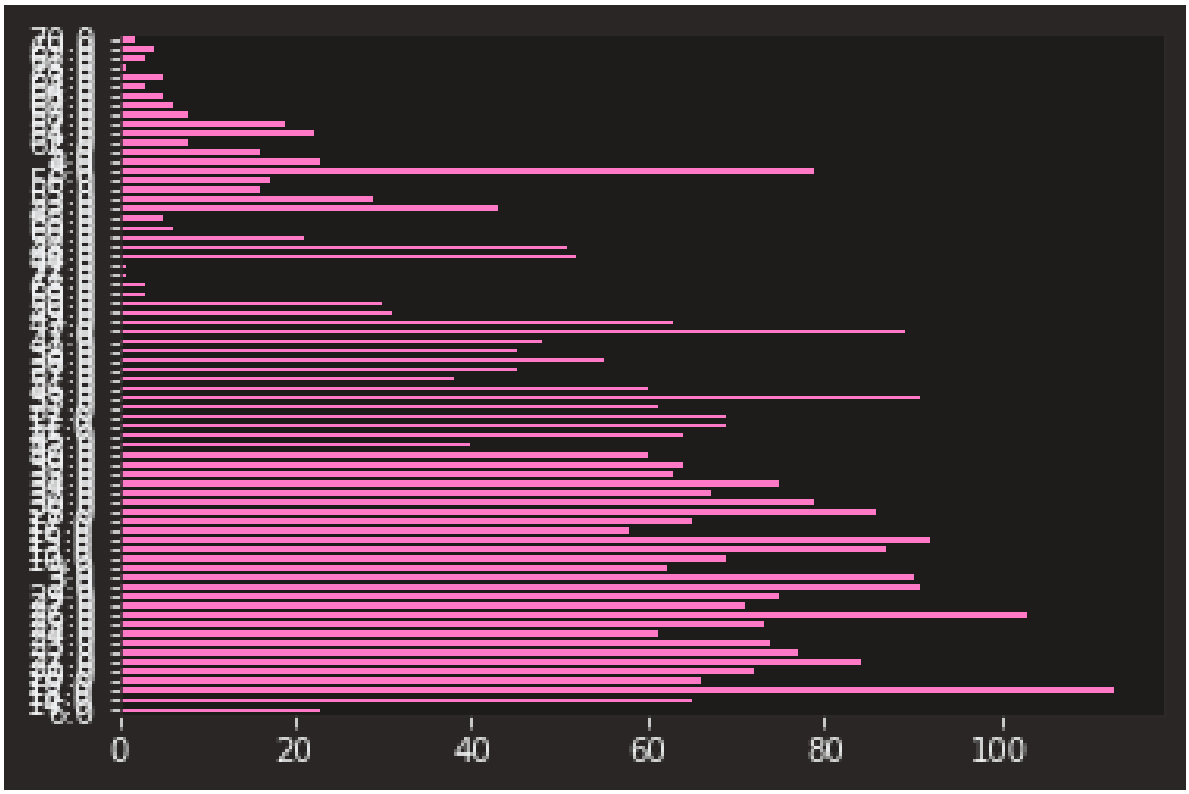
Since Negative Binomial did not work due to the negative alpha, we took a brief glance at a Random Forest regression. The accuracy score for the Random Forest was 80%. There was a zero-division error for the classification report so results such as the F-score were not available.

Table 6 – Random Forest Results

Variable	**Importance**
Sector1TimeSeconds'	9.421%.
CumulativeLap'	8.47%.
Sector2TimeSeconds'	6.959%.
Sector3TimeSeconds'	6.898%.
CumulativePit'	5.435%.
TyreLife'	4.298%.
EndPosition'	2.19%.
TrackStatusCat'	16.64%.
Cumulative'	11.057%.
PitTime'	10.172%.
PitInTimeSeconds'	10.074%.
DriverCat'	1.543%.
Gap'	1.23%.
Points'	1.067%.
Stint'	1.043%.
StartPosition'	0.891%.
TeamCat'	0.881%.
DriverNumberCat'	0.71%.
FreshTyreCat'	0.592%.
CompoundCat'	0.43%.
PitOutTimeSeconds'	0.0%.

