NFL games maximum

Hello for this project I am going to attempt to find a 20 week game for the NFL. So like a once in a 20 week score basically. So like what is the score of the year or like a once in a year score and so on.

Questions for office hours and so I don't forget

1 - I had some questions on my model fitting the GEV and Gumbel models basically sucked so I think it is because there isn't enough observations to fit it based on week and we are going to have to do a once in a year model on that data. However I got a GP model to fit which makes sense as it isn't constrained by week like the other models and therefore has to small of a sample size to not be overly skewed on the initial observation. Basically I was you to look at my GP model fit in comparison to the histogram. It under estimates the first value however, I still think it is a pretty good fit actually.

2 - When I am calculating my once in blank week prediction I am having some trouble on how exactly to scale it. There is definitely no statistically normal or correct way to do it because I am going to have to take into account playoffs and so forth. As well that could be a reason why my GEV model did work because it definitely would be forced to take like 1 game in for the superbowl for that week so I am just realizing that I would need to clean that before I fit it and only fit it on the regular season. - 2 post realization - However, for the GP model. How would I scale it basically. I am thinking I would have to scale it basically by season. However the regular season total number of games changed throughout the data set. Like the super bowl for the years 2021 and 2022 was week 22 however it was 21 for every other year. As well there was one more regular season game those two years. As well after week 17 (one week is a bye so they play 16 games but over 17 weeks) there is a schedule of 4 games, 4 games, 2 games, then 1 game. So How do I take that into account when basically coming up with the correct 1 in a season game. Do I just divide it by the total number of games in a season. Which I am thinking is correct now so I am going to do that now and the we will see.

3 - This is a continuation of question 2. When I am looking at what you did for the GP on the boulder rain data set. I see you are only focusing on the 4 months or whatever. Then when you calculate your once in 20 year max you take those days divided by the total days to get the correct maxima? So if I want to get the 1 season estimate it should just be 1*average games in a season correct. As well for the 1 in 10 season it is just 1*average games in a season?

Data Loading

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Reading in games data set. Said to include the pbp <- pbp %>% pipeline part but I feel as though that is nessary to join this with other datasets. however, we don't really need to do this.

```
games <- read.csv("http://www.habitatring.com/games.csv")

## Said to include this when importing but idk what pbp is supposed be and where it comes from so I am
# pbp <- pbp %>%
#   inner_join(games,by=c("game_id"="game_id","away_team"="away_team","home_team"="home_team"))
```

What we are going to do on this dataset now is we are going to create a week variable then we are going to group on the maxes for each of those weeks score wise. Then we can fit our extremes fit on that dataset. In order to check it we can plot a hist and see if it follows more of an extremes distribution or normal. :)

Data cleaning and wrangling

no NA's in important columns

```
sum(is.na(games))
```

```
## [1] 17808
```

```
Games <- games %>% drop_na(week) %>%
  drop_na(season) %>%
  drop_na(away_score) %>%
  drop_na(home_score) %>%
  drop_na(game_id)
sum(is.na(Games[,c("season","away_score","week","home_score", "game_id")]))
```

```
## [1] 0
```

Creating a unique week column which is combination of year and week. So we can group by games each week.

```
library(stringr)
Games$uniqueWeek <- str_c(Games$season, '-', Games$week)
```

Creating new DF: team scores

```
gameScores <- Games[, c("game_id", "uniqueWeek","week", "season", "away_team", "away_score","home_team"
```

```
sum(is.na(gameScores))
```

```
## [1] 0
```

```
head(gameScores)
```

```
##           game_id uniqueWeek week season away_team away_score home_team
## 1 1999_01_MIN_ATL     1999-1    1   1999       MIN         17       ATL
## 2  1999_01_KC_CHI     1999-1    1   1999        KC         17       CHI
## 3 1999_01_PIT_CLE     1999-1    1   1999       PIT         43       CLE
## 4  1999_01_OAK_GB     1999-1    1   1999       OAK         24        GB
## 5 1999_01_BUF_IND     1999-1    1   1999       BUF         14       IND
## 6  1999_01_SF_JAX     1999-1    1   1999        SF          3       JAX
##   home_score
## 1         14
## 2         20
## 3          0
## 4         28
## 5         31
## 6         41
```

now need to split it up so each team is separate. Like instead of having a home team and an away team transform it into a dataframe that has: uniqueWeek, week, season, team, score, home/away.

