La Vuelta

Laura Cebollero Ruiz, Alexandre Rodríguez Garau

4th January, 2019

1 Introduction

In this project we intend to predict the duration of the different stages of the cycling race *La Vuelta* using the information provided in the file `Vuelta0.mtp`.

```
library("readxl")
library("knitr")
library("ggplot2")
library("reshape2")
library("MASS")
library("DataExplorer")
```

We have exported the data to an extension that R can read: `.csv`.

```
data<- read.csv("Vuelta01.csv", sep = ';', header = TRUE, dec=",")
```

2 Data exploration

To predict the length of the stages we will use a set that contains 14 explanatory variables and a response variable called `ForecastedTime`. Let's take a look at the summary of the variables:

```
kable(summary(data[,1:6]))
```

Time	Distance	HeightIncr	AccumIncr	portsE	ports1
Min. :171.6	Min. :111.0	Min. :-940.0	Min. : 190	Min. :0.0000	Min. :0.000
1st Qu.:258.1	1st Qu.:167.2	1st Qu.:-180.0	1st Qu.: 510	1st Qu.:0.0000	1st Qu.:0.000
Median :301.4	Median :196.0	Median : 70.0	Median :1350	Median :0.0000	Median :0.000
Mean :302.6	Mean :193.0	Mean : 227.9	Mean :1477	Mean :0.1524	Mean :0.581
3rd Qu.:345.8	3rd Qu.:219.5	3rd Qu.: 540.0	3rd Qu.:2300	3rd Qu.:0.0000	3rd Qu.:1.000
Max. :439.7	Max. :264.0	Max. :2310.0	Max. :4216	Max. :2.0000	Max. :3.000

```
kable(summary(data[,6:11]))
```

ports1	ports2	ports3	year	week	bef_mount
Min. :0.000	Min. :0.0000	Min. :0.0000	Min. :1.000	Min. :1.000	Min. :0.0000
1st Qu.:0.000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:2.000	1st Qu.:1.000	1st Qu.:0.0000
Median :0.000	Median :0.0000	Median :1.0000	Median :4.000	Median :2.000	Median :0.0000
Mean :0.581	Mean :0.4857	Mean :0.8476	Mean :3.619	Mean :2.048	Mean :0.2952
3rd Qu.:1.000	3rd Qu.:1.0000	3rd Qu.:2.0000	3rd Qu.:5.000	3rd Qu.:3.000	3rd Qu.:1.0000
Max. :3.000	Max. :3.0000	Max. :4.0000	Max. :6.000	Max. :3.000	Max. :1.0000

```
kable(summary(data[,12:16]))
```

aft_mount	bef_tt	aft_tt	last	ForecastedTime
Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. :0.00000	Min. :180.0
1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.00000	1st Qu.:263.4
Median :0.0000	Median :0.0000	Median :0.0000	Median :0.00000	Median :307.8
Mean :0.2952	Mean :0.1333	Mean :0.1905	Mean :0.04762	Mean :306.7
3rd Qu.:1.0000	3rd Qu.:0.0000	3rd Qu.:0.0000	3rd Qu.:0.00000	3rd Qu.:348.2
Max. :1.0000	Max. :1.0000	Max. :1.0000	Max. :1.00000	Max. :437.5

In the previous tables we can see a short summary of the variables. **`ports1`**, **`ports2`** and **`ports3`** indicate the number of mountain sections in a given stage and their category (1,2 or 3). **`year`** corresponds to the year in which this data was recorded (1 to 6). **`Week`** is the week of the race in wich the stage takes place (1 to 3). The variables **`bef_mount`** and **`aft_mount`** tell us whether a stage took place before or after a mountain stage, respectively. Similarly, the variables **`bef_tt`** and **`aft_tt`** indicate if a stage took place before or after a time trial stage. Finally, the variable **`last`** tells us if that stage was the last of the whole race.

By the looks of the summary we can't seem to find any outliers or abnormal values. Most of the explanatory variables range from 0 to very low values and are natural numbers. The only continuous variables are **`ForecastedTime`**, **`Distance`**, **`HeightInc`** and **`AccumIncr`**. This makes sense because the first variable indicates time and time is a continuous variable and the rest indicate distance which can also be continous.

The variable **`Distance`** seems to be quite balanced with a mean of 193 and min and max values of 111 and 264 respectively. **`HeightIncr`**, however, is the only variable that presents negative values and has very high variance. Its minimum value is -940 and the maximum value is 2310.

By taking a closer look at the data we detect some abnormal values in the **`last`** variable: Since this variable indicates if a stage was the last of the whole race, then there should as many stages with this value equal to 1 as years of data have been recorded. Since the recorded stages are from 6 different years then there should be 6 rows, but instead there are only 5, meaning that there is at least 1 missing stage.

```r
data[data$last == 1,]$year
```

```
## [1] 6 5 4 2 1
```

If we look at the data we can easily see that the missing value belongs to the year 3. However, this will probably not greatly affect our predicting power. There is something to be said about the variables **`year`**, **`week`**, **`bef_mount`**, **`aft_mount`**, **`bef_tt`**, **`aft_tt`** and **`last`**: all of these variables are categorical and indicate the group or category a row belongs to. For this, we should transform these variables into factors.

```r
data$year      <- as.factor(data$year)
data$week      <- as.factor(data$week)
data$bef_mount <- as.factor(data$bef_mount)
data$aft_mount <- as.factor(data$aft_mount)
data$aft_tt    <- as.factor(data$aft_tt)
data$bef_tt    <- as.factor(data$bef_tt)
data$last      <- as.factor(data$last)
```

Now that we have transformed the categorical columns to factors we should take another look at the summary of these variables:

```
kable(summary(data[,9:15]))
```

year	week	bef_mount	aft_mount	bef_tt	aft_tt	last
1:15	1:33	0:74	0:74	0:91	0:85	0:100
2:18	2:34	1:31	1:31	1:14	1:20	1: 5
3:17	3:38	NA	NA	NA	NA	NA
4:17	NA	NA	NA	NA	NA	NA
5:18	NA	NA	NA	NA	NA	NA
6:20	NA	NA	NA	NA	NA	NA

We can se that for the variables **`year`** and **`week`** the variables are quite balanced, each group contains a very similar amount of observations. The rest, however, are more unbalanced. `bef_mount` has more than twice 0s than 1s, which means there are twice the stages that do not precede a mountain stage. The same happens with `aft_mount`, which makes sense for it means there are only 31 stages after a mountain stage. Since they are the same numbers as the `bef_mount`, this tells us that the mountain stages are always in the middle and are not the first or last stages.

In `bef_tt` there are 9 times more stages **not** preceding a time trial stage than does preceding it. There are 4 times less after trial stages. And finally, as mentioned before, there are only 5 last stages and 100 that are normal ones.

Let's finally take a closer look to the numerical attributes:

```
num <- 6:8; num <- append(num, 16)
kable(summary(data[,1:5]))
```

Time	Distance	HeightIncr	AccumIncr	portsE
Min. :171.6	Min. :111.0	Min. :-940.0	Min. : 190	Min. :0.0000
1st Qu.:258.1	1st Qu.:167.2	1st Qu.:-180.0	1st Qu.: 510	1st Qu.:0.0000
Median :301.4	Median :196.0	Median : 70.0	Median :1350	Median :0.0000
Mean :302.6	Mean :193.0	Mean : 227.9	Mean :1477	Mean :0.1524
3rd Qu.:345.8	3rd Qu.:219.5	3rd Qu.: 540.0	3rd Qu.:2300	3rd Qu.:0.0000
Max. :439.7	Max. :264.0	Max. :2310.0	Max. :4216	Max. :2.0000

```
kable(summary(data[,num]))
```

ports1	ports2	ports3	ForecastedTime
Min. :0.000	Min. :0.0000	Min. :0.0000	Min. :180.0
1st Qu.:0.000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:263.4
Median :0.000	Median :0.0000	Median :1.0000	Median :307.8
Mean :0.581	Mean :0.4857	Mean :0.8476	Mean :306.7
3rd Qu.:1.000	3rd Qu.:1.0000	3rd Qu.:2.0000	3rd Qu.:348.2
Max. :3.000	Max. :3.0000	Max. :4.0000	Max. :437.5

We can see that the mean and median in Time are pretty close, which means that there is not an outlier that drastically decreases or increases the mean. The same happens with Distance, AccumIncr and ForacastedTime.

However, HeightIncr has a low Median but a high mean, which means that in general there is not a stable increase of height but some stages have a high increase of it. And we can see that there is at least one stage that consists of a decrease of almost 1km, whereas there is at least one stage of an increase of over 2km.

As for the ports, although it is a numerical variable, it actually uses integer values and natural ones should not allowed. We contemplated using the number of ports as a categorical variable. For example, in portsE we can see there are between 0 and 2 mountain passes:

```
table(data$portsE)
```

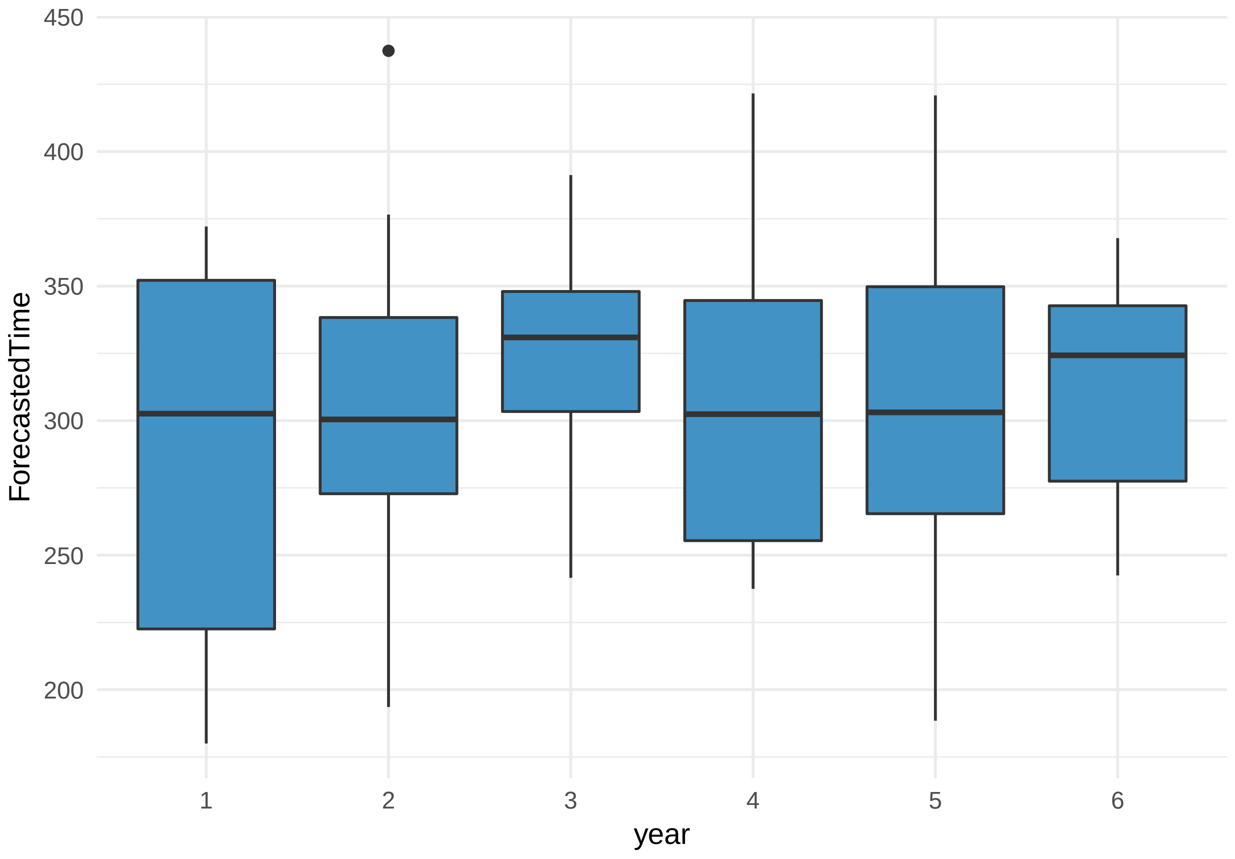
```
##
##  0  1  2
## 91 12  2
```

However, by enforcing said restriction changing the variable as a categorical one, we would not be allowing new categorical variables like using 5 mountain passes in stage port3. Thus, meaning we would be restricted to a maximum of 3 in ports1 and ports2, 2 mountain passes on portsE, and a maximum of 4 mountain passes in ports3, which would not be correct, for maybe in the future they may be a case where there are 3 mountain passes in portsE, where there is still no case as seen in the table above.