To perform the Random Forest, we had to remove the RACE category. We also created a train/test set for prediction purposes. We then re-ran the mean and variance of the lap number. For the Saudi Arabia race, the variance was 46.567 with a mean of 14.833. For the full dataset the variance was 225.903 with a mean of 26.847. This has even more over-dispersion than the original analysis.

Figure 3 – Frequency of Pit Stops



We then looked at the mean and variance of the tire compounds. In the same race, the mean was .479 with a variance of .297, a difference of 60%. Looking at the full dataset, the mean was 2.904 with a variance of 3.01, a difference of -4%. There is still overdispersion although arguably not significant.

After testing the Poisson Regression, Consul's Poisson, Famoye's Poisson, Random Forest, and a Negative Binomial algorithm on a singular dataframe, the next attempt was working with a Quasi-Poisson process. There is no set feature in the Python framework that allows for a Quasi-Poisson model to be ran. The one model online which used a Quasi-Poisson process essentially backdoored R into the Python script, transforming the dataframe into an R dataframe and then running the R script on it. When this was

done to our data, we were able to get it into the R format but “glm” was not recognized as an R command.

At this point, we decided to simply write the dataframe into a CSV and work on it from RStudio. Unfortunately, this meant that data needed replicated manipulation – such as making the Driver or Team name a factor. After cleaning the data, we installed the PSCL package to allow us to run the statistical tests. In R, the normal Poisson AIC was 20872, the Negative Binomial AIC was similar at 20857. All GLMs use the same log-linear mean function ($\log(\mu) = x > \beta$) but make different assumptions about the remaining likelihood.

We then tried some further statistical tests, such as Generalized Estimating Equations through the R Package, GeePack. This did not work due to the model matrix being rank deficient. We then tried the FlexMix package - https://ro.uow.edu.au/cgi/viewcontent.cgi?referer=&httpsredir=1&article=3410&context=commpapers. Flex mix did not work due to NAN Log Likelihood Error which can appear for numerous reasons.

The NLME package was considered as well but removed from consideration due to the Gaussian distribution requirement (https://cran.r-project.org/web/packages/nlme/nlme.pdf).

Our next, and final attempt was a Bayesian analysis of Poisson data. The first step was to run a simulation for the normal model. We did this via the Stan package using the arguments provided in the example we followed, including family = gaussian, prior_intercept, prior_aux, prior. Prior should either be the default prior or specified with precedent analysis. Prior_intercept is the prior distribution for the intercept after centering all the predictors - this also has a default value. Prior_aux is not always applicable but in the Gaussian family it refers to the “sigma” or standard error deviation. For Negative Binomial model it refers to “reciprocal dispersion”. Poisson models do not have auxiliary parameters. We started with two chains of 500 iterations. That led to sampling errors where the estimated Bayesian Fraction of Missing Information was low. With the errors, we updated the chains to 4 and iterations to 1000. Even with four chains, the estimated Bayesian Fraction of Missing Information was low. Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be unreliable. Tail

Effective Samples Size (ESS) is too low, indicating posterior variances and tail quantiles may be unreliable. We then increased it to 4 chains and 2*1000 iterations. This took over 15 hours to run and led to several errors.

We did continue to proceed as if we received a useful, or at the least a working, model. So, our next step was to create a posterior prediction using the normal simulation. We plotted the results as a histogram to compare against the original histogram showing the actual lap number that pit stops were made.

Since we know the model is right skewed, we can assume that the output would not accurately depict the probability with the normal distribution histogram output. Our next step was to run the Bayesian prior simulation with the Poisson package instead of a Gaussian family model. At this point, we would summarize the model to determine the specification of the informative priors. From this step we could add a few lines of code that visualizes n number of prior plausible models dependent upon a specified factor.

Now we could begin the simulation of the posterior for each state by calling the prior model and setting the prior to false. We could then review the posterior simulation performed by looking at the MCMC trace, density, and autocorrelation plots. Another good visualization to run would be another posterior predictive check. Ideally the newly created histograms would follow the actual histogram. If it does not, then we would need to re-evaluate the arguments or the model itself.