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
teamNames <- melt(gameScores[,c("game_id","uniqueWeek", "week", "season", "away_team", "home_team")],va
teamScores <- melt(gameScores[,c("game_id","uniqueWeek", "week", "season", "away_score", "home_score")]
```

now join two dataframes together with an inner join

```
scores <- teamNames
scores$score <- teamScores$score
```

removing extra columns

```
scores <- scores[, !names(scores) %in% c("variable.y")]
colnames(scores)[colnames(scores) == "variable"] ="home/away"
```

Testing

```
#okay it is fucked the teams and games does not match up it is an error in the merge
#can test with game idea which is pretty ideal
scores[scores$game_id == "2005_12_NYG_SEA",]#couple random game id's
```

```
##              game_id uniqueWeek week season home/away team score
## 1752 2005_12_NYG_SEA    2005-12   12   2005 away_team  NYG    21
## 8173 2005_12_NYG_SEA    2005-12   12   2005 home_team  SEA    24
```

```
scores[scores$game_id == "1999_01_MIN_ATL",]
```

```
##              game_id uniqueWeek week season home/away team score
## 1    1999_01_MIN_ATL     1999-1    1   1999 away_team  MIN    17
## 6422 1999_01_MIN_ATL     1999-1    1   1999 home_team  ATL    14
```

```
scores[scores$game_id == "2003_15_HOU_TB",]
```

```
##             game_id uniqueWeek week season home/away team score
## 1260 2003_15_HOU_TB    2003-15   15   2003 away_team  HOU     3
## 7681 2003_15_HOU_TB    2003-15   15   2003 home_team   TB    16
```

```
scores[scores$game_id == "2013_02_CAR_BUF",]
```

```
##               game_id uniqueWeek week season home/away team score
## 3734  2013_02_CAR_BUF     2013-2    2   2013 away_team  CAR    23
## 10155 2013_02_CAR_BUF     2013-2    2   2013 home_team  BUF    24
```

```
games[games$game_id == "2005_12_NYG_SEA",c("season", "week", "away_team", "away_score", "home_team", "ho
```

```
##      season week away_team away_score home_team home_score
## 1752   2005   12       NYG         21       SEA         24
```

```
games[games$game_id == "1999_01_MIN_ATL",c("season", "week", "away_team", "away_score", "home_team", "ho
```

```
##   season week away_team away_score home_team home_score
## 1   1999    1       MIN         17       ATL         14
```

```
games[games$game_id == "2003_15_HOU_TB",c("season", "week", "away_team", "away_score", "home_team", "hom
```

```
##      season week away_team away_score home_team home_score
## 1260   2003   15       HOU          3        TB         16
```

```
games[games$game_id == "2013_02_CAR_BUF",c("season", "week", "away_team", "away_score", "home_team", "ho
```

```
##      season week away_team away_score home_team home_score
## 3734   2013    2       CAR         23       BUF         24
```

```
#, "1999_01_MIN_ATL", "2003_15_HOU_TB", "2013_02_CAR_BUF")
```

```
rm(list = setdiff(ls(), "scores"))
```

EDA

hist of all scores. Sort of seems normally distributed but not really. Definitely skewed as well has a really far right tail

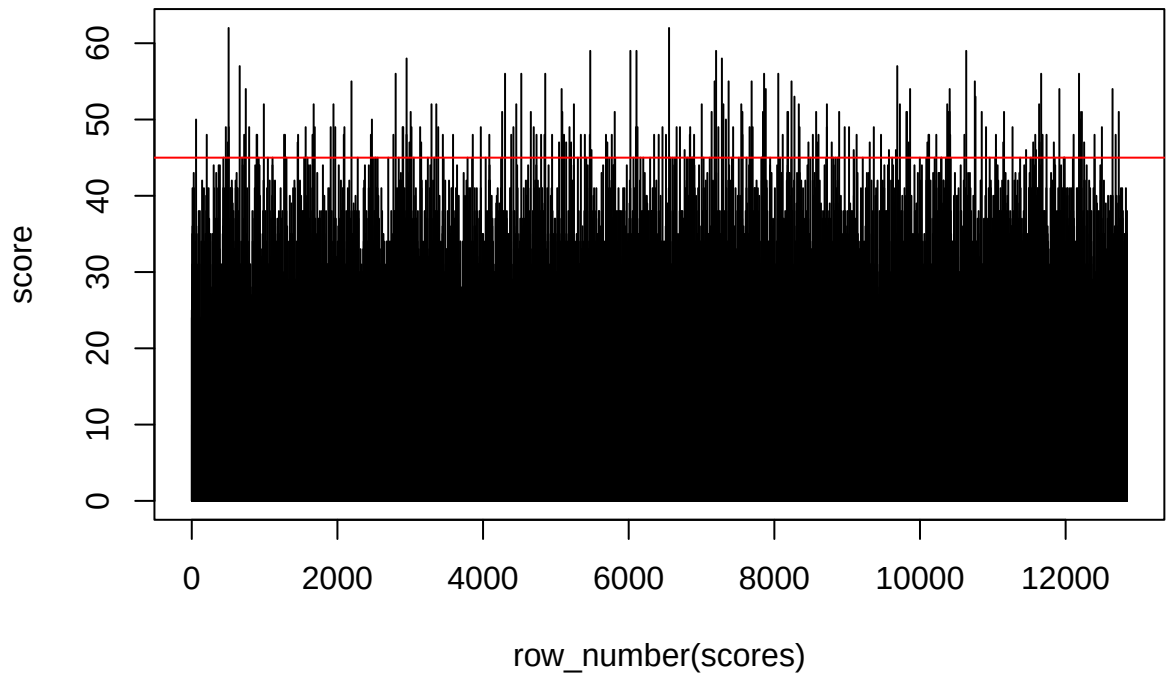
```
length(unique(scores$uniqueWeek))
```