3 Data exploration visualization

Now, in order to find outliers and to see how the data behaves we will plot some boxplots. First of all we would like to see if the forecasted times are different depending on the year in which the race took place.

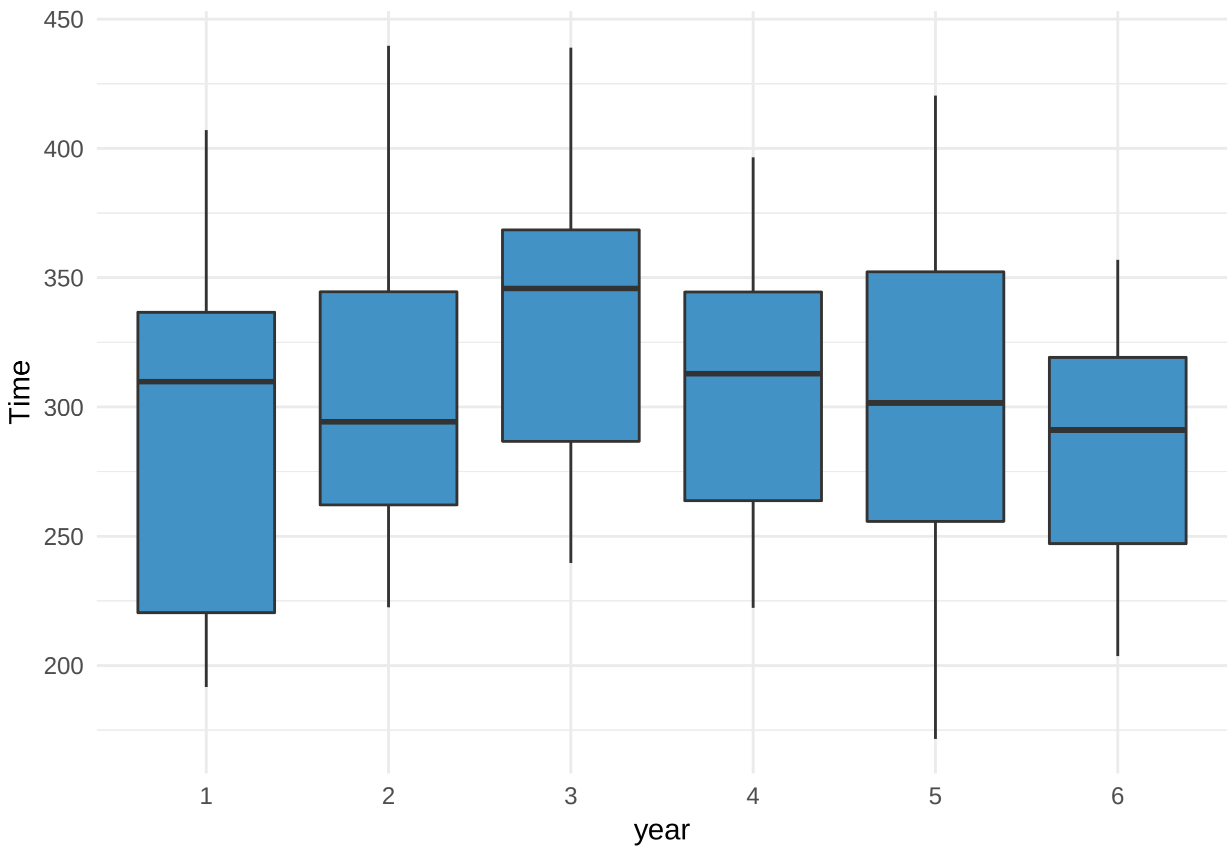
```
ggplot(data = data) +
  aes(x = year, y = ForecastedTime) +
  geom_boxplot(fill = "#4292c6") +
  theme_minimal()
```



At first glance we see that the medians are really similar among all years, excepting year 3 and year 6, which are about 25 units higher. The interquartile ranges for every year are different. Even though the Q3 percentile is very similar for all years, the Q1 percentile varies from 225 to 300. It is important to say that there appears to be an outlier in the second year.

Let's take a look at real Times:

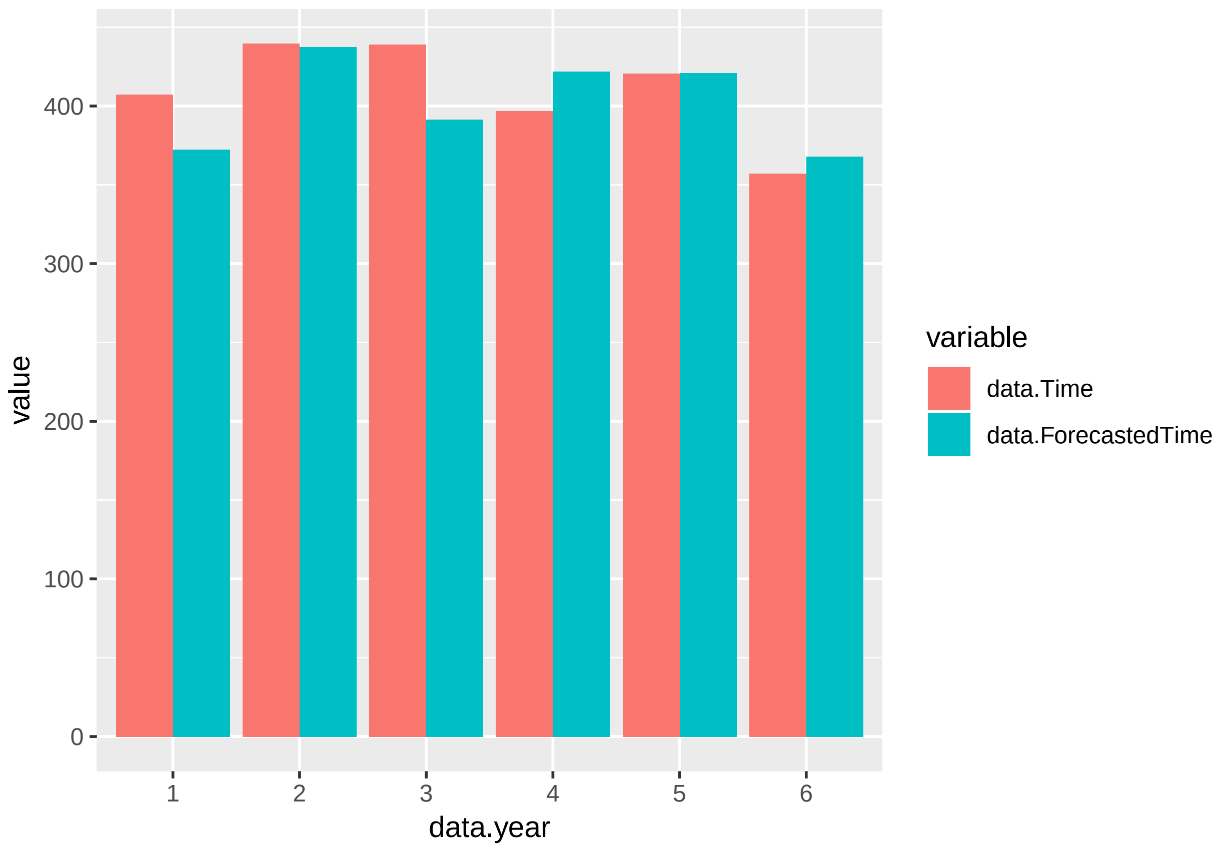
```
ggplot(data = data) +
  aes(x = year, y = Time) +
  geom_boxplot(fill = "#4292c6") +
  theme_minimal()
```



We can see how the highest Time is on the 3rd year, and there's a lot of variance on the 1st and 5th years' races. The second year seems to have some stages with a very high time, so there's a lot of variance on the upper bound, while on the 5th year there does is a lot of variance on the lower bound, meaning there are some stages with less Time than the rest. The 5th year is the only one where this phenomena is so visible.

And let's compare the forecasted Times to the real Times:

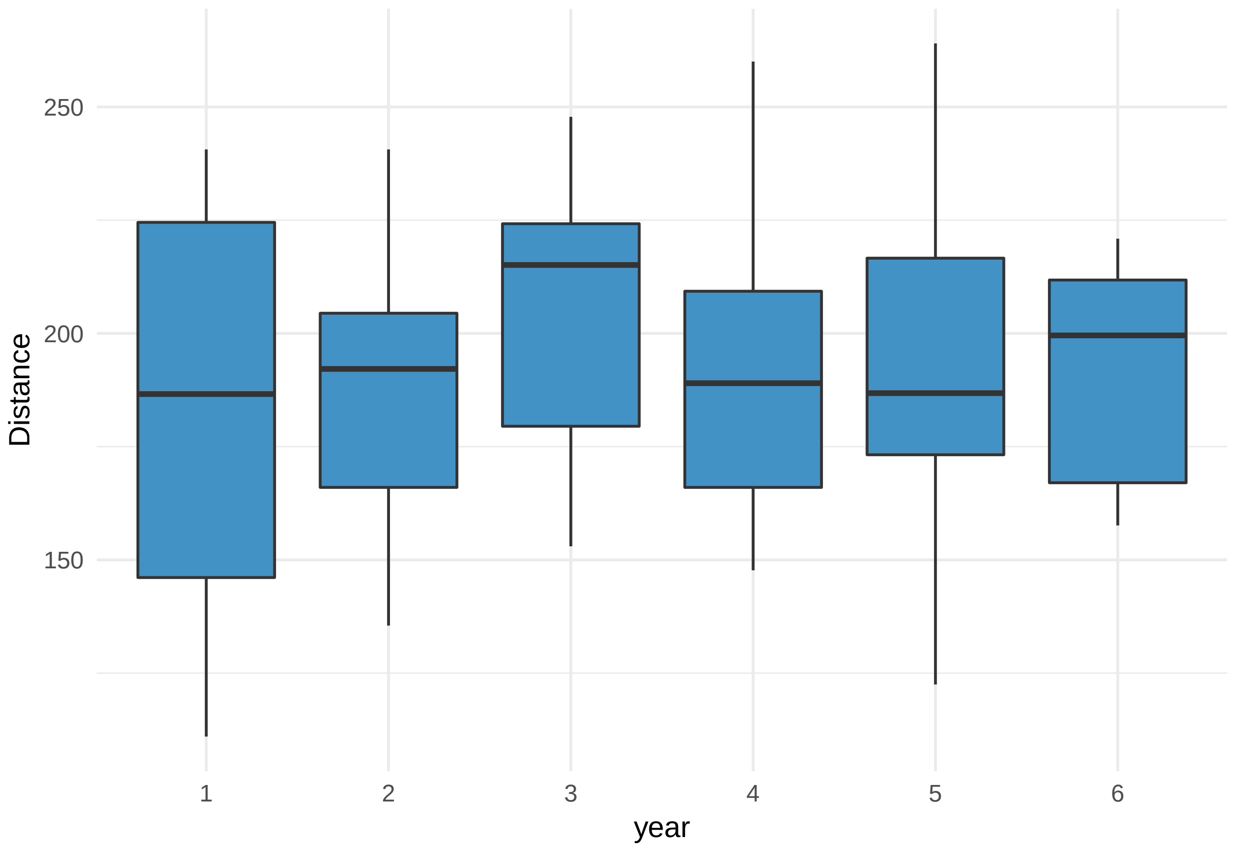
```
df1 <- data.frame(data$Time, data$ForecastedTime, data$year)
df2 <- melt(df1, id.vars='data.year')

ggplot(df2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can see how the Forecasted time is pretty on par with the real time, so the prediction is very good and varies very little in small units.

The worst predictions have been obtained on the 1st and 3rd year. And it has nailed the predictions on the 2nd, 5th and 6th year. An overestimation has been done on the 4th year.

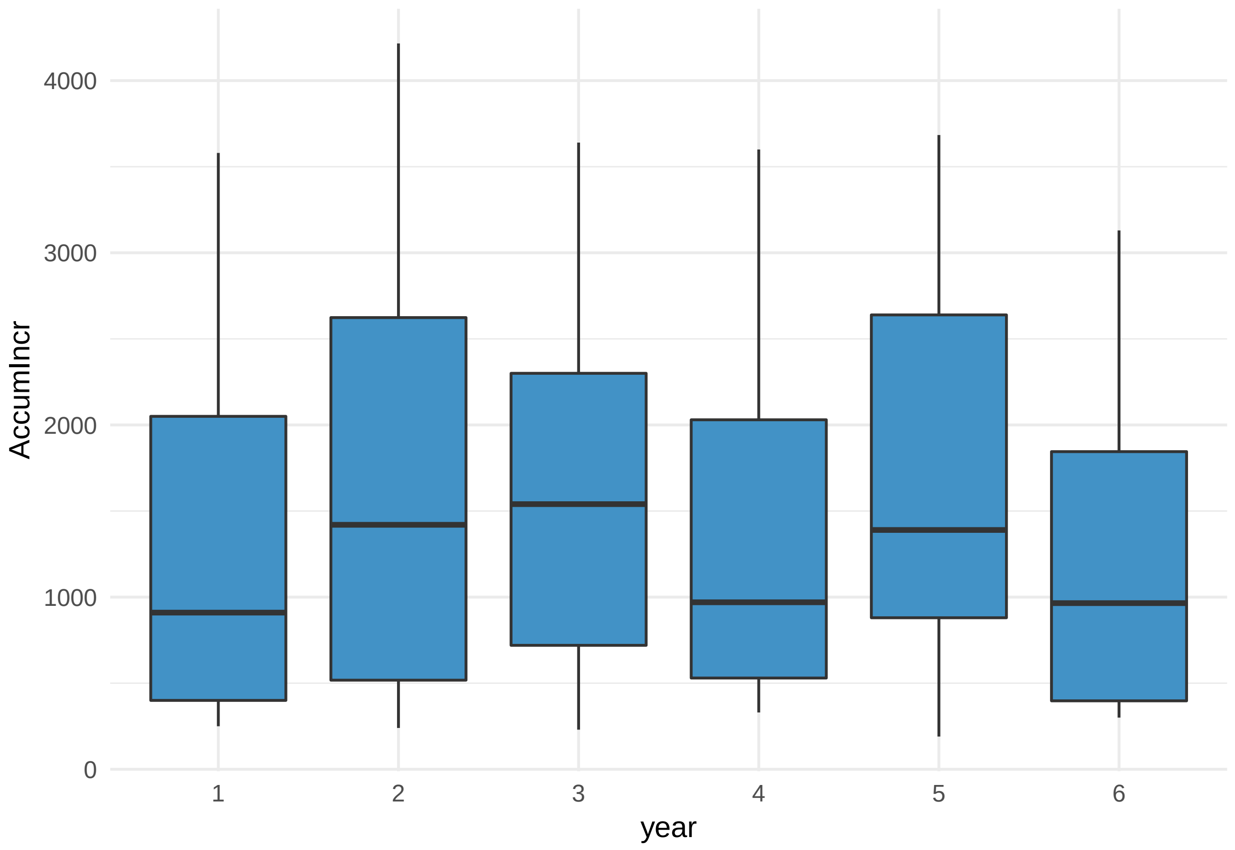
```
ggplot(data = data) +
  aes(x = year, y = Distance) +
  geom_boxplot(fill = "#4292c6") +
  theme_minimal()
```



In the plot above we can see how for each year, the distance does not vary a lot, except on the 3rd year, where there seems to be a higher overall distance but with small variance. The 1st year seems to be the one with most variance on distances between stages and the 5th one does not seem to vary so much, except on some stages qhere the distance differs more (between 100 and past 250). The 6th year seems to be the year with less variance in distance between stages.

Since the 4th year has very little distance, it may have affected the forecasted Time, which was above the real time.