Assuming the model is somewhat accurate, we could analyze the results. We could do that analysis visually or by running a confident interval analysis in TidyVerse for the variables. Further detailed analysis could be done by filtering variables to a specific value, such as Driver, Team, or Race. The final step would then evaluate the model's accuracy. We would do this by simulating the posterior predictions, plotting them, summarizing them, and see what how many standard deviations the pit stop lap number is from the posterior mean prediction. To truly exhibit data science understanding for the thesis as well as model accuracy, one should then implement cross validation.

Results

Here is a snapshot of the dataframe for one race, with the results dataframe merged, against the total laps dataframe before the results data is merged. The total dataframe incorporates the RACE name as well as the Year. The Abu Dhabi dataframe incorporates Grid Position, Position, Points and Total Time. The differences are highlighted in Table 7.

Table 7 – Comparing Dataframe of One Race Against The Total Dataframe

Abu Dhabi 2019	
Index	0
Time	0 days 00:35:15.764000
DriverNumber	44
LapTime	NA
LapNumber	1
PitOutTime	0 days 00:00:04.732000
PitInTime	NA
Sector1Time	NA
Sector2Time	0 days 00:00:43.430000
Sector3Time	0 days 00:00:40.714000
Sector1SessionTime	NA
Sector2SessionTime	0 days 00:34:35.110000
Sector3SessionTime	0 days 00:35:15.912000
SpeedI1	279
SpeedI2	289
SpeedFL	219
SpeedST	293
IsPersonalBest	FALSE
Compound	MEDIUM
TyreLife	4
FreshTyre	FALSE
Stint	1
LapStartTime	0 days 00:33:31.194000
Team	Mercedes
Driver	HAM
TrackStatus	2
IsAccurate	FALSE
LapStartDate	43800.55106
GridPosition	1
Position	1
Points	26
TotalTime	0 days 00:01:44.570000

Total	
Race	Australia
Index	0
Time	0 days 00:08:44.363000
DriverNumber	5
LapTime	NA
LapNumber	1
PitOutTime	0 days 00:00:04.103000
PitInTime	NA
Sector1Time	NA
Sector2Time	0 days 00:00:24.142000
Sector3Time	0 days 00:00:36.352000
Sector1SessionTime	NA
Sector2SessionTime	0 days 00:08:08.653000
Sector3SessionTime	0 days 00:08:44.559000
SpeedI1	275
SpeedI2	290
SpeedFL	286
SpeedST	241
IsPersonalBest	FALSE
Compound	NA
TyreLife	NA
FreshTyre	NA
Stint	NA
LapStartTime	0 days 00:07:07.988000
Team	Ferrari
Driver	VET
TrackStatus	1
IsAccurate	FALSE
LapStartDate	NA
Year	2018

In total, the Abu Dhabi dataframe has a dimensionality of 1075 x 32, while the laps dataframe has a dimensionality of 110,125 x 30.

After working through and merging the data across all the years and between dataframes, we ended up with a final dataframe which has a dimensionality of 3416 x 19.

Table 8 – Variables in Final Dataframe			
Index	116	261	315
DriverNumberCat	32	1	27
LapNumber	6	14	18
CompoundCat	5	6	6
TyreLife	5	13	20
FreshTyre	TRUE	TRUE	FALSE
Stint	1	1	1
TeamCat	13	14	5
Year	2018	2018	2018
RACE	Australia	Australia	Australia
DriverCat	5	7	23
TrackStatusCat	1	0	0
GridPosition	17	20	2
Position	19	18	3
Points	0	0	15
TotalTime	581.634	1328.885	1627.51
GapSeconds	6.932	10.382	13.192
RaceCat	1	1	1
FreshTyreCat	1	1	0

In this final dataframe, we selected only those laps which had a pit stop. We transformed the categorical variables into factors and Boolean variables into integers. At this point, we had a workable dataframe and began with basic analysis.

In the “pit_in_time_lap_number” dataframe, we looked at which lap number teams and drivers took a pit stop. In this Table 9, team codified as #5, with the driver codified as 23, took 1 pit stop at lap 18 of the race.

Table 9 – Variables in Pit Stop Dataframe	
Index	0
Year	2018
Race	Australia
Team	5
Driver	23
Mean	18
Min	18
Max	18
NumberOfStops	1

We then visualized the frequency of pit stops and found that predominantly there are only 1 or 2 pit stops per driver per race. We also looked at the normalized data.