```
## [1] 506
```

```
scoreMax<- tapply( scores$score,scores$uniqueWeek,
                   max, na.rm=TRUE)
weekMax<-  names( scoreMax)
```

time series plot Notes: Looks pretty consitant over time. I honestly thought that it would slope upwards and have higher scores more recently with passing becoming more popular in the nfl. Interesting to see that, that isn't entirely the case.

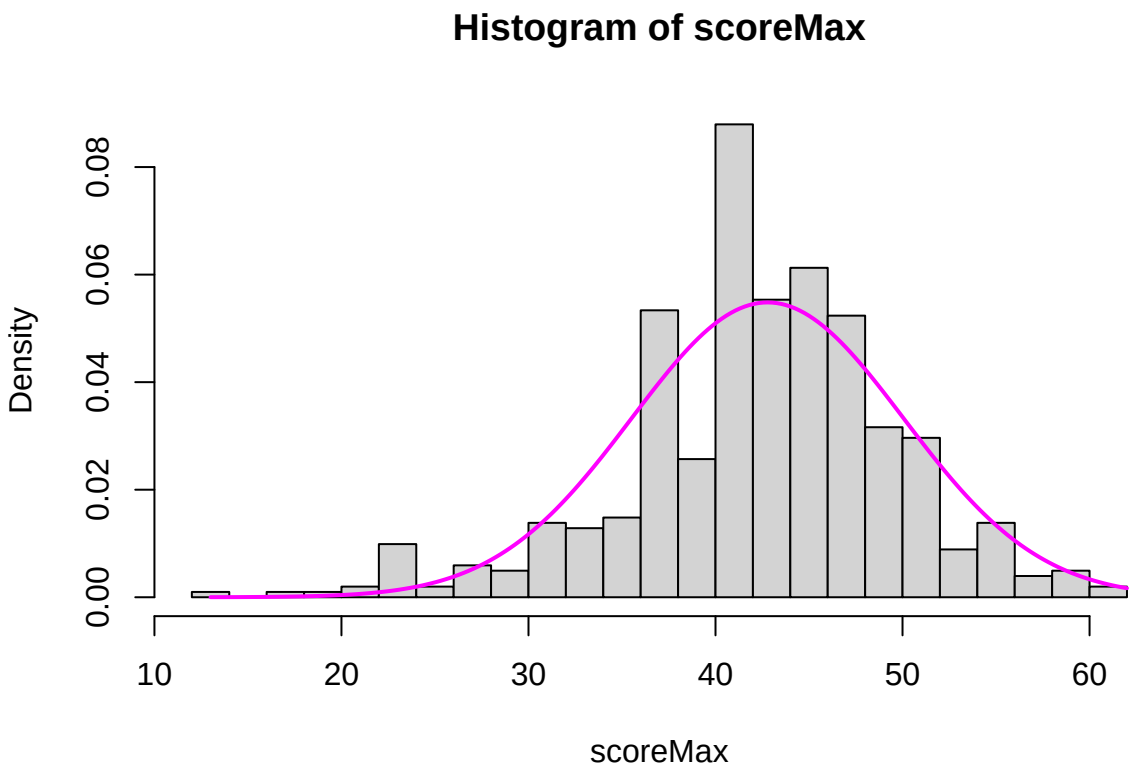
```
plot( score ~ row_number(scores), data= scores,type="h")
abline( h=45, col="red")
```




```
#points( weekMax, scoreMax,col="orange3",pch=16 )
#too much work to get maxs to plot on there without showing us to much
```

You can see here that the normal distribution doesn't capture the max at all. However it does get the tails relativly well which is pretty interesting.

```
xGrid<- seq( min( scoreMax), max( scoreMax), length.out=200)
normFun <- dnorm(xGrid, mean = mean(scoreMax), sd = sd(scoreMax))
hist(scoreMax, probability=TRUE,  nclass=20)
lines(xGrid, normFun, col = "magenta", lwd = 2)
```



Maximum libraries

```
suppressMessages(library( fields))
suppressMessages(library( scales))
suppressMessages(library( extRemes))
```

Fitting the Maxima

Fitting GEV and Compairing it to the gumbel

```
outGEV <- fevd(scoreMax, type = "GEV")
summary(outGEV)#these values sort of suck making me think it didn't really converge well
```

```
##
## fevd(x = scoreMax, type = "GEV")
##
## [1] "Estimation Method used: MLE"
##
##
##  Negative Log-Likelihood Value:  6570.093
##
##
##  Estimated parameters:
```

