Project 4

Objective

For this project, I'm utilizing a dataset used in a previous study by Carnegie Melon University where different machine learning models were tested to try to find a way to discriminate between users via their typing rhythms. The main goal of that study was to find a machine learning model with low enough error rates to obtain European Union approval so that it could be used legally and with partnership from European governments. The fields in the dataset include 51 subjects, 8 sessions per subject, 50 repetitions of typing the password: "$.tie5Roanl$", and 31 fields for actions required to type the password. Also, this dataset simulates typing the same password multiple times(50 in this case) in a given day, represented by repetition number, and 8 days in a row, represented by session number. (Killourhy & Maxion, 2009)

I'm starting with a null hypothesis that a person's typing dynamics change over time, short term and long term, which I'm going to attempt to reject. To do this, I will analyze the dataset via a mixed model approach. The response variable I am interested in here is total time to enter the password, while I am interested in the effects of the variables, rep which is the times the password was repeatedly typed in a day, session which is the days of the continuous repetitions of the password, and subject, which is the person typing the password. For this project, rep and session are fixed effects and subject is the nested random effect I will be testing, since rep and session are consistent actions but subject varies depending on who was available to be tested at the time. Assumptions with this dataset are that it has been cleaned and verified of any typos and entry errors.

```
# function to install packages if they don't exist
usePackage <- function(p) {
    if (!is.element(p, installed.packages()[,1]))
        install.packages(p, dep = TRUE)
    require(p, character.only = TRUE)
}
# Load packages used in this project
usePackage("readxl")
```

```
## Loading required package: readxl
```

```
## Warning: package 'readxl' was built under R version 4.1.2
```

```
usePackage("ggplot2")
```

```
## Loading required package: ggplot2
```

```
usePackage("knitr")
```

```
## Loading required package: knitr
```

```
## Warning: package 'knitr' was built under R version 4.1.2
```

```
usePackage("formattable")
```

```
## Loading required package: formattable
```

```
## Warning: package 'formattable' was built under R version 4.1.1
```

```
usePackage("dplyr")
```

```
## Loading required package: dplyr
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## 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
```

```
usePackage("tidyr")
```

```
## Loading required package: tidyr
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
usePackage("lme4")
```

```
## Loading required package: lme4
```

```
## Warning: package 'lme4' was built under R version 4.1.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
usePackage("gee")
```

```
## Loading required package: gee
```

```
## Warning: package 'gee' was built under R version 4.1.2
```

```
usePackage("Matrix")
usePackage("multcomp")
```

```
## Loading required package: multcomp
```

```
## Warning: package 'multcomp' was built under R version 4.1.2
```

```
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 4.1.1
```

```
## Loading required package: survival
```

```
## Loading required package: TH.data
```

```
## Warning: package 'TH.data' was built under R version 4.1.1
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## The following object is masked from 'package:formattable':
##
##     area
```

```
##
## Attaching package: 'TH.data'
```

```
## The following object is masked from 'package:MASS':
##
##     geyser
```

```
usePackage("car")
```

```
## Loading required package: car
```

```
## Warning: package 'car' was built under R version 4.1.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.1.2
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
usePackage("MASS")
usePackage("aod")
```

```
## Loading required package: aod
```

```
## Warning: package 'aod' was built under R version 4.1.2
```

```
##
## Attaching package: 'aod'
```

```
## The following object is masked from 'package:survival':
##
##     rats
```

```
usePackage("DT")
```

```
## Loading required package: DT
```

```
## Warning: package 'DT' was built under R version 4.1.2
```

```
usePackage("broom")
```

```
## Loading required package: broom
```

```
## Warning: package 'broom' was built under R version 4.1.2
```

```
#Before being able to access the file, I had to open it in Excel and save it without making any changes. Not really sure why
but doing this allowed me to open the file in R to add its contents to a dataframe
PasswordData.df <- as.data.frame(read_excel("DSL-StrongPasswordData.xls", sheet="Sheet1"))


#Here took a peak at the dataset, viewing the top 6 rows and a summary of the data

head(PasswordData.df)
```

	subject <chr>	sessionIndex <dbl>	rep <dbl>	H.period <dbl>	DD.period.t <dbl>	UD.period.t <dbl>	H.t <dbl>	DD.t.i <dbl>	UD.t.i <dbl>
1	s002	1	1	0.1491	0.3979	0.2488	0.1069	0.1674	0.0605
2	s002	1	2	0.1111	0.3451	0.2340	0.0694	0.1283	0.0589
3	s002	1	3	0.1328	0.2072	0.0744	0.0731	0.1291	0.0560
4	s002	1	4	0.1291	0.2515	0.1224	0.1059	0.2495	0.1436
5	s002	1	5	0.1249	0.2317	0.1068	0.0895	0.1676	0.0781
6	s002	1	6	0.1394	0.2343	0.0949	0.0813	0.1299	0.0486