While frequency of pit stops is good to look at to get a good understanding of the data, why a pit stop is made is dependent on how the tires are handling. Similar to the lap number dataframe and visualizations in Table 9, we created a tire life dataframe and visualizations.

Table 10 – Variables in Pit Stop Dataframe 2nd Showing	
Index	15
Year	2018
Race	Australia
Team	13
Driver	16
TyreLife	[19.0, 7.0]
Mean	23.5
Min	20
Max	27
NumberOfStops	2

We hand-picked this row of the dataframe to show that there were 2 stops made, one at lap 20 and one at lap 27. That corresponds with a tire life of 19 laps for the first pit stop and 7 laps for the second pit stop. It also shows the mean, minimum, and maximum values regarding the pit stop lap numbers.

With the dataframe prepared and basic analysis done, Poisson modelling could be attempted. The results for the normal Poisson regression are in Table 11:

Table 11 – Poisson Regression Results

Variable	Coef	Std Erro	Z	P	0.025	0.975
TyreLife	0.1357	0.002	61.849	0	0.131	0.14
Cumulative	0.0006	8.62E-06	74.35	0	0.001	0.001
CumulativeLap	0.001	1.31E-05	73.849	0	0.001	0.001
CumulativePit	0.0015	2.16E-05	68.217	0	0.001	0.002
Sector1TimeSeconds	0.0372	0.001	65.751	0	0.036	0.038
Sector2TimeSeconds	0.05	0.001	68.866	0	0.049	0.051
Sector3TimeSeconds	0.0371	0.001	67.139	0	0.036	0.038
PitTime	0.0025	4.29E-05	58.611	0	0.002	0.003
PitOutTimeSeconds	0	0	nan	nan	0	0
PitInTimeSeconds	0.0004	5.19E-06	76.482	0	0	0
Stint	1.0934	0.015	72.19	0	1.064	1.123
Gap	0.0027	0	13.765	0	0.002	0.003
Fresh Tyre - FALSE	2.8084	0.05	56.027	0	2.71	2.907
Fresh Tyre - True	2.5713	0.056	45.564	0	2.461	2.682
TeamCat	0.3661	0.007	54.139	0	0.353	0.379
DriverCat	0.1939	0.003	60.429	0	0.188	0.2
DriverNumberCat	0.1997	0.003	59.707	0	0.193	0.206
TrackStatusCat	0.5792	0.01	55.519	0	0.559	0.6

Moving on to Consul's Poisson, the results are in Table 12:

Table 12 – Consul's Poisson Result

Variable	coef	std err	z	P>|z|	[0.025	0.975]
Stint	1.0922	0.031	35.145	0	1.031	1.153
Stint_alpha	1.2417	0.216	5.747	0	0.818	1.665
TyreLife	0.1365	0.005	25.987	0	0.126	0.147
TyreLife_alpha	1.987	0.281	7.069	0	1.436	2.538
Sector1TimeSeconds	0.0374	0.001	28.658	0	0.035	0.04
Sector1TimeSeconds_alpha	1.6397	0.271	6.06	0	1.109	2.17
Sector2TimeSeconds	0.0506	0.001	34.544	0	0.048	0.053
Sector2TimeSeconds_alpha	1.123	0.1	5.339	0	0.711	1.535
Sector3TimeSeconds	0.0374	0.001	30.331	0	0.035	0.04
Sector3TimeSeconds_alpha	1.4451	0.255	5.665	0	0.945	1.945
DriverCat	0.1984	0.008	25.492	0	0.183	0.214
DriverCat_alpha	1.8954	0.291	6.504	0	1.324	2.467
DriverNumberCat	0.2015	0.008	24.005	0	0.185	0.218
DriverNumberCat_alpha	2.0324	0.312	6.513	0	1.421	2.644
TeamCat	0.3762	0.017	21.984	0	0.343	0.41
TeamCat_alpha	2.2141	0.334	6.636	0	1.56	2.868
FreshTyreFalse	2.7766	0.073	38.12	0	2.634	2.919
FreshTyreTrue	2.6103	0.079	33.005	0	2.455	2.765
FreshTyre_alpha	0.4366	0.137	3.191	0.001	0.168	0.705