```
##  location     scale     shape
## -417.7597 5972.2602  334.9728
##
##  AIC = 13146.19
##
##  BIC = 13158.86
```

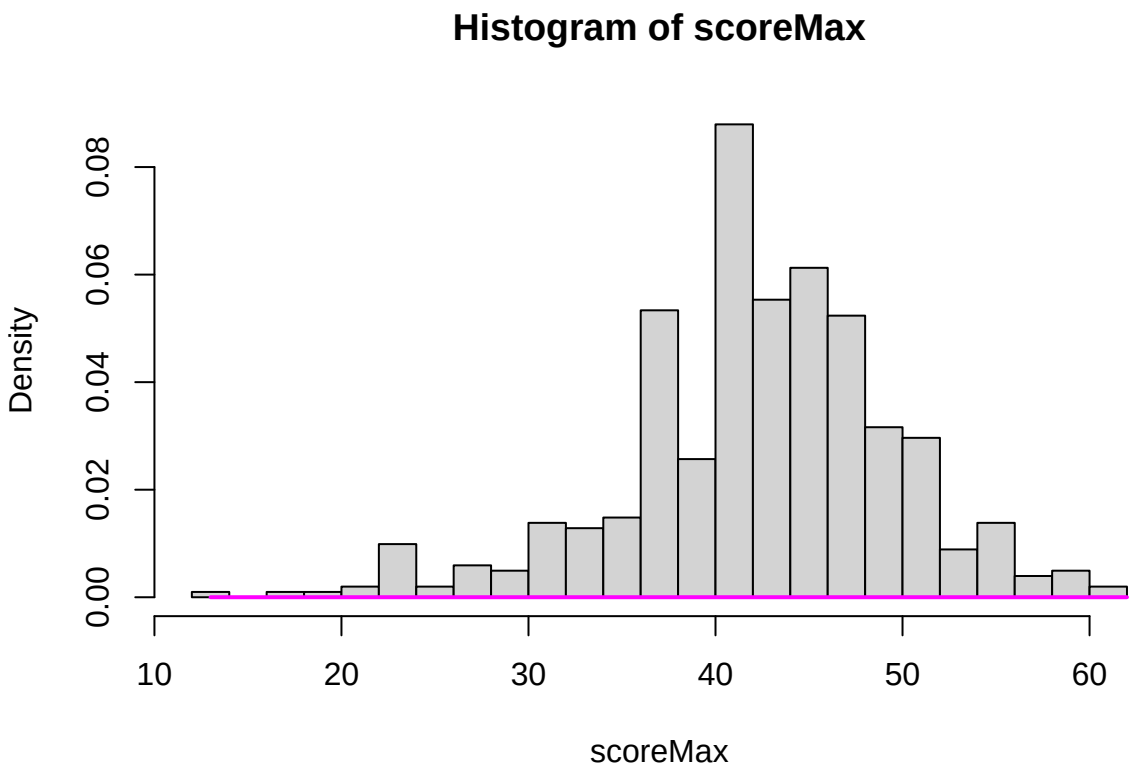
Shape parameter is really big so I am almost positive that it isn't a gumbel. However, when looking at the summary function there isn't SE or Estimated parameter covariance matrix print out in the summary which I don't really understand. This makes me slightly worried that the fit doesn't really work. Which would be unfortunate.

```
outGEV$results$hessian#this prints it out however all of the variables are really small
```

```
##               location         scale         shape
## location -2.212738e-03  6.607287e-06 -1.176181e-04
## scale     6.607287e-06 -1.919047e-08 -9.065574e-06
## shape    -1.176181e-04 -9.065574e-06 -4.178873e-03
```

plotting a hist with the new distribution in order to evaluate. Yeah something happend and it isn't converging correctly for this data. So I am going to try some other models and some other ideas in order to best fit this data. I am going to try a gumbel model as maybe forcing the shape to 0 will help the other variables to converge as well as GP model in maybe more data per point will help out the creation of this model. If not we can always try to fit it by year as well.