6 rows | 1-10 of 35 columns

```
summary(PasswordData.df)
```

```
##    subject          sessionIndex       rep         H.period
##  Length:20400       Min.   :1.00   Min.   : 1.0   Min.   :0.00140
##  Class :character   1st Qu.:2.75   1st Qu.:13.0   1st Qu.:0.07440
##  Mode  :character   Median :4.50   Median :25.5   Median :0.08950
##                     Mean   :4.50   Mean   :25.5   Mean   :0.09338
##                     3rd Qu.:6.25   3rd Qu.:38.0   3rd Qu.:0.10790
##                     Max.   :8.00   Max.   :50.0   Max.   :0.37610
##   DD.period.t       UD.period.t          H.t              DD.t.i
##  Min.   : 0.0187   Min.   :-0.2358   Min.   :0.00930   Min.   :0.0011
##  1st Qu.: 0.1469   1st Qu.: 0.0498   1st Qu.:0.06600   1st Qu.:0.1136
##  Median : 0.2059   Median : 0.1087   Median :0.08100   Median :0.1404
##  Mean   : 0.2641   Mean   : 0.1708   Mean   :0.08573   Mean   :0.1691
##  3rd Qu.: 0.3064   3rd Qu.: 0.2124   3rd Qu.:0.09980   3rd Qu.:0.1839
##  Max.   :12.5061   Max.   :12.4517   Max.   :0.24110   Max.   :4.9197
##      UD.t.i             H.i              DD.i.e            UD.i.e
##  Min.   :-0.16210   Min.   :0.00320   Min.   : 0.0014   Min.   :-0.16000
##  1st Qu.: 0.02720   1st Qu.:0.06200   1st Qu.: 0.0893   1st Qu.: 0.00740
##  Median : 0.05780   Median :0.07710   Median : 0.1209   Median : 0.04120
##  Mean   : 0.08336   Mean   :0.08157   Mean   : 0.1594   Mean   : 0.07781
##  3rd Qu.: 0.09640   3rd Qu.:0.09690   3rd Qu.: 0.1731   3rd Qu.: 0.09340
##  Max.   : 4.79990   Max.   :0.33120   Max.   :25.9873   Max.   :25.91580
##       H.e            DD.e.five        UD.e.five          H.five
##  Min.   :0.00210   Min.   :0.0013   Min.   :-0.1505   Min.   :0.0014
##  1st Qu.:0.06860   1st Qu.:0.2166   1st Qu.: 0.1332   1st Qu.:0.0610
##  Median :0.08340   Median :0.2890   Median : 0.2004   Median :0.0742
##  Mean   :0.08914   Mean   :0.3774   Mean   : 0.2883   Mean   :0.0769
##  3rd Qu.:0.10270   3rd Qu.:0.4568   3rd Qu.: 0.3694   3rd Qu.:0.0906
##  Max.   :0.32540   Max.   :4.9618   Max.   : 4.8827   Max.   :0.1989
##  DD.five.Shift.r  UD.five.Shift.r    H.Shift.r        DD.Shift.r.o
##  Min.   :0.1694   Min.   :0.0856   Min.   :0.00140   Min.   :0.0494
##  1st Qu.:0.3079   1st Qu.:0.2297   1st Qu.:0.07020   1st Qu.:0.1565
##  Median :0.3775   Median :0.3020   Median :0.09350   Median :0.2014
##  Mean   :0.4389   Mean   :0.3620   Mean   :0.09594   Mean   :0.2509
##  3rd Qu.:0.4860   3rd Qu.:0.4089   3rd Qu.:0.11670   3rd Qu.:0.2834
##  Max.   :8.3702   Max.   :8.2908   Max.   :0.28170   Max.   :4.1523
##   UD.Shift.r.o         H.o              DD.o.a          UD.o.a
##  Min.   :-0.0865   Min.   :0.00690   Min.   :0.0012   Min.   :-0.22870
##  1st Qu.: 0.0547   1st Qu.:0.07150   1st Qu.:0.1064   1st Qu.: 0.01700
##  Median : 0.1022   Median :0.08630   Median :0.1316   Median : 0.04440
##  Mean   : 0.1550   Mean   :0.08835   Mean   :0.1569   Mean   : 0.06858
##  3rd Qu.: 0.1910   3rd Qu.:0.10190   3rd Qu.:0.1676   3rd Qu.: 0.08030
##  Max.   : 4.0120   Max.   :0.68720   Max.   :2.8567   Max.   : 2.81520
##       H.a            DD.a.n          UD.a.n             H.n
##  Min.   :0.0040   Min.   :0.0011   Min.   :-0.23550   Min.   :0.0037
##  1st Qu.:0.0821   1st Qu.:0.0961   1st Qu.:-0.00900   1st Qu.:0.0673
##  Median :0.1019   Median :0.1250   Median : 0.02270   Median :0.0853
##  Mean   :0.1063   Mean   :0.1507   Mean   : 0.04441   Mean   :0.0899
##  3rd Qu.:0.1223   3rd Qu.:0.1746   3rd Qu.: 0.06890   3rd Qu.:0.1079
##  Max.   :2.0353   Max.   :3.3278   Max.   : 2.52420   Max.   :0.3577
##      DD.n.l           UD.n.l             H.l           DD.l.Return
##  Min.   :0.0013   Min.   :-0.1758   Min.   :0.00370   Min.   :0.0083
##  1st Qu.:0.1276   1st Qu.: 0.0235   1st Qu.:0.07740   1st Qu.:0.2100
##  Median :0.1725   Median : 0.0955   Median :0.09370   Median :0.2630
##  Mean   :0.2026   Mean   : 0.1127   Mean   :0.09559   Mean   :0.3218
##  3rd Qu.:0.2288   3rd Qu.: 0.1457   3rd Qu.:0.11110   3rd Qu.:0.3502
##  Max.   :4.0252   Max.   : 3.9782   Max.   :0.34070   Max.   :5.8836
##   UD.l.Return        H.Return
##  Min.   :-0.1245   Min.   :0.00290
##  1st Qu.: 0.1141   1st Qu.:0.06990
##  Median : 0.1603   Median :0.08550
##  Mean   : 0.2263   Mean   :0.08831
##  3rd Qu.: 0.2551   3rd Qu.:0.10370
##  Max.   : 5.8364   Max.   :0.26510
```

Exploratory Analysis