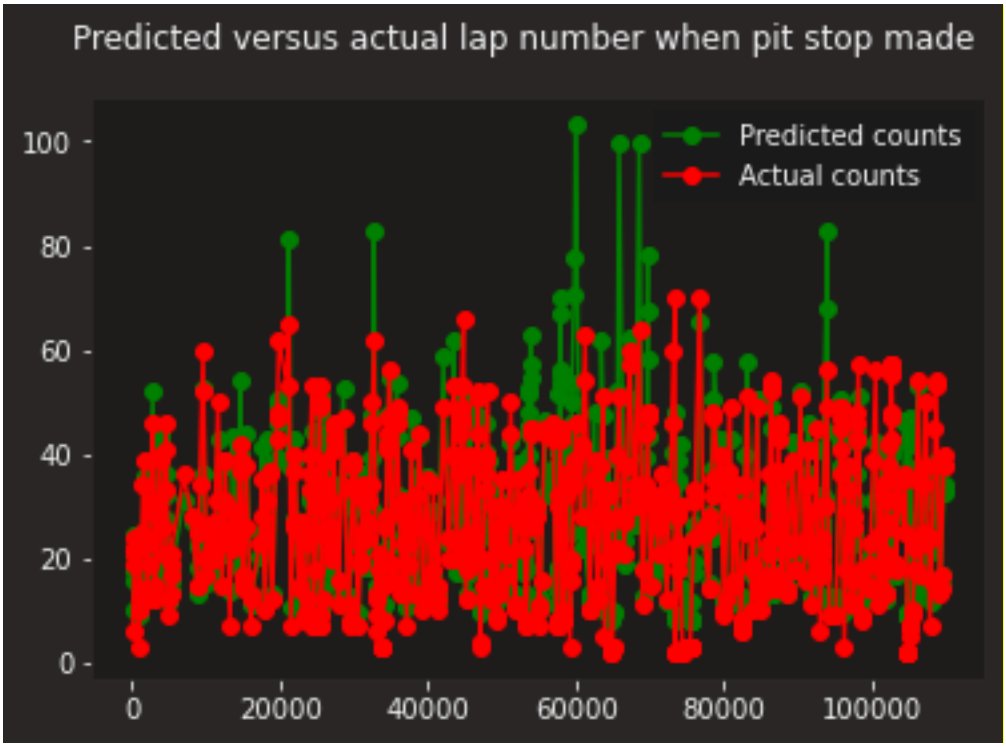
Famoye's Poisson did not provide any results, only errors.

The next step was a Negative Binomial. Results shown in Table 13:

Table 13 – Negative Binomial Regression Results						
Variable	**coef**	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	4.9346	26.488	0.186	0.852	-46.982	56.851
DriverNumberCat	0.0012	0.016	0.074	0.941	-0.03	0.032
CompoundCat	-0.0036	0.214	-0.017	0.987	-0.423	0.416
TyreLife	0.0001	0.033	0.004	0.996	-0.064	0.065
FreshTyreCat	-0.0065	0.258	-0.025	0.98	-0.512	0.499
Stint	-0.0529	0.58	-0.091	0.927	-1.19	1.085
Gap	-4.25E-05	0.005	-0.008	0.993	-0.01	0.01
TeamCat	0.0005	0.045	0.011	0.991	-0.087	0.088
DriverCat	0.0005	0.021	0.025	0.98	-0.041	0.042
TrackStatusCat	0.0262	0.11	0.238	0.812	-0.19	0.243
StartPosition	-0.0038	0.022	-0.168	0.867	-0.048	0.04
EndPosition	-0.0009	0.03	-0.031	0.975	-0.06	0.058
Points	0.0003	0.024	0.013	0.989	-0.046	0.047
PitTime	-0.001	0.007	-0.146	0.884	-0.015	0.013
PitInTimeSeconds	-0.0008	0.007	-0.115	0.909	-0.014	0.012
PitOutTimeSeconds	-2.58E-13	1.38E-12	-0.186	0.852	-2.97E-12	2.46E-12
Sector1TimeSeconds	-0.0009	0.01	-0.09	0.928	-0.021	0.019
Sector2TimeSeconds	-0.0024	0.023	-0.105	0.917	-0.047	0.043
Sector3TimeSeconds	0.002	0.017	0.119	0.905	-0.032	0.036
Cumulative	0.0007	0.005	0.148	0.882	-0.009	0.01
CumulativeLap	0.0004	0.002	0.18	0.857	-0.004	0.004
CumulativePit	0.0003	0.003	0.122	0.903	-0.005	0.006