```
xGrid <- seq(min(scoreMax), max(scoreMax), length.out = 200)
pars <- outGEV$results$par
GEVpdf <- devd(xGrid, loc = pars[1], scale = pars[2], shape = pars[3])
hist(scoreMax, probability = TRUE, nclass = 20)
lines(xGrid, GEVpdf, col = "magenta", lwd = 2)
```



Gumbel Fit due to forcing the shape parameter to 0 we might be able to converge to a better model

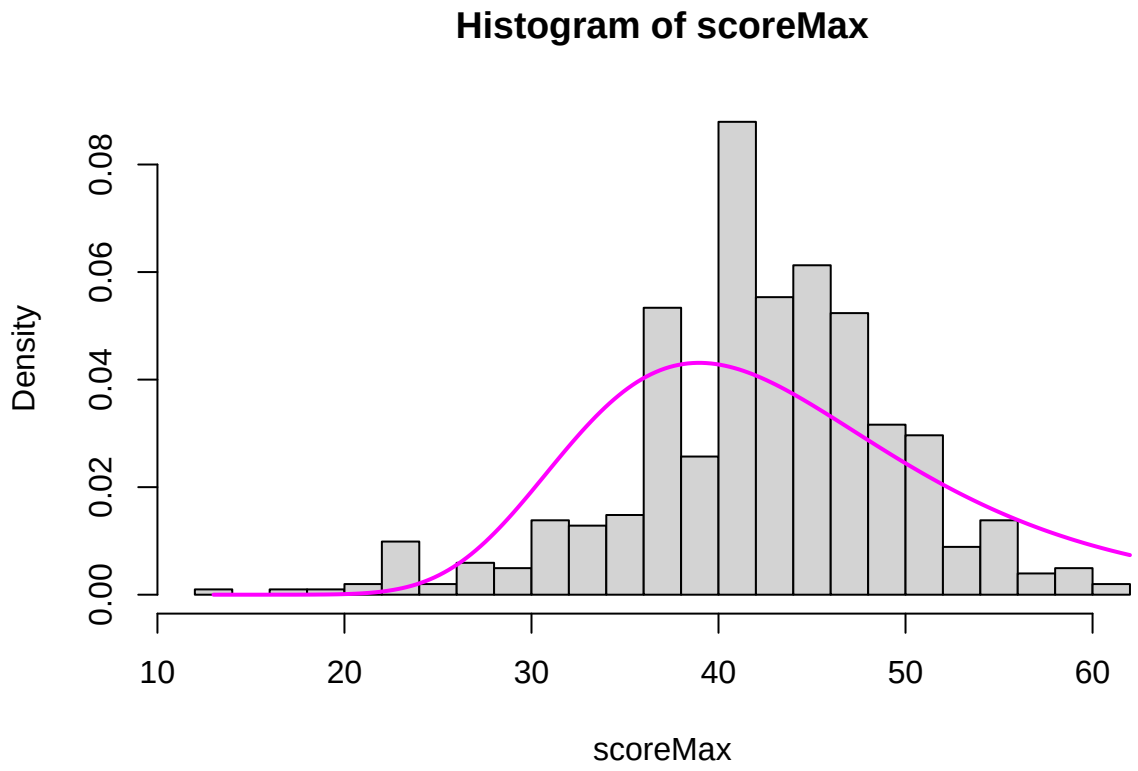
```
outGumbel <- fevd(scoreMax, type = "Gumbel")
summary(outGumbel)#this looks way better than the other model. Maybe the issue was that we don't have e
```

```
##
## fevd(x = scoreMax, type = "Gumbel")
##
## [1] "Estimation Method used: MLE"
##
##
##  Negative Log-Likelihood Value:  1817.483
##
##
##  Estimated parameters:
##  location     scale
## 38.963807  8.531862
##
##  Standard Error Estimates:
##  location     scale
## 0.4034326 0.2489471
##
##  Estimated parameter covariance matrix.
##            location      scale
## location 0.16275784 0.03425359
## scale    0.03425359 0.06197466
```

```
##
##  AIC = 3638.966
##
##  BIC = 3647.419
```

Now we are going to try a new hist plot

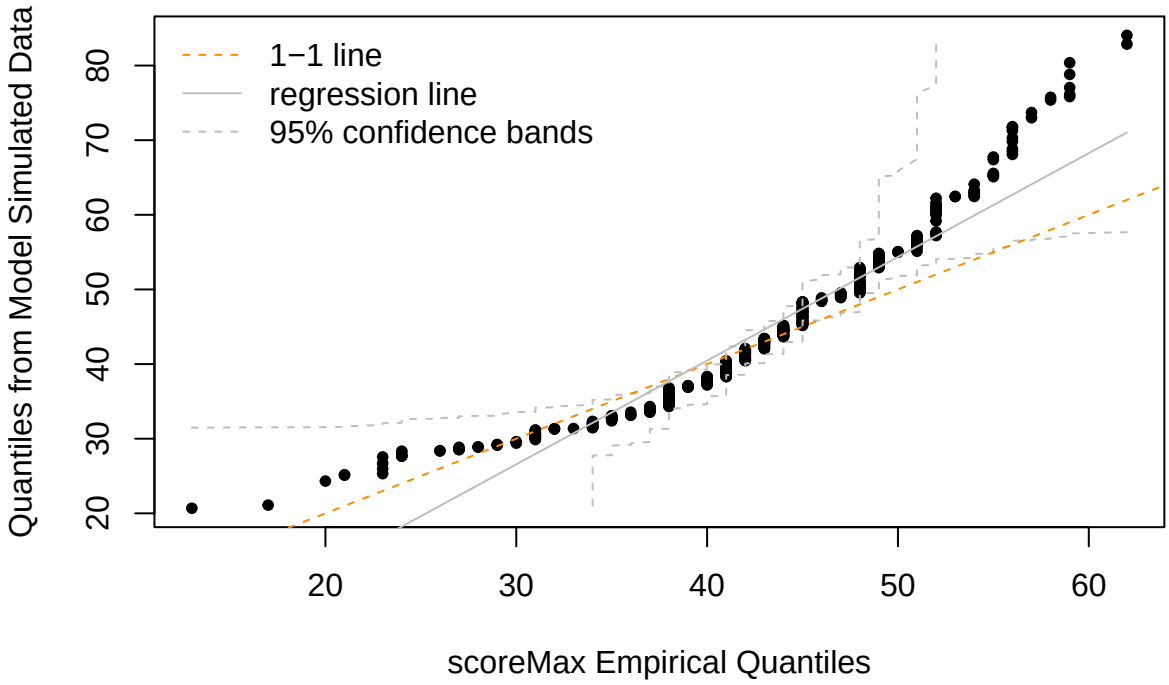
```
xGrid <- seq(min(scoreMax), max(scoreMax), length.out = 200)
pars <- outGumbel$results$par
GEVpdf <- devd(xGrid, loc = pars[1], scale = pars[2], shape = 0)
hist(scoreMax, probability = TRUE, nclass = 20)
lines(xGrid, GEVpdf, col = "magenta", lwd = 2)#honestly fits about as well as a normal distribution. Do
```