```r
ggplot(data = data) +
  aes(x = year, y = AccumIncr) +
  geom_boxplot(fill = "#4292c6") +
  theme_minimal()
```



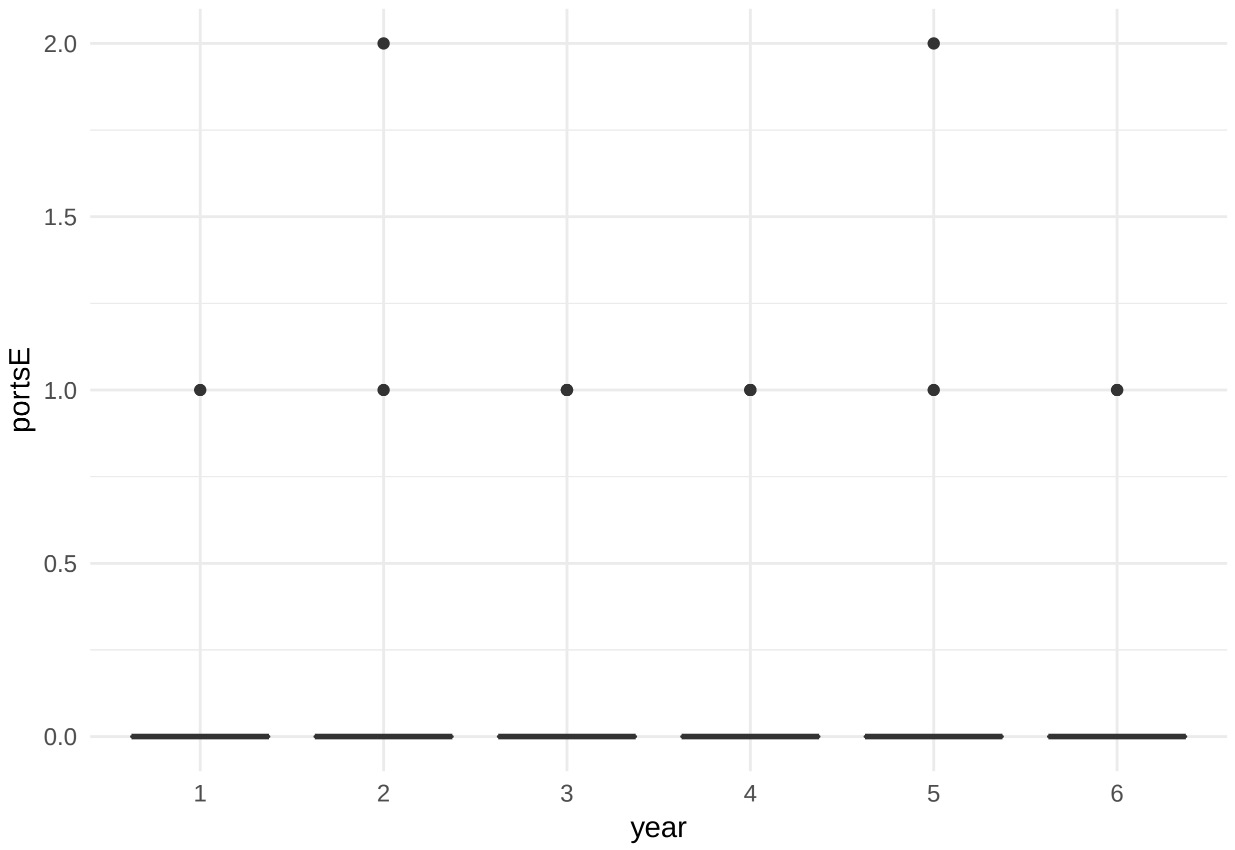
Now, taking a look at the accumulated number of meters climbed on the stages for each year, we can see that the 2nd year is the one with most variance and, in general, most of them vary a lot on the upper bound but not so much on the lower one.

The 3rd year seems to be the one with a highest median and the 4th one seems to be the one with the lowest one.

Now we are going to use a boxplot to see where most ports are. For example, with portsE:

```r
ggplot(data = data) +
  aes(x = year, y = portsE) +
  geom_boxplot(scale = 'area', adjust = 1, fill = '#0c4c8a') +
  theme_minimal()
```

```
## Warning: Ignoring unknown parameters: scale, adjust
```



We can see that that boxplot is not hat helpful, since we explained that it only has integer variables.

Thus we are going to stick to the tables to see how the number of mountain passes distributes:

```
table(data$portsE)
```

```
##
##  0  1  2
## 91 12  2
```

```
table(data$ports1)
```

```
##
##  0  1  2  3
## 68 16 18  3
```

```
table(data$ports2)
```

```
##
##  0  1  2  3
## 70 24  6  5
```

```
table(data$ports3)
```

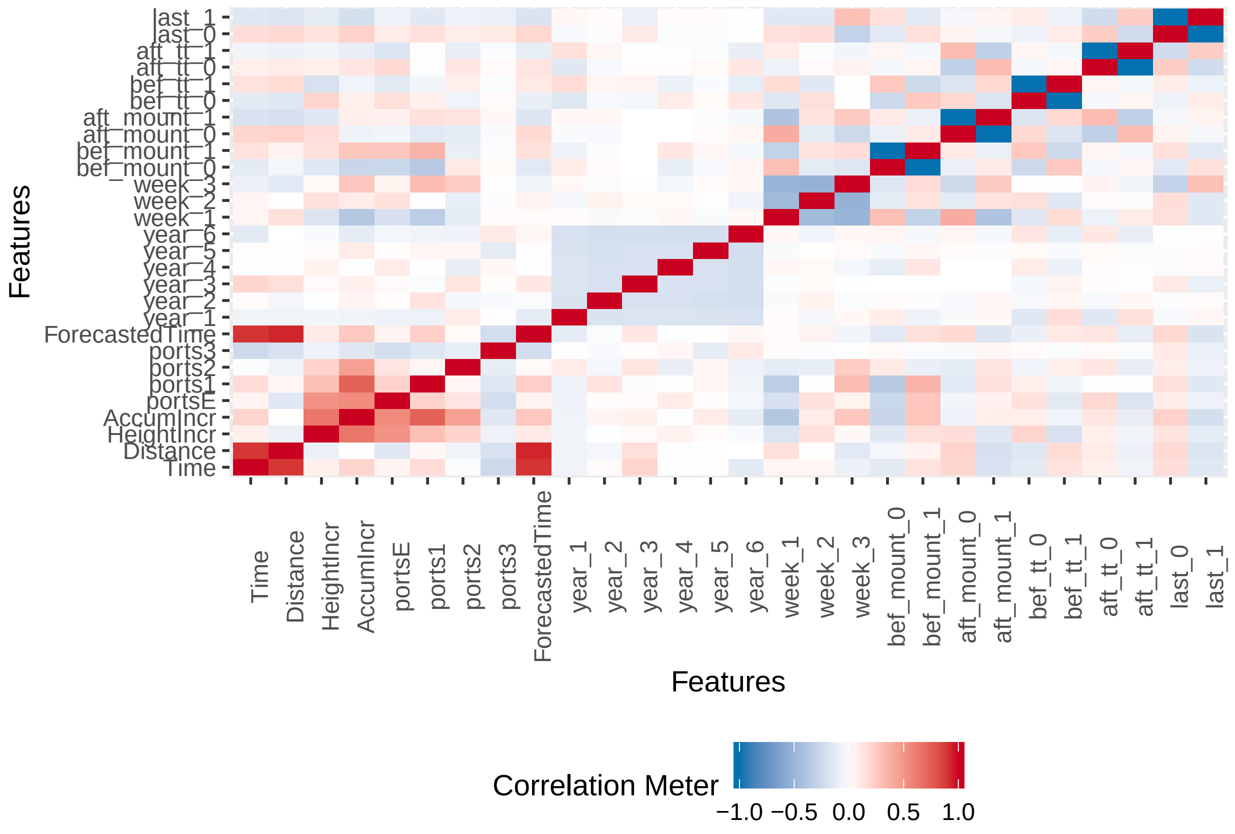
```
##
##  0  1  2  3  4
## 52 25 22  4  2
```

Looking at the ports tables we can see that in portsE, there are mostly no mountain passes, twelve cases with 1 pass and 2 exceptions with 2 passes. On the ports1 and ports2 the same happens but there are some stages where there are 1 or 2 passes, with 3 exceptions with 3 passes. Finally, in ports3 the tendency of having no passes still lasts but there are more cases of 2 passes and two stages with 4 passes, which are the exceptions on the tendency.

3.1 Correlation plot

In order to see if we can find some kind of relation between the variables of our dataset we will use a correlations plot:

```r
plot_correlation(data)
```



From this plot we immediately see that Time and Distance are highly correlated. This makes a lot of sense because, as a general rule, the longer the stage is the longer it will take cyclists to complete. Of course we could find stages that are long but downhill, which would make it possible for the cyclists to complete a long stage in less time, although this is not the usual. This tells us that probably Distance will be an important variable when it comes to predicting the value of the variable Time as a linear combination of the other variables.

It also seems apparent that Time and ForecastedTime are highly correlated, which was to be expected since these two variables represent the same data and ForecastedTime is a prediction of the variable Time. No other relevant correlations can be observed.

4 Model selection

Let's first try creating a linear model. We believe that the year has no influence over the stages so we will not include it into our model.