The issue with the Negative Binomial was a negative alpha value, which is mathematically impossible to implement in further analysis. Nonetheless, we did make the negative alpha value an absolute value to run the predicted vs actual analysis, resulting in Figure 4.

Figure 4 – Predicted vs Actual Visualization for Negative Binomial



Moving on from the Poisson family analysis, we tried Random Forest next. The results for the importance of variables in a Random Forest model are shown in Table 14:

Table 14 – Random Forest Variable Importance

Variable	Importance
Sector1TimeSeconds'	9.421%.
CumulativeLap'	8.47%.
Sector2TimeSeconds'	6.959%.
Sector3TimeSeconds'	6.898%.
CumulativePit'	5.435%.
TyreLife'	4.298%.
EndPosition'	2.19%.
TrackStatusCat'	16.64%.
Cumulative'	11.057%.
PitTime'	10.172%.
PitInTimeSeconds'	10.074%.
DriverCat'	1.543%.
Gap'	1.23%.
Points'	1.067%.
Stint'	1.043%.
StartPosition'	0.891%.
TeamCat'	0.881%.
DriverNumberCat'	0.71%.
FreshTyreCat'	0.592%.
CompoundCat'	0.43%.
PitOutTimeSeconds'	0.0%.

The accuracy score of the RF model was 42.6% so that was discarded. We could have attempted an XGBoost and that should be looked at in the future.

From Python, we then switched over to analysis in R. We first ran a regular Poisson regression as we did in Python. It resulted in a dispersion parameter to be 1 with null deviance of 31,025.8 on 3414 degrees of freedom, residual deviance of 3762.5 on 3250 degrees of freedom, and 20,872 AIC score, on 5 of Fisher Scoring iterations.

We then ran a Quasi-Poisson analysis in R which showed a dispersion parameter to be 1.10401 with null deviance of 31,025.8 on 3414 degrees of freedom, residual deviance of 3762.5 on 3250 degrees of freedom, and a non-applicable AIC score, on 5 of Fisher Scoring iterations.

We then ran a Negative Binomial analysis which showed a dispersion parameter to be 1, null deviance of 28641.5 on 3414 degrees of freedom and residual deviance of 6452.4 on 3250 degrees of freedom with an AIC of 20857 on 1 iteration of Fisher Scoring. The theta was 295.3 with a standard error of 68.5. A warning occurred while fitting theta that the alternation limit was reached. The 2x log-likelihood was -20525.33.

We could not do a hurdle or zip analysis due to non-zero minimum counts.

The GeeGLM was rank deficient.

FlexMix had a non-applicable log-likelihood.

We then attempted a Bayesian Poisson process. We started with two chains of 500 iterations. That lead to sampling errors where the estimated Bayesian Fraction of Missing Information was low. Even though this attempt appeared to be unsuccessful, we implemented a 10-fold cross validation resampling method and increased the number of chains and iterations. There were two errors – the maximum tree depth was exceeded and the chains did not mix, which was indicated by the lowest R-hat number of 4.22.

Discussion

In this paper, we attempted to predict when a driver or team will make a pit stop. We utilized Poisson and related methods. We included all races over a five-year period. Our analysis did not work at any step of the process. Another step which could be taken is making a Bayesian Negative Binomial regression model. One thing that should be considered is normalized the course data as the wide disparity in course lap numbers can alter the analysis.

Further analysis should include a boosted Random Forest or look towards non-Poisson analysis. If no headway can be made with non-Poisson models, then reducing the data to only specific years or specific courses would be advised. Ideally further analysis should include a way to incorporate the weather data into our other dataframes.

If none of the prior steps lead to useful results, perhaps a change in question from predicting when a pit stop would be made to what tires a driver will put on the car would be appropriate.