```
#at least it fit this time. However, yeah I wouldn't use this model to make predictions
```

Yeah when you look at the qq plot you can see pretty obvious skewing in the data. Maybe we have to also include a trend to address this or transform the maxes if that is a thing that works with the Extreme Value Distribution.

```
plot(outGumbel, type = "qq2")
```



Gumble results

honestly fits about as well as a normal distribution. Doesn't really capture the max that well. Yeah I think our sample size is too small bringing down certain ends more than they should and so they are being over considered. At least it fit this time. However, yeah I wouldn't use this model to make predictions. Yeah after adjusting the shape parameter I am pretty confident that it is the initial values that are dragging down the distribution and the reason it isn't really working is more of a lack of sample size for each weeks game. As most of these really low values will be taken out if you increase the sample size as they would no longer be maximum.

GP model

Choose 42 points for the threshold as it seemed like a good dividing line based off of the initial plot. However, initially was 45 and changed to 49.0 as 49.0 is 7*7 so seven touchdowns. As touchdowns in football go up by 7s so I felt it was a more inclusive threshold than 45. As well the mean score is 22 so that was around double it seems good.

```
outGP0 <- fevd(scores$score, type = "GP", threshold = 42.0)
summary(outGP0)
```

```
##
## fevd(x = scores$score, threshold = 42, type = "GP")
##
```

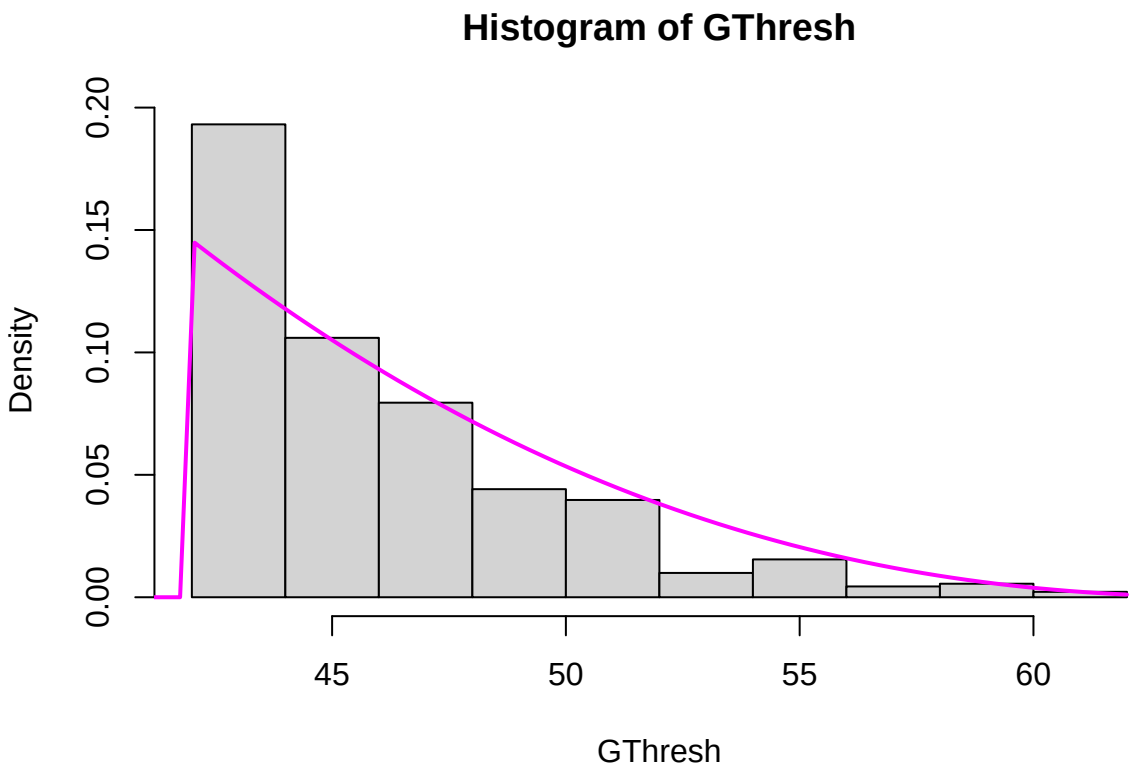
```
## [1] "Estimation Method used: MLE"
##
##
##  Negative Log-Likelihood Value:  963.02
##
##
##  Estimated parameters:
##      scale      shape
##  6.8628307 -0.3020854
##
##  Standard Error Estimates:
##      scale      shape
## 0.41681562 0.03482626
##
##  Estimated parameter covariance matrix.
##              scale        shape
## scale  0.17373526 -0.012264643
## shape -0.01226464  0.001212869
##
##  AIC = 1930.04
##
##  BIC = 1937.851
```

These values seem much much better. Actually seems as though it is converging which is good.

New histogram plot. Just to check to see if it actually seems as the other ones didn't really.

Whenever I subtract the threshold from the data. And you plot the return GP function from the threshold on it honestly doesn't look terrible. Much better even whenever you do 10 bin instead of 20. Really cool to look at though. This actually might work although I definitely massaged the data alot so this most likely needs to be verified. It does under estimate the initial a bit for sure but I wouldn't say that it is horrible.