```r
lm.model <- lm(Time ~ Distance + HeightIncr + AccumIncr  + ports1 + ports2 + ports3 + bef_mount + aft_m
summary(lm.model)
```

```
##
## Call:
## lm(formula = Time ~ Distance + HeightIncr + AccumIncr + ports1 +
##     ports2 + ports3 + bef_mount + aft_mount + bef_tt + aft_tt +
##     last, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -50.82 -13.90  -3.30  11.57  58.80
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.019196  18.102582  -2.155  0.03371 *
## Distance      1.661638   0.083100  19.996  < 2e-16 ***
## HeightIncr    0.001041   0.005041   0.206  0.83690
## AccumIncr     0.018024   0.005654   3.188  0.00195 **
## ports1       -8.017926   5.098681  -1.573  0.11922
## ports2       -6.696263   3.984025  -1.681  0.09616 .
## ports3       -2.581659   2.475275  -1.043  0.29966
## bef_mount1    5.650151   5.821637   0.971  0.33429
## aft_mount1    0.582433   6.145878   0.095  0.92470
## bef_tt1       5.790429   7.512843   0.771  0.44282
## aft_tt1       5.169633   6.754871   0.765  0.44602
## last1        11.195790  12.024858   0.931  0.35424
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.91 on 93 degrees of freedom
## Multiple R-squared:  0.854,  Adjusted R-squared:  0.8367
## F-statistic: 49.44 on 11 and 93 DF,  p-value: < 2.2e-16
```

We can see we have a high adjusted R squared. However there are many variables that are not contributing to the the model, so we will take them away. The variables that we will keep are: Distance, heightIncr AccumIncr. We will also keep Ports2 even though it doesn't seem to be that important.

```r
lm.model2 <- lm(Time ~ Distance  + AccumIncr + ports2, data)
summary(lm.model2)
```

```
##
## Call:
## lm(formula = Time ~ Distance + AccumIncr + ports2, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -54.769 -13.308  -2.277  14.607  62.103
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36.015106  14.800328  -2.433   0.0167 *
## Distance      1.665626   0.073784  22.574  < 2e-16 ***
## AccumIncr     0.012950   0.002472   5.239 8.85e-07 ***
## ports2       -4.142799   3.231176  -1.282   0.2027
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.73 on 101 degrees of freedom
## Multiple R-squared:  0.8438, Adjusted R-squared:  0.8391
## F-statistic: 181.8 on 3 and 101 DF,  p-value: < 2.2e-16
```

With this new model we can see that the adjusted R^2 has increased because we used less parameters, so we will keep this model for now. However we could try removing ports2 since it has been deemed irrelevant.

```r
lm.model3 <- lm(Time ~ Distance  + AccumIncr, data)
summary(lm.model3)
```

```
##
## Call:
## lm(formula = Time ~ Distance + AccumIncr, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.718 -12.772  -3.763  14.480  62.436
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -37.392677  14.807793  -2.525   0.0131 *
## Distance      1.673312   0.073771  22.682  < 2e-16 ***
## AccumIncr     0.011516   0.002211   5.208 9.96e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.81 on 102 degrees of freedom
## Multiple R-squared:  0.8412, Adjusted R-squared:  0.8381
## F-statistic: 270.2 on 2 and 102 DF,  p-value: < 2.2e-16
```

Both R^2 and Adjusted R^2 have decreased ever so slightly but, considering we took one parameter away we will probably keep this model.

Performing a Chisquare test with anova, we can decide which model fits better:

```r
anova(lm.model2, lm.model, test="Chisq")
```

```
## Analysis of Variance Table
##
## Model 1: Time ~ Distance + AccumIncr + ports2
## Model 2: Time ~ Distance + HeightIncr + AccumIncr + ports1 + ports2 +
##     ports3 + bef_mount + aft_mount + bef_tt + aft_tt + last
##   Res.Df   RSS Df Sum of Sq Pr(>Chi)
## 1    101 56893
## 2     93 53174  8    3719.3   0.5908
```

We can see that the p-value is greater than 0.05, so we are going to choose lm.model2 over lm.model, which has almost all variables.

Now, if we check whether to choose lm.model3 or lm.model2, we can see that again the p-value is greater than 0.05, so we are going to choose the simpler model: Time ~ Distance + AccumIncr.

```r
anova(lm.model3, lm.model2, test="Chisq")
```

```
## Analysis of Variance Table
##
```

```
## Model 1: Time ~ Distance + AccumIncr
## Model 2: Time ~ Distance + AccumIncr + ports2
##   Res.Df   RSS Df Sum of Sq Pr(>Chi)
## 1    102 57819
## 2    101 56893  1    925.99   0.1998
```

Since these are nested models, we are going to check their deviance, AIC, perform again the Pearson test(Chi square):

```
computeMetrics <- function(m, df) {

    cat("Deviance: ", deviance(m), "\n")
    cat("AIC: ", AIC(m), "\n")
    cat("Pearson test X^2: ", sum(residuals(m, type = "pearson")^2), "\n")
    cat(" X^2/(N-P): ", sum(residuals(m, type = "pearson")^2)/(nrow(df) - length(m$coefficients)), "\n"
}
cat(" \n * Metrics for lm.model \n ------- \n")
```

```
##
##  * Metrics for lm.model
##  -------
```

```
computeMetrics(lm.model, data)
```

```
## Deviance:  53173.8
## AIC:  977.85
## Pearson test X^2:  53173.8
##  X^2/(N-P):  571.7612
```

```
cat(" \n\n * Metrics for lm.model2 \n ------- \n")
```

```
##
##
##  * Metrics for lm.model2
##  -------
```

```
computeMetrics(lm.model2, data)
```

```
## Deviance:  56893.12
## AIC:  968.9489
## Pearson test X^2:  56893.12
##  X^2/(N-P):  563.2982
```

```
cat("\n\n * Metrics for lm.model3 \n ------- \n")
```

```
##
##
##  * Metrics for lm.model3
##  -------
```

```
computeMetrics(lm.model3, data)
```

```
## Deviance:  57819.11
## AIC:  968.6441
## Pearson test X^2:  57819.11
##  X^2/(N-P):  566.854
```

We can see that the difference on the deviance on the 3 models is small. The lowest AIC is obtained with the third model, although only by a small margin. Performing the Chi square test we can see that the greatest value is found on the 3rd model, and is the same as the deviance, for we are working with a linear model, thus assuming a normal distribution.

So we have decided to work with the linear model number 3, which is the simplest one, lm.model3, that follows the formula

$$Time \sim Distance + AccumIncr$$

4.1 Residuals check.

In this section we will perform residual analysis in order to test whether the assumptions for our linear model are actually fulfilled.

```r
ncol(data)
```

```
## [1] 16
```

```r
nrow(data)
```

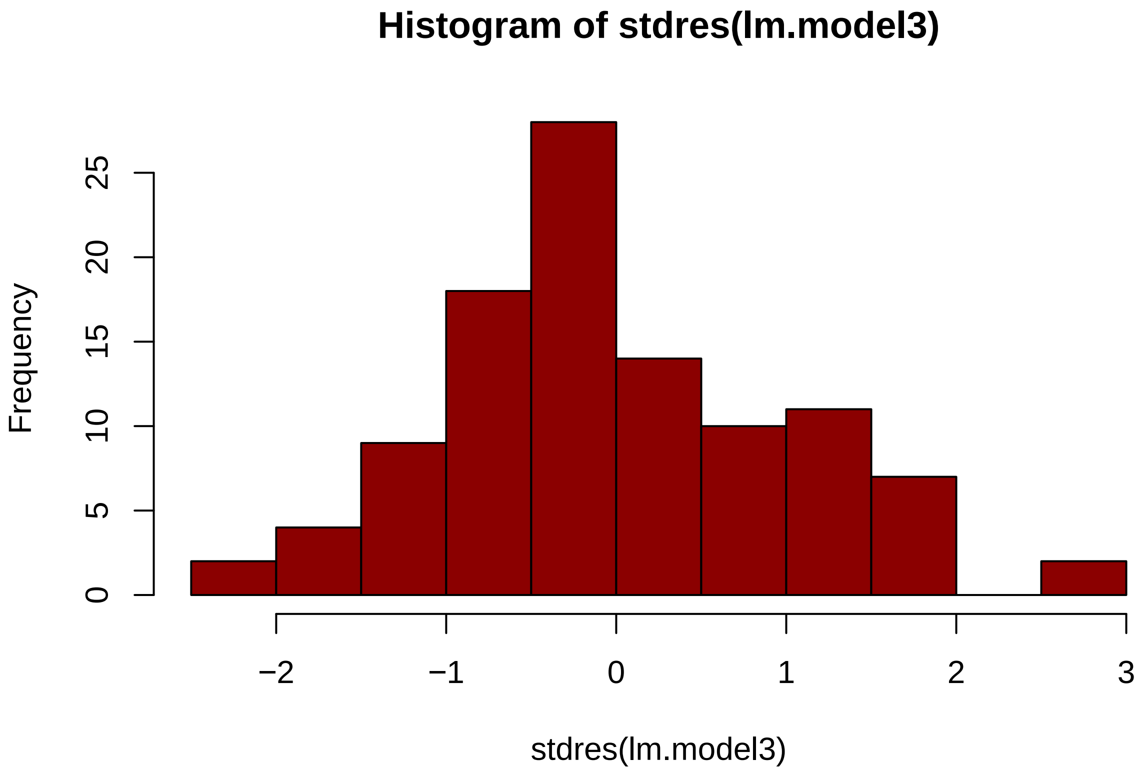
```
## [1] 105
```

Since we have 105 observations and 16 parameters, thus

$$N_{obs} >> p$$

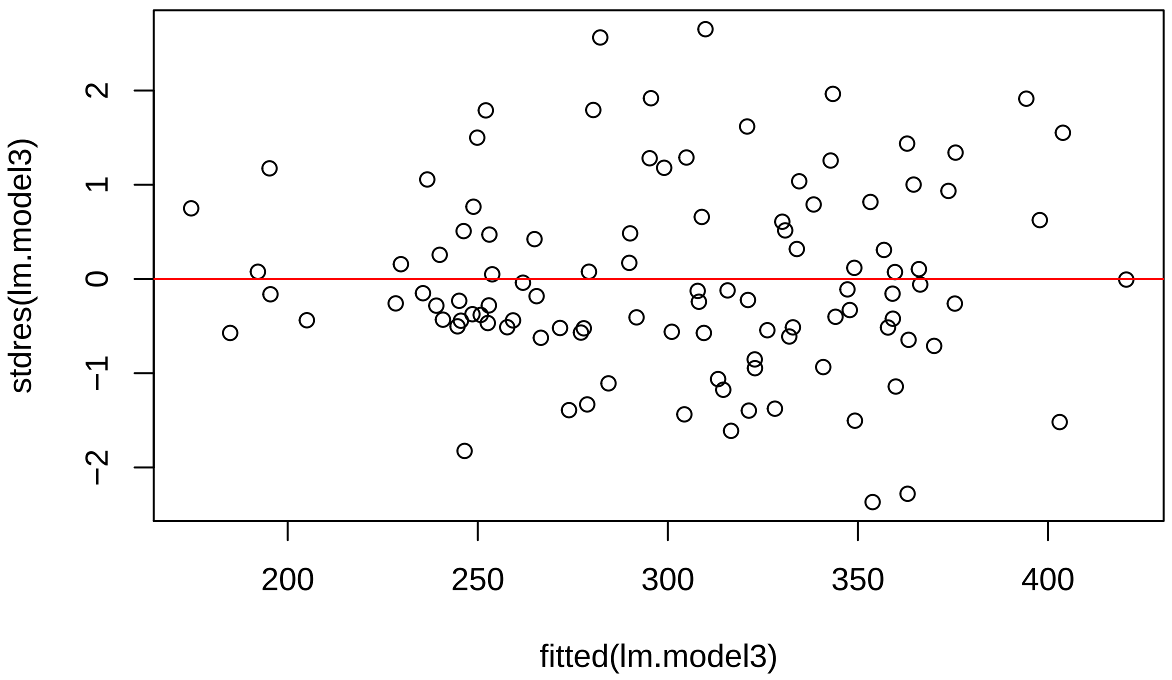
and we can check the histogram of standard residuals.