References

Asher, R (2023, February 28). *F1 cost Cap: What is it and how does it work?* Motorsport.com. Retrieved March 8, 2023, from https://us.motorsport.com/f1/news/f1-cost-cap-what-is-it-how-it-works/10379800/#:~:text=The%20F1%20cost%20cap%20limits,cap%20to%20be%20%24175m

Bell, A., Smith, J., Sabel, C.E., & Jones, K. (2016). Formula for success: Multilevel modelling of Formula One Driver and Constructor performance, 1950–2014. *Journal of Quantitative Analysis in Sports, 12*, 112 - 99.

Bogomolov, A., Nevezhin, V., & Zvyagintseva, E.P. (2019). Random Event and Probability in Mathematical Modeling of Economic Processes. *Proceedings of the International Scientific Conference "Far East Con" (ISCFEC 2018)*.

Bruce, B. de M. (2010). In *Predictioneer's game: Using the logic of brazen self-interest to see and shape the future*. essay, Random House Trade Paperbacks.

Chimka, J. R., & Talafuse, T. P. (2016). Poisson regression analysis of additional strokes assessed at golf. International Journal of Sports Science & Coaching, 11(4), 619–622. https://doi.org/10.1177/1747954116654785

Collantine, K. (2016, March 22). *2016 Australian GP tyre strategies and pit stops - F1 fanatic*. RaceFans. Retrieved March 7, 2023, from https://www.racefans.net/2016/03/20/2016-australian-grand-prix-tyre-strategies-and-pit-stops-2/

Ding, A. (n.d.). *Formula 1: Using the fastf1 python API to visualize race pace*. Formula 1: Using the FastF1 Python API to visualize race pace | Home. Retrieved March 7, 2023, from https://allending.ca/Formula1_RacePace_FastF1

Dixit, A. K., & Nalebuff, B. (2010). *The art of strategy: A game theorist's guide to success in Business and Life*. Norton.

Dolmeta, P., Argiento, R., & Montagna, S. (2021). Bayesian GARCH modeling of functional sports data. *Statistical Methods & Applications*.

The Ergast Developer API is an experimental web service which provides a historical record of motor racing data for non-commercial purposes. Ergast developer API. (n.d.). Retrieved March 7, 2023, from http://ergast.com/mrd/

F1® Tires. Pirelli. (n.d.). Retrieved March 8, 2023, from https://www.pirelli.com/tires/en-us/motorsport/f1/tires

F1Metrics. (2015, September 3). *Building a race simulator*. f1metrics. Retrieved March 7, 2023, from https://f1metrics.wordpress.com/2014/10/03/building-a-race-simulator/

Insights, T. (n.d.). *Tracing insights*. Substack. Retrieved March 7, 2023, from https://tracinginsights.substack.com/

Jasper. (2021, October 13). *Formula 1 data analysis tutorial - 2021 Russian GP: "to box, or not to box?"*. Medium. Retrieved March 7, 2023, from https://medium.com/towards-formula-1-analysis/formula-1-data-analysis-tutorial-2021-russian-gp-to-box-or-not-to-box-da6399bd4a39

Jasper. (2022, March 31). *How to analyze formula 1 telemetry in 2022-A python tutorial*. Medium. Retrieved March 7, 2023, from https://medium.com/towards-formula-1-analysis/how-to-analyze-formula-1-telemetry-in-2022-a-python-tutorial-309ced4b8992

Jasper. (2022, March 7). *Analyzing formula 1 race pace using Python*. Medium. Retrieved March 7, 2023, from https://medium.com/towards-formula-1-analysis/analyzing-formula-1-race-pace-using-python-c053d80f48ff

Johnson, Alicia A., et al. "Poisson & Negative Binomial Regression." *Bayes Rules!: An Introduction to Applied Bayesian Modeling*, CRC Press, Boca Raton, FL, 2022.

Kesteren, E.V., & Bergkamp, T.L. (2022). Bayesian Analysis of Formula One Race Results: Disentangling Driver Skill and Constructor Advantage.

Lee, J. (2022). A Finite-State Stationary Process with Long-Range Dependence and Fractional Multinomial Distribution. *Fractal and Fractional*.