```
GThresh <- scores$score[scores$score >= 42]
xGrid <- seq(0, max(scoreMax), length.out = 200)
pars <- outGP0$results$par
GPpdf <- devd(xGrid, loc = 42, scale = pars[1], shape = pars[2], type = "GP", threshold = 0)
hist(GThresh, probability = TRUE, nclass = 10)
lines( xGrid, GPpdf, col = "magenta", lwd = 2)
```

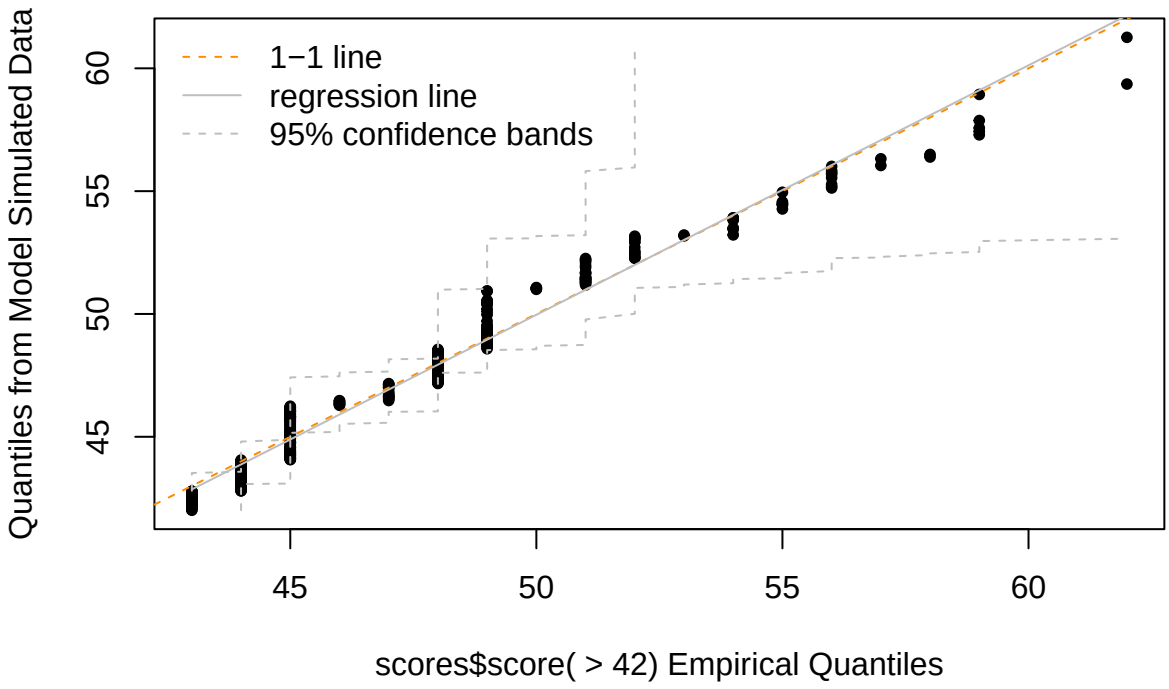


Makes me feel as though a GP function doesn't really map the pdf like the GEV function as it isn't really working. As well whenever I change the values it doesn't really change and it is mainly focused on the tail. Which makes sense as whenever I looked at book documenting the Generalized Pareto Distribution it says it was mainly designed in order to fit tails so it seems as though that is the only thing that it is fitting. So maybe we cannot look at that at this spot. As well I realize I am comparing to the density of score max which isn't used at all for this model so it needs to be compared to something else.

QQ plot

This honestly doesn't look entirely terrible As the data seems to follow the regression line better than I thought. It is a little blocky but that is mainly because it only increases by whole digits. The CI are wide but that is definitely expected with extremes. As we have 500 weeks of extremes here and we can still bootstrap and see what that gives us. First lets find like a 1 in a season game.

```
plot(outGP0, type = "qq2")
```



When we calculate our return level then we scale the model in order to adjust for the correct return level and we can do it in week as there is x weeks in a season.

so this model includes playoffs however there is fewer games for playoffs so it chooses 22 weeks

```
max(scores$week)
```

```
## [1] 22
```

```
length(scores$week[scores$week == 22])#only 2 years with 22 games played as divide by 2 when using scor
```

```
## [1] 4
```

```
scores[scores$week == 22,]#due to change in number of games over time. Going to scale the number of gam
```

```
##              game_id uniqueWeek week season home/away team score
## 6137  2021_22_LA_CIN    2021-22   22   2021 away_team   LA    23
## 6421  2022_22_KC_PHI    2022-22   22   2022 away_team   KC    38
## 12558 2021_22_LA_CIN    2021-22   22   2021 home_team  CIN    20
## 12842 2022_22_KC_PHI    2022-22   22   2022 home_team  PHI    35
```

```
mean_games_season <- mean(table(scores$season))
```

Confidence Intervals

```
season <- 1
look <- return.level(outGP0, season*(mean_games_season), do.ci=TRUE)
look#score by a team
```

```
## fevd(x = scores$score, threshold = 42, type = "GP")
##
## [1] "Normal Approx."
##
## [1] "535.083333333333-year return level: 63.042"
##
## [1] "95% Confidence Interval: (53.4268, 72.6562)"
```

A once in a season game according is predicted to be 63.042 points so rounded so its possible 63 points. with a confidence interval of around (53, 73).

once in a ten season game. Doesn't really do well over 1 season. The once in a 10 season is only 1 point higher after rounding. However the CI is bigger and the lower bound is bigger so in order to find this we are going to have to bootstrap it up.

```
season <- 10
look <- return.level(outGP0, season*(mean_games_season), do.ci=TRUE)
look
```

```
## fevd(x = scores$score, threshold = 42, type = "GP")
##
## [1] "Normal Approx."
##
## [1] "5350.83333333333-year return level: 63.882"
##
## [1] "95% Confidence Interval: (44.9125, 82.8513)"
```

Big Boot Strap Time ::)))

bootstrapping with the GP distribution. We are going to try to calculate the value and the CI for a once in a decade game.

```
#highPrecentChanges is a dataframe that only contains PrecentChanges >= 2. Made it accidentially but di
parTrue <- outGP0$results$par
boot <- 500#start with 500 then we can up it for the final amount
n <- length(scores$score)
parMLE <- matrix(NA, boot, 2)
seasons <- 1

set.seed(420)
for (i in 1:boot){#problem is located in the scale variable. Being estimated way to high

  if( i%%25==0){cat(i, " ")}#looks nice, technically isn't needed

  suppressWarnings(
```

```
    simData <- revd(n, type = "GP", scale=parTrue["scale"], shape= parTrue["shape"], threshold = 0)
  )

  temp <- fevd(simData, type = "GP", threshold = 0)#get the same thing wether or not you do 2.0 or 0.0 j
  parMLE[i,] <- temp$results$par
}
```

```
## 25  50  75  100  125  150  175  200  225  250  275  300  325  350  375  400  425  450  475  500
```

```
Q <- sum(scores$score >= 42) / length(scores)

returnBoot <- rep(NA, boot)
#calculates the confidence interval from the boot results
for (i in 1:boot){
  returnBoot[i] <- qevd( 1/(Q*mean_games_season*seasons) ,threshold=42,
                scale= parMLE[i,1],
                shape= parMLE[i,2], type= "GP",
                lower.tail  =FALSE)
}

mean(returnBoot)
```

```
## [1] 63.72796
```

```
quantile(returnBoot, probs = c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 63.28279 64.16971
```

I don't know if I should add mean_games_season into the qevd() function to be multiples with season and Q. I think I should because that is the number of observations per the season however when it is added there is a very slight difference between the 1 in a season game and the 1 in 10 seasons game. However, that matches more closely to what this data says so I am going to go with it is accurate. So the difference between a 1 in a season game and the 1 in 10 seasons game is 1 point. 63 to 64. However the bootstrap gave us a much better CI for the 1 in 10 seasons game with a range of (64 to 65) points. Rounded so it is possible. While the 1 in a season CI returned a range of (63 to 64). This does makes sense because when you see the tail end of the distribution it falls off exstremely fast.