```r
hist(stdres(lm.model3), col="darkred")
```



It is easy to see that the shape of histogram of the residuals resemble a normal distribution.

```r
plot(fitted(lm.model3), stdres(lm.model3))
abline(h=0, col="red")
```

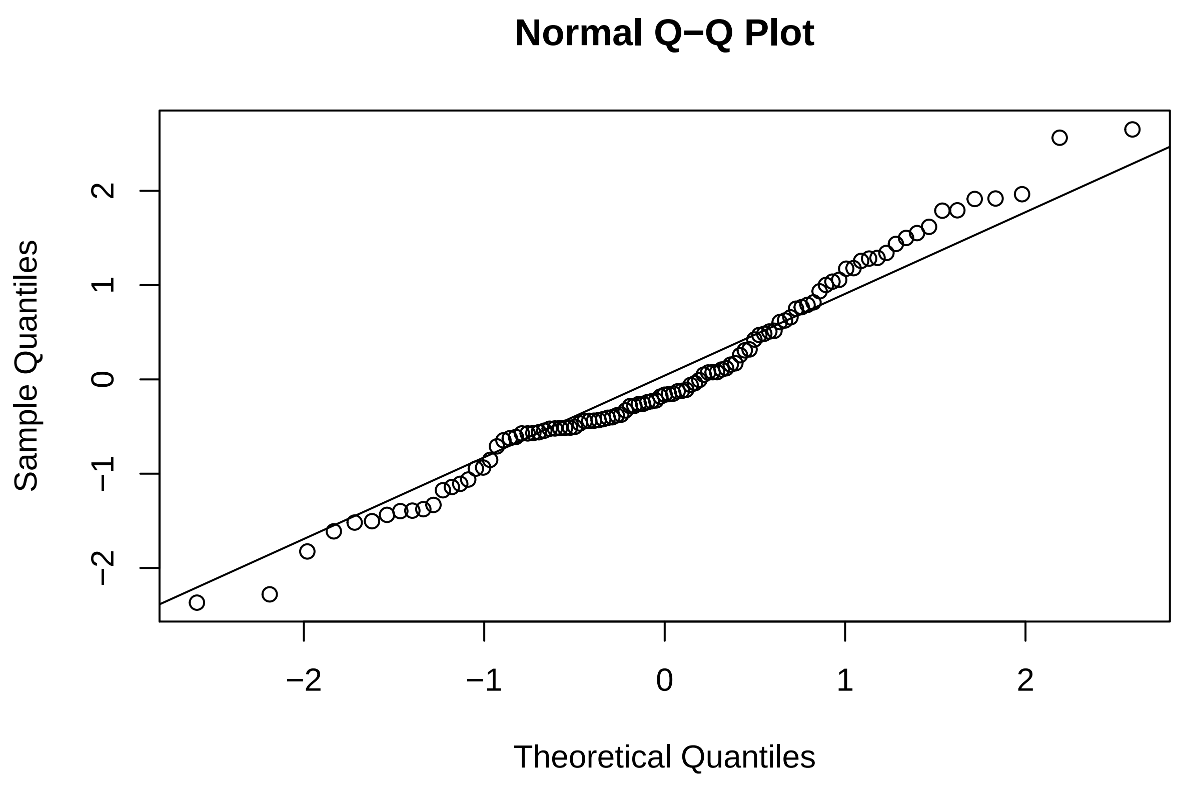


We can see that there is no shape and the points are just scattered randomly, thus there is no pattern.

It seems that 95% or more of the points range between -2 and 2, and thus we can see that Homoscedasticity is complied because there is no shape.

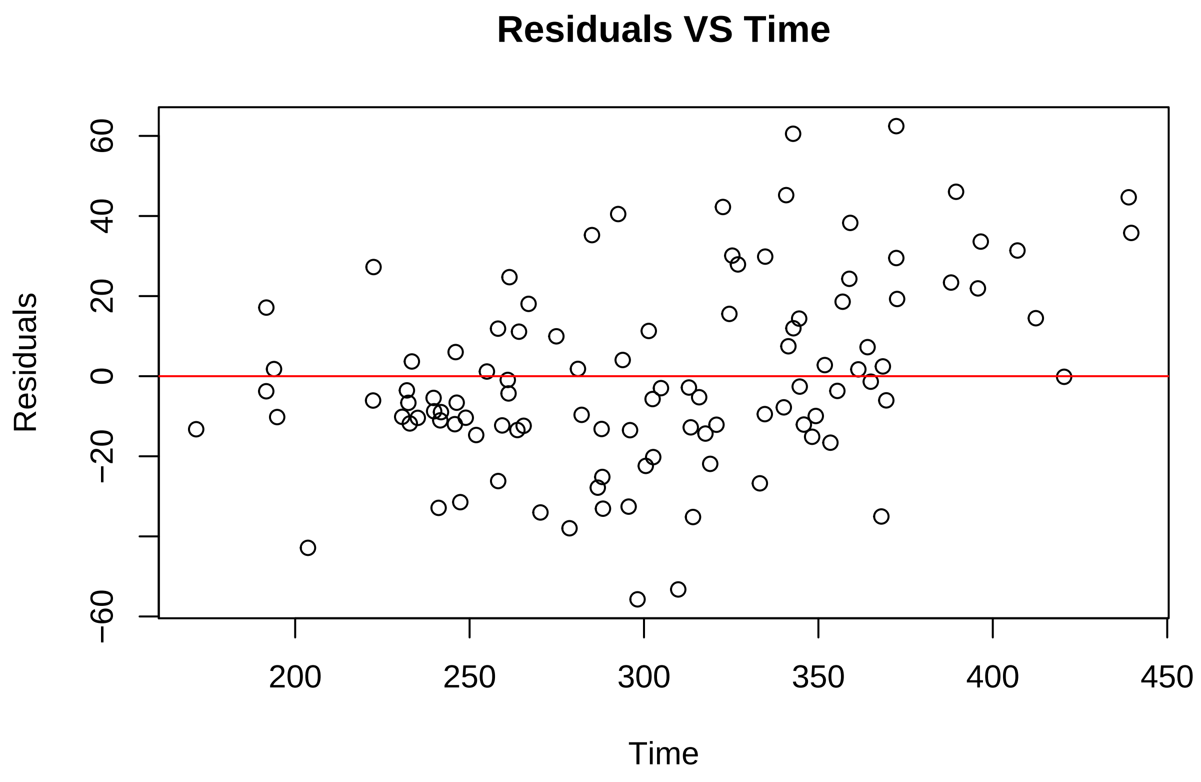
If we qqplot below the residuals we can see that the standardised residuals follow a normal distirbution and thus the normality assumption is satisfied.

```r
res = stdres(lm.model3)
qqnorm(res); qqline(res)
```



Finally we are going to check the independence of resiudals against the predicted value:

```r
res <- residuals(lm.model3)
plot(res ~ data$Time, ylab = 'Residuals', xlab = 'Time', main = 'Residuals VS Time')
abline(h = 0, col='red')
```



We

can see that there is indeed no pattern in the residuals vs time so we can assume independence of variables.

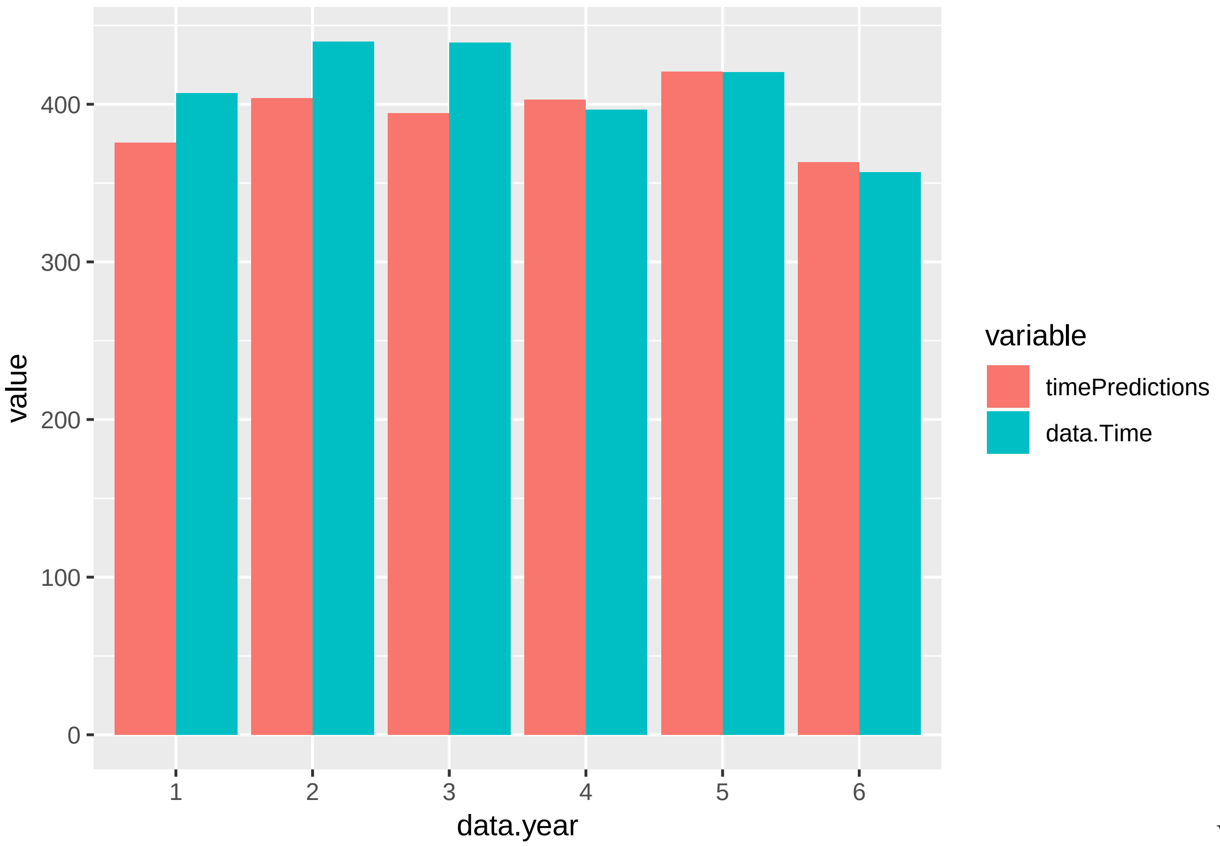
5 Predictions

Now we are going to use our model to predict the time of each stage:

```
timePredictions <- predict(lm.model3, data)
```

Let's compare the predictions with the real time:

```
dfOrg <- data.frame(timePredictions, data$Time, data$year)
dfOrg2 <- melt(dfOrg, id.vars='data.year')

ggplot(dfOrg2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



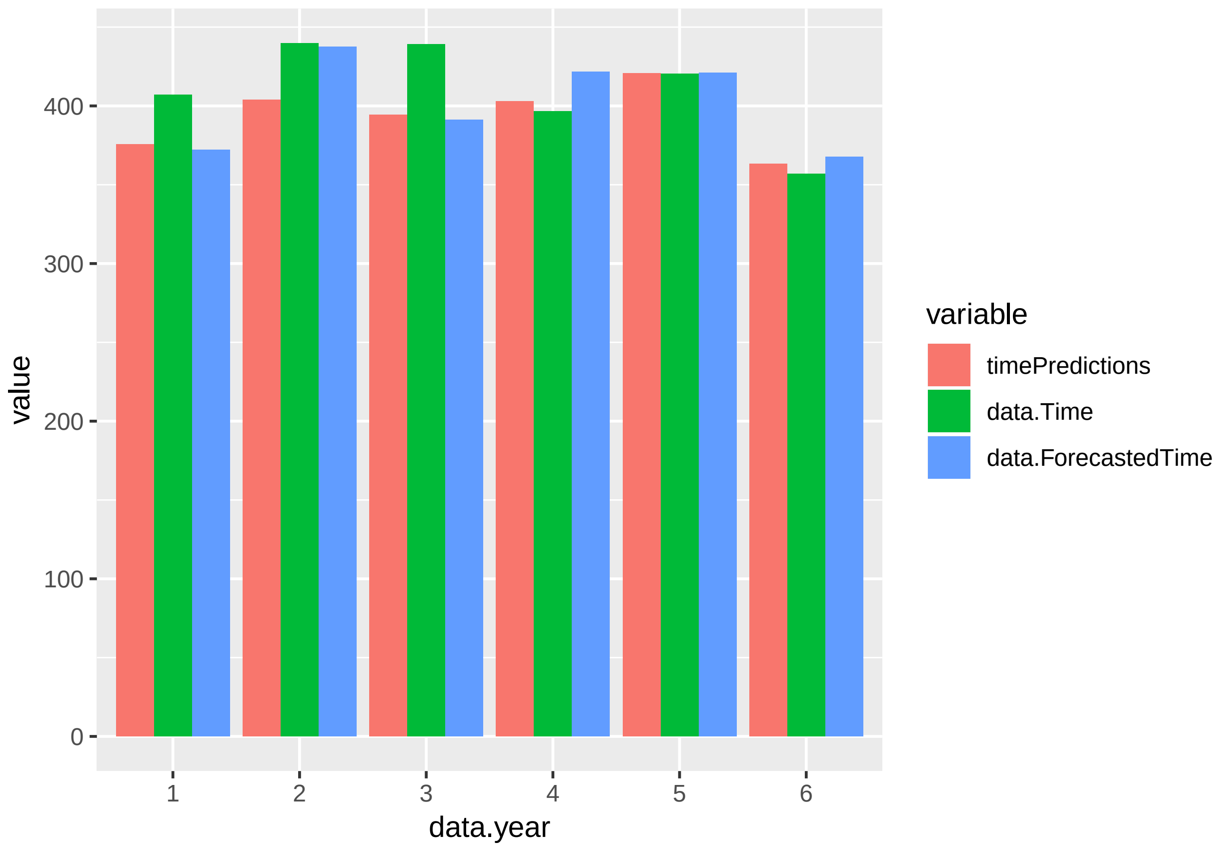
We can see that the predictions fall a little bit short in the first 3 years, but are pretty accurate on the last 3 years.

Let's compare the Time variables:

- Time: The real time.
- Forecasted Time: The time prediction made by an expert.
- timePredictions: Our predictions using our model following the formula `Time ~ Distance + AccumIncr`.

```
dfOrgExp <- data.frame(timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')
```

```r
ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can see how our predictions are pretty similar, almost identical, to those done by an expert. The only notable difference is that our model falls shorter on the prediction in the second year.

On the other hand, it does better in the 4th and 6th year, surpassing the real time way less than the prediction made by an expert, with surpasses the predictions by more units.

In conclusion, looking at the graphic above, we are pretty satisfied with the model we have created and the predictions are good enough.

6 Generalized Linear Model

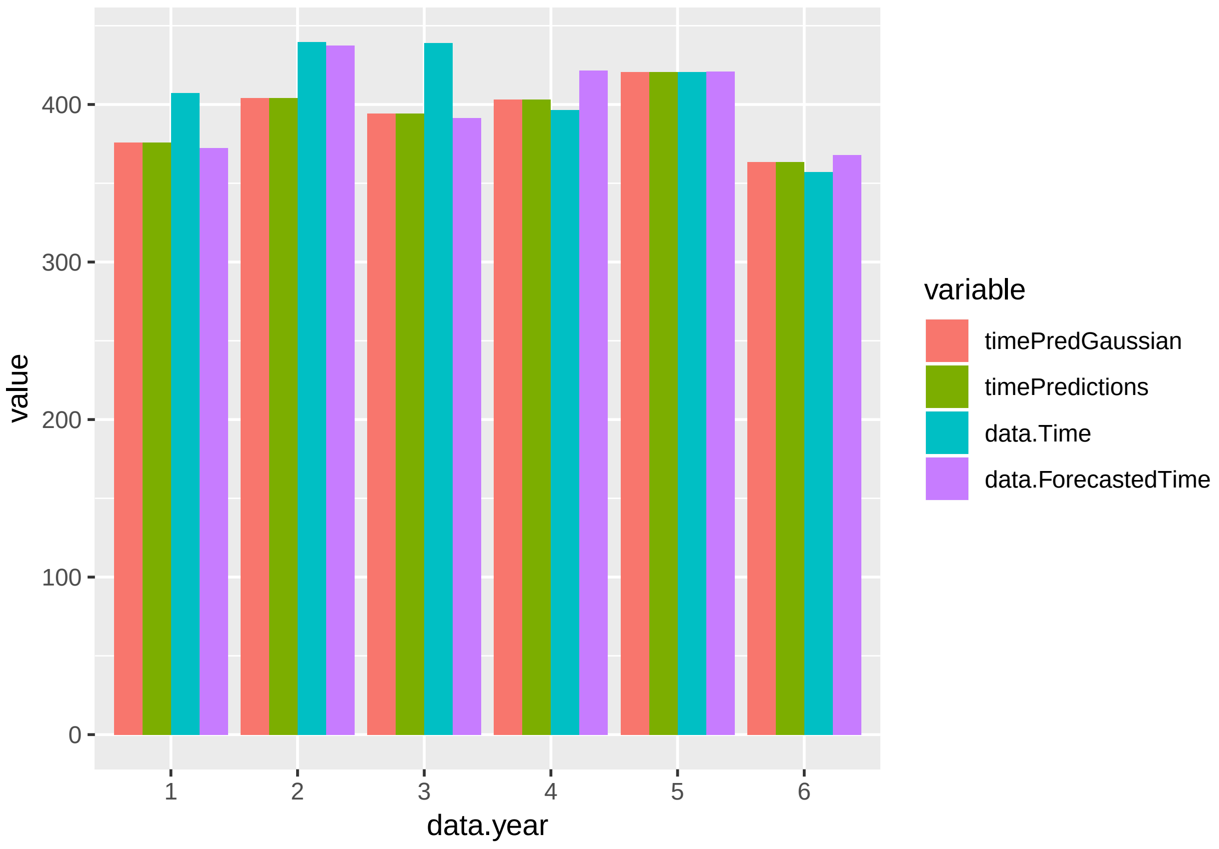
Having done a Linear Model, we have asked ourselves if there was any motivation to do a Generalized Linear Model.

As fas as we know, Time follows a Normal Distribution. This means that the response variable follows a Gaussian distribution, and thus the family of the GLM would be a Gaussian. Having as the link function the identity, then our Linear Model is a subcase of a Generalized Linear Model. Because of this, creating a GLM with a Gaussian and the identity as the link function implies having the same Linear Model that we have.

Just to verify it, let's create a GLM with the same formula, stating that the family is Gaussian and that the link function is "identity".

```r
glm.model <-glm(Time ~ Distance + AccumIncr, data=data, family=gaussian(link="identity"))
timePredGaussian <- predict(glm.model , data)
```

```r
dfOrgExp <- data.frame(timePredGaussian, timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can see how, indeed, the predictions are the same because the model is the same. And if we take a look at the metrics, we can further state that, indeed, they are the same model:

```r
# m: model
# df: Data Frame
computeMetrics <- function(m, df) {

    cat("Deviance: ", deviance(m), "\n")
    cat("AIC: ", AIC(m), "\n")
    cat("Pearson test X^2: ", sum(residuals(m, type = "pearson")^2), "\n")
    cat(" X^2/(N-P): ", sum(residuals(m, type = "pearson")^2)/(nrow(df) - length(m$coefficients)), "\n"
}
computeMetrics(lm.model3, data)
```

```
## Deviance:  57819.11
## AIC:  968.6441
## Pearson test X^2:  57819.11
##  X^2/(N-P):  566.854
```

```r
computeMetrics(glm.model, data)
```

```
## Deviance:  57819.11
## AIC:  968.6441
## Pearson test X^2:  57819.11
##  X^2/(N-P):  566.854
```

6.1 Formula change

So, instead of using the previous formula chosen by us looking at the R squared and using the Pearson Test, we are going to select the terms by using the AIC on the model.

So we start with the whole terms that we deem important, leaving out the years, for they should not determine the time and are only informative:

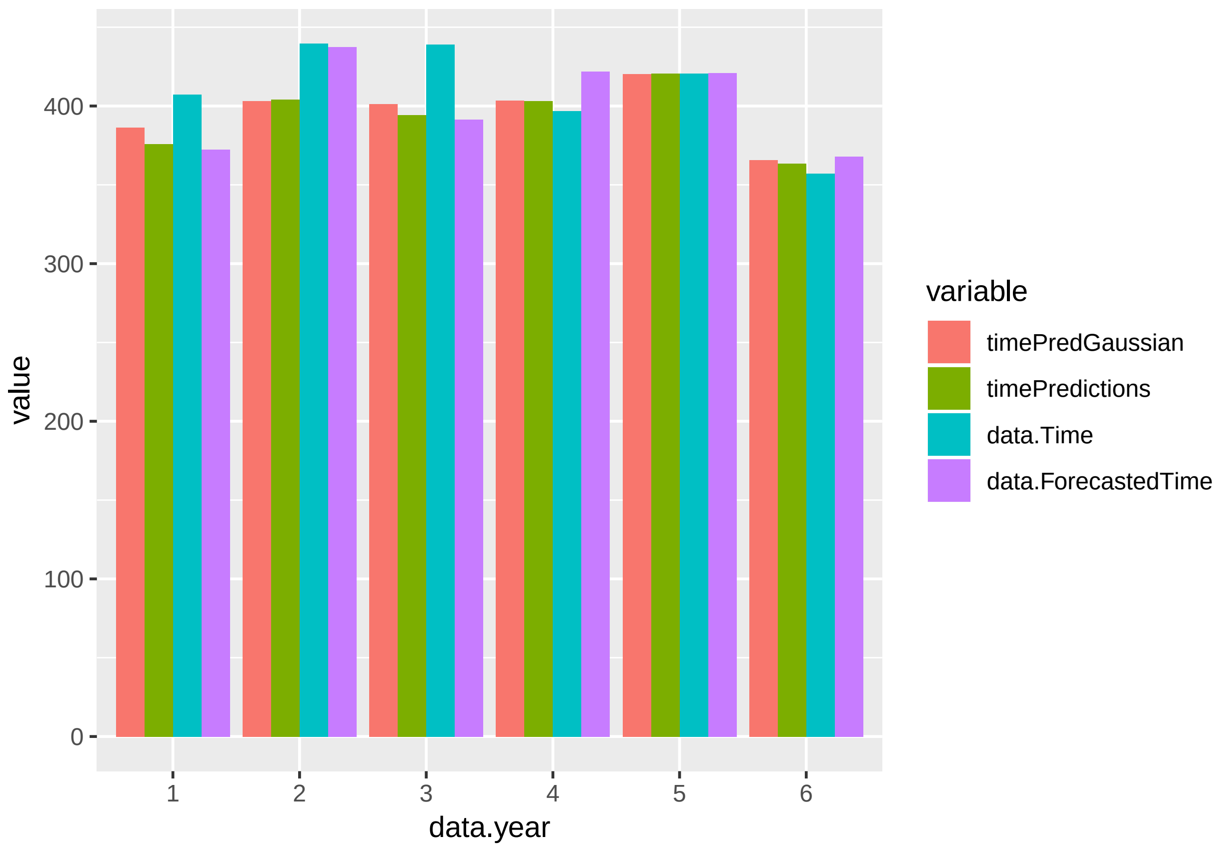
```
glm.model <-glm(Time ~ Distance + HeightIncr + AccumIncr  + ports1 + ports2 + ports3 + bef_mount + aft_r
```

```
glm.model = stepAIC(glm.model)
```

```
summary(glm.model)
```

```
##
## Call:
## glm(formula = Time ~ Distance + AccumIncr + ports1 + ports2,
##     family = gaussian(link = "identity"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -53.383  -13.556   -2.212   14.578   62.389
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -38.584984  14.816959  -2.604   0.0106 *
## Distance      1.668171   0.073376  22.735  < 2e-16 ***
## AccumIncr     0.017713   0.004054   4.369 3.05e-05 ***
## ports1       -6.457969   4.371809  -1.477   0.1428
## ports2       -6.620031   3.623814  -1.827   0.0707 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 556.7818)
##
##     Null deviance: 364119  on 104  degrees of freedom
## Residual deviance:  55678  on 100  degrees of freedom
## AIC: 968.68
##
## Number of Fisher Scoring iterations: 2
```

```
timePredGaussian <- predict(glm.model , data)

dfOrgExp <- data.frame(timePredGaussian, timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can see that the final formula is:

$$Time \sim Distance + AccumIncr + ports1 + ports2$$

so now it takes into account ports1 and ports2.

The predictions have improved slightly on the 1st and 3rd but are not that well on the 2nd, 4th and 6th. So for now we prefer the linear model over this one.

```
anova(lm.model3, glm.model, test='Chisq')
```

```
## Analysis of Variance Table
##
## Model 1: Time ~ Distance + AccumIncr
## Model 2: Time ~ Distance + AccumIncr + ports1 + ports2
##   Res.Df   RSS Df Sum of Sq Pr(>Chi)
## 1    102 57819
## 2    100 55678  2    2140.9   0.1462
```

Additionally, using the Chi square test we can see that the residual sum of squares is greater on the first model, which is to be expected for it is a simpler model. However, looking at the p-value we can see that it is greater than 0.05, thus we are sticking with the simpler linear model.

6.2 GLM: Gaussian distribution, link function: log

Now, since we know that the Time follows a Normal distribution, we will keep the family as a Gaussian, but we will use as link function the logarithm. This means that once we have predicted the values we have to use an exponential to change them to the original units:

```r
glm.model2 <-glm(Time  ~ Distance + HeightIncr + AccumIncr  + ports1 + ports2 + ports3 + bef_mount + af
```

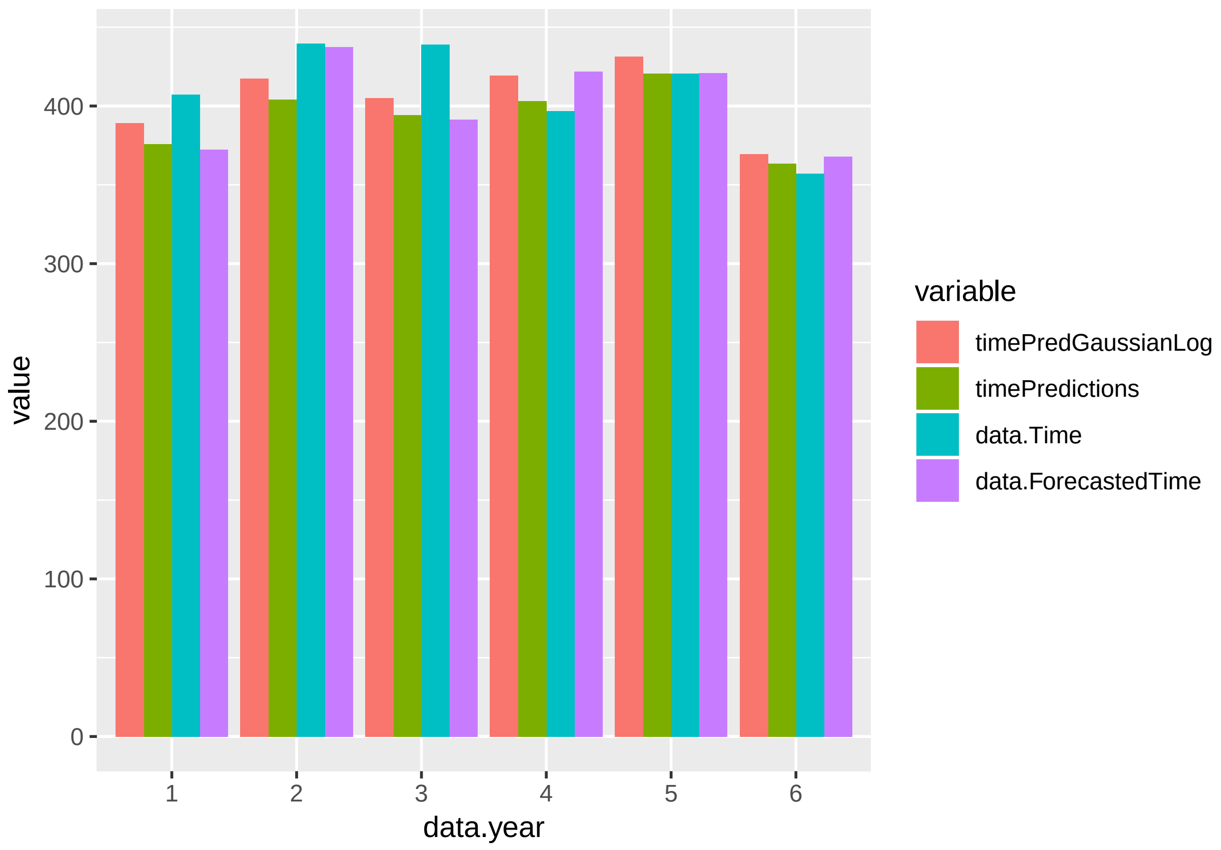
```r
glm.model2 = stepAIC(glm.model2)
```

```r
summary(glm.model2)
```

```
##
## Call:
## glm(formula = Time ~ Distance + AccumIncr + ports1 + ports2 +
##     ports3, family = gaussian(link = "log"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -57.706  -12.896   -2.996   13.996   63.557
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.593e+00  5.577e-02  82.365  < 2e-16 ***
## Distance     5.459e-03  2.561e-04  21.314  < 2e-16 ***
## AccumIncr    5.504e-05  1.264e-05   4.354 3.27e-05 ***
## ports1      -1.924e-02  1.360e-02  -1.415   0.1602
## ports2      -2.132e-02  1.190e-02  -1.792   0.0762 .
## ports3      -1.177e-02  8.436e-03  -1.396   0.1659
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 566.385)
##
##     Null deviance: 364119  on 104  degrees of freedom
## Residual deviance:  56072  on  99  degrees of freedom
## AIC: 971.42
##
## Number of Fisher Scoring iterations: 4
```

We can see how the formula now takes into account ports1, 2 and 3, as well as the distance and AccumIncr as before.

```r
timePredGaussianLog <- predict(glm.model2 , data)
timePredGaussianLog <- exp(timePredGaussianLog)

dfOrgExp <- data.frame(timePredGaussianLog, timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can see how this model seems to get closer to the real values in the first 3 years. However, we can see how it surpasses by a great amount the prediction in the 5th year.

```
anova(lm.model3, glm.model2, test="Chisq")
```

```
## Analysis of Variance Table
##
## Model 1: Time ~ Distance + AccumIncr
## Model 2: Time ~ Distance + AccumIncr + ports1 + ports2 + ports3
##   Res.Df   RSS Df Sum of Sq Pr(>Chi)
## 1    102 57819
## 2     99 56072  3      1747   0.3788
```

Performing an anova with Chisquare test we can see how again we are going to stick with the Linear Model.

6.3 GLM: Gaussian distribution, link function: inverse

Let's check if using the inverse function serves any purpose or better results:

```
glm.model3 <-glm(Time  ~ Distance + HeightIncr + AccumIncr  + ports1 + ports2 + ports3 + bef_mount + af
```

```
glm.model3 = stepAIC(glm.model3)
```

```
summary(glm.model3)
```

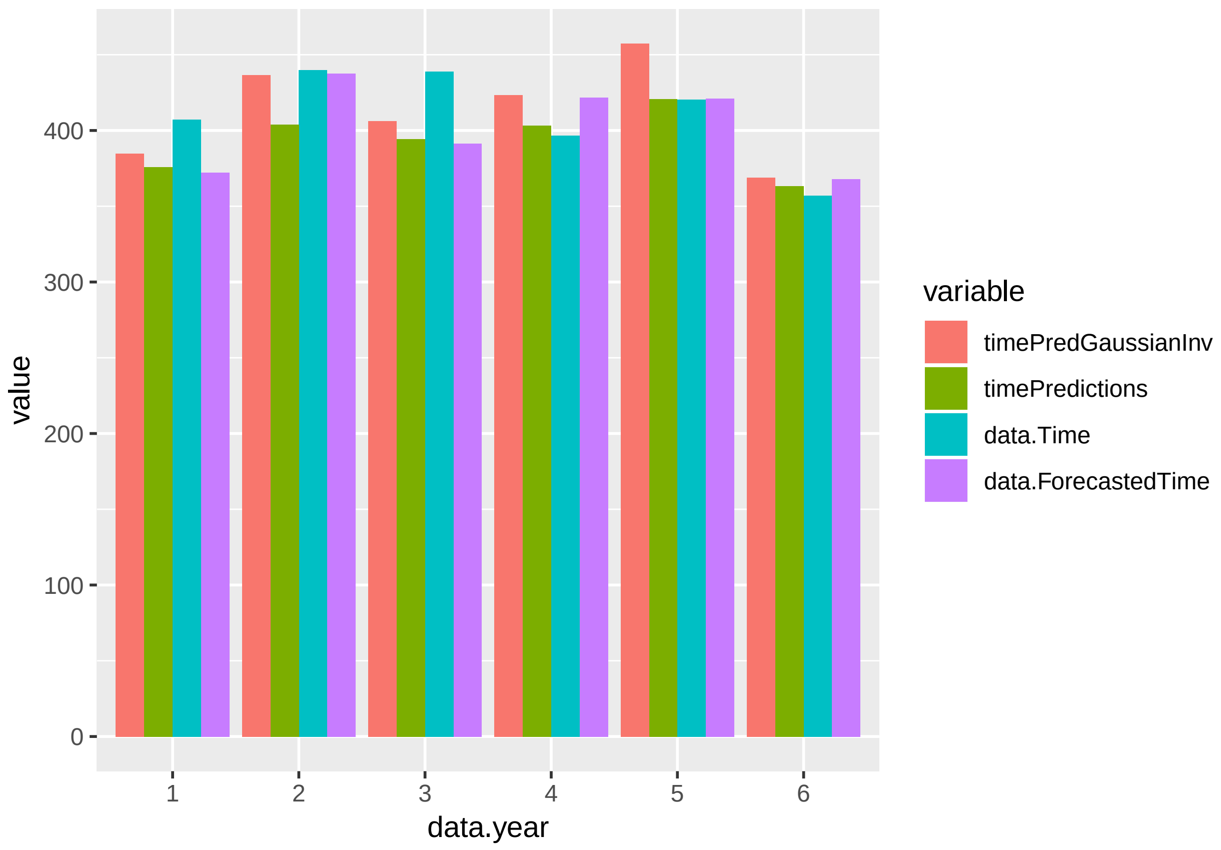
```
##
## Call:
## glm(formula = Time ~ Distance + AccumIncr + bef_mount, family = gaussian(link = "inverse"),
##     data = data)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -55.281  -15.871   -4.269   12.301   72.713
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.973e-03  1.903e-04  36.637  < 2e-16 ***
## Distance    -1.753e-05  8.580e-07 -20.428  < 2e-16 ***
## AccumIncr   -1.129e-07  2.328e-08  -4.849 4.51e-06 ***
## bef_mount1  -8.439e-05  5.454e-05  -1.547    0.125
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 633.0343)
##
##     Null deviance: 364119  on 104  degrees of freedom
## Residual deviance:  63936  on 101  degrees of freedom
## AIC: 981.2
##
## Number of Fisher Scoring iterations: 5
```

We can see how the formula now takes into bef_mount, as well as the distance and AccumIncr as before. Ports have been totally dismissed.

Using the inverse as the link function does not help either:

```
timePredGaussianInv <- predict(glm.model3 , data)
timePredGaussianInv <- 1/timePredGaussianInv

dfOrgExp <- data.frame(timePredGaussianInv, timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



The predictions are pretty far off on the 5th year but better on the rest of years.

```
anova(lm.model3, glm.model3, test="Chisq")
```

```
## Analysis of Variance Table
##
## Model 1: Time ~ Distance + AccumIncr
## Model 2: Time ~ Distance + AccumIncr + bef_mount
##   Res.Df   RSS Df Sum of Sq Pr(>Chi)
## 1    102 57819
## 2    101 63936  1   -6117.3
```

```
AIC(lm.model3)
```

```
## [1] 968.6441
```

```
AIC(glm.model3)
```

```
## [1] 981.204
```

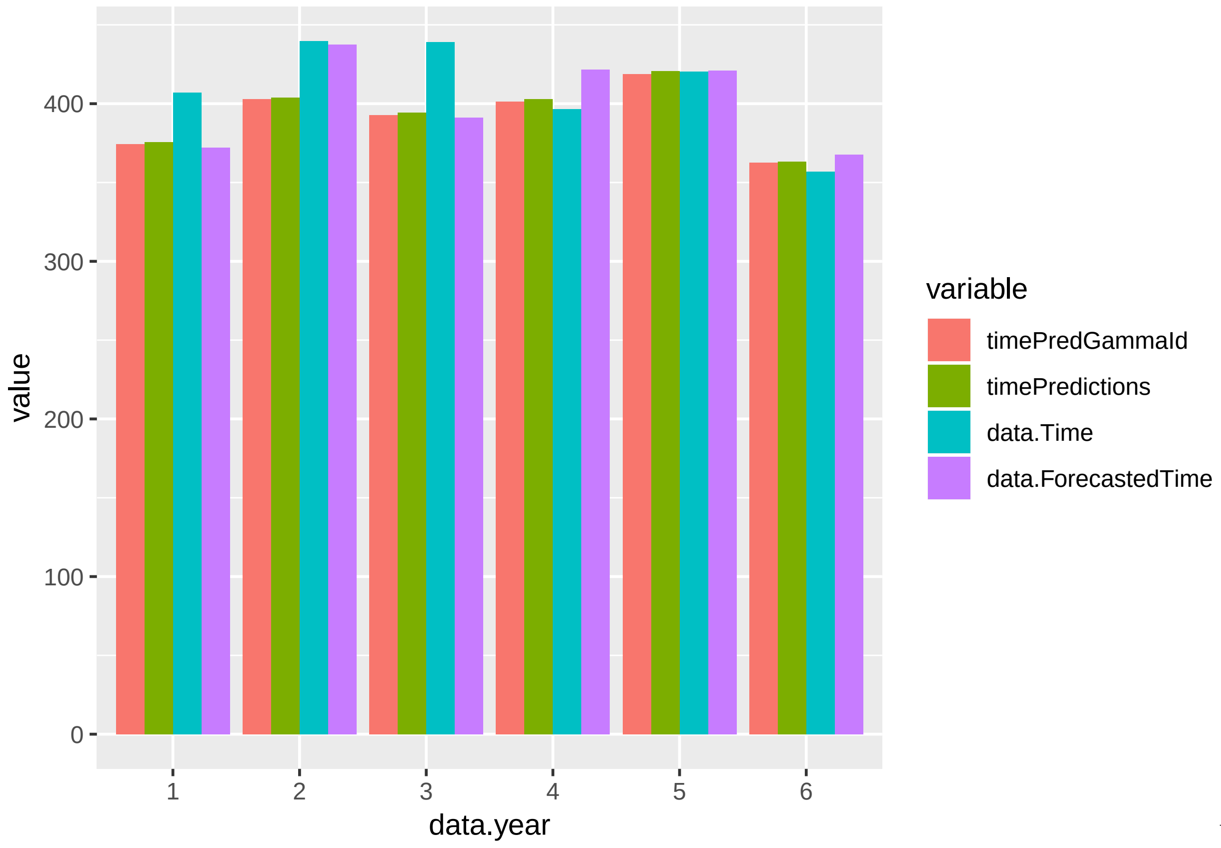
We can see that the AIC are pretty similar, but it is smaller in the linear model. So we are going to stick to that one.

6.4 GLM: Gamma distribution, link function: identity

Since time cannot be negative and is a continuous variable, we are going to consider using the Gamma distribution as the family distribution for the response variable.

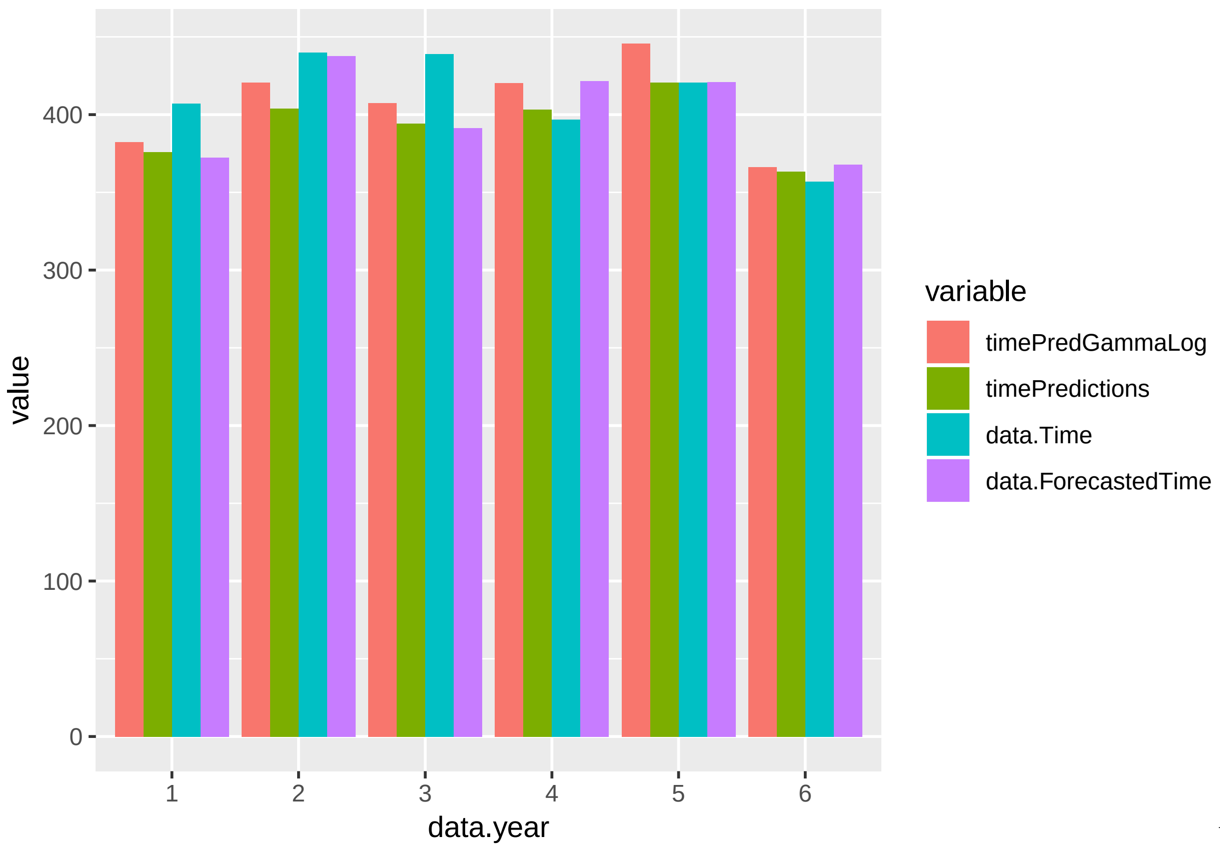
Let's first start with the identity as the link function:

```
glm.model4 <-glm(Time ~ Distance + AccumIncr, data=data, family=Gamma(link = "identity"))
timePredGammaId <- predict(glm.model4 , data)

dfOrgExp <- data.frame(timePredGammaId, timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can see how there's not much of an improvement using the identity as link function. Let's try using the logarithm:

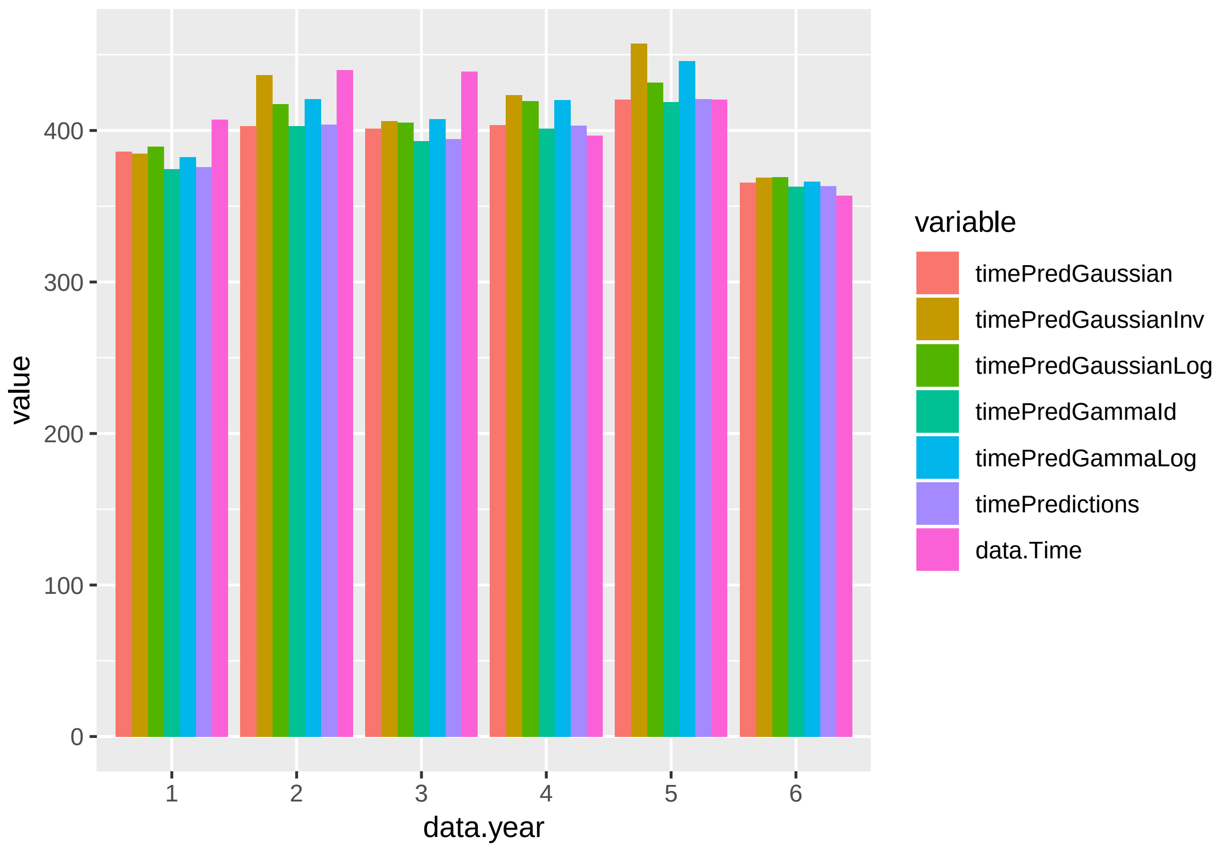
```
glm.model5 <-glm(Time~Distance + AccumIncr, data=data, family=Gamma(link="log"))
timePredGammaLog <- predict(glm.model5 , data)
timePredGammaLog <- exp(timePredGammaLog)

dfOrgExp <- data.frame(timePredGammaLog, timePredictions, data$Time, data$ForecastedTime, data$year)
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



We can, again, see how there is an improvement over the linear model in the first 3 years, but a greater amount on the prediction in the 4th and 5th years.

6.5 All models comparison

```
dfOrgExp <- data.frame(timePredGaussian, timePredGaussianInv, timePredGaussianLog, timePredGammaId, tim
dfOrgExp2 <- melt(dfOrgExp, id.vars='data.year')

ggplot(dfOrgExp2, aes(x=data.year, y=value, fill=variable)) +
    geom_bar(stat='identity', position='dodge')
```



Looking at the graphic above, we can see that all predictions are short on the first year. The Gaussian with inverse link function seems to be the one that performs better on the second year. However, it is also the worst performing on the 4th and 5th year, overestimating the time a lot.

By looking at these comparisons, we prefer to use the linear model over any Generalized Linear model.

7 Conclusions

After many tests and having checked the response variable, we can determine that:

1. Time follows a Normal Distribution.
2. Hypothesis of homocedasticity, normality and independence cannot be rejected.
3. A Linear Model using Distance and AccumIncr seems to perform well with the data at hand and the predictions are quite accurate.
4. We cannot justify the usage of a Generalized Linear Model.

7.1 Future Estimations

This model has been created using observations from 6 years. We think that with the data at hand, we would be able to estimate accurately enough the Time for the riders to finish each stage, as long as the organizers of `La Vuelta` do not change a lot other parameters than Distance and AccumIncr that may have a greater impact than now.

For example, we can take a strange case where there are a lot of ports (mountain passes). We would not be taking them into account in our model. However, having a lot of ports may have an impact on the number of stops done by a rider, thus having an impact on the final time.

However, organizers of `La Vuelta` tend and try to make the competition similar to other years so that it is fair for those competitors participating each year, so that having won on one year holds the same value as having won 3 years later.

Thus, we think our model will hold well enough for another 15 `Vueltas` later, although having more data for more years would have been better, since we are trying to predict the Times for twice the years of data we have currently.