Maiza, A. (2020, October 30). *Reinforcement learning for formula 1 race strategy*. Medium. Retrieved March 7, 2023, from https://towardsdatascience.com/reinforcement-learning-for-formula-1-race-strategy-7f29c966472a

Medland, C. (2022, August 28). *Strategy guide: What are the possible race strategies for the 2022 Belgian Grand Prix?: Formula 1®*. Formula 1. Retrieved March 7, 2023, from https://www.formula1.com/en/latest/article.strategy-guide-what-are-the-possible-race-strategies-for-the-2022-belgian.3rU2eOmR5tEMNrpukMd5Ar.html

Mercedes-AMG Petronas Formula One Team. (2022, November 17). *Mercedes feature: Data and electronics in F1 - explained*. Silver Arrows Net. Retrieved March 8, 2023, from https://www.silverarrows.net/news/mercedes-feature-data-and-electronics-in-f1-explained/

Nigro, V. (2020, June 11). *Formula 1 race predictor*. Medium. Retrieved March 7, 2023, from https://towardsdatascience.com/formula-1-race-predictor-5d4bfae887da

Pandey, P. (2022, March 22). *Accessing formula -1 race's 🏎 historical data using Python*. Medium. Retrieved March 7, 2023, from https://pandeyparul.medium.com/accessing-formula-1-races-historical-data-using-python-b7c80e544f50

Predicting baseball. Sports Management Degrees. (n.d.). Retrieved March 7, 2023, from https://www.sports-management-degrees.com/baseball/

Pretorius, L. (2022, January 12). *Flags in F1 explained*. One Stop Racing. Retrieved March 8, 2023, from https://onestopracing.com/flags-in-f1-explained/#:~:text=Each%20flag%20used%20in%20F1,means%20a%20driver%20is%20disqualified

R/f1technical. reddit. (n.d.). Retrieved March 7, 2023, from https://www.reddit.com/r/F1Technical/

Santos-Fernandez, E., Wu, P. & Mengersen, K. (2019). Bayesian statistics meets sports: a comprehensive review. *Journal of Quantitative Analysis in Sports*, *15*(4), 289-312. https://doi.org/10.1515/jqas-2018-0106

Schaefer, P. (n.d.). *TheOehrly/Fast-F1: FastF1 is a python package for accessing and analyzing formula 1 results, schedules, timing data and telemetry*. GitHub. Retrieved March 7, 2023, from https://github.com/theOehrly/Fast-F1

Spiliopoulos, L. (2016). Randomization and Serial Dependence in Professional Tennis Matches: Do Strategic Considerations, Player Rankings and Match Characteristics Matter? *Behavioral & Experimental Economics eJournal*.

Stavropoulos, V. (2018, May 21). *A thorough analysis of The pit stop strategy in formula 1*. Statathlon. Retrieved March 7, 2023, from https://statathlon.com/analysis-of-the-pit-stop-strategy-in-f1/

Tulabandhula, T., & Rudin, C. (2014). Tire Changes, Fresh Air, and Yellow Flags: Challenges in Predictive Analytics for Professional Racing. *Big data, 2 2*, 97-112

Wunderlich, F., Seck, A., & Memmert, D. (2021). The influence of randomness on goals in football decreases over time. An empirical analysis of randomness involved in goal scoring in the English Premier League. *Journal of Sports Sciences, 39*, 2322 - 2337.

Python Packages

```
import fastf1
import numpy as np
import pandas as pd

import fastf1 as ff1
from fastf1.core import Laps
from fastf1 import utils
from fastf1 import plotting

import matplotlib
import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection
from matplotlib import cm

from timple.timedelta import strftimedelta
import os
```

R Packages

```
library(tidyverse)
library(readxl)
library(knitr)
library(bookdown)
library(reticulate)

#for shiny
library(shiny)
library(plotly)
library(readxl)
library(tidyverse)
library(magrittr)
library(jpeg)
library(officer)

#for analysis
library(lubridate)
library(pscl)
library(lmtest)
library(geepack)
library(flexmix)
library(lme4)
library(bpr)
library(bayesplot)
library(tidybayes)
library(broom.mixed)
library(rstanarm)
library(bayesrules)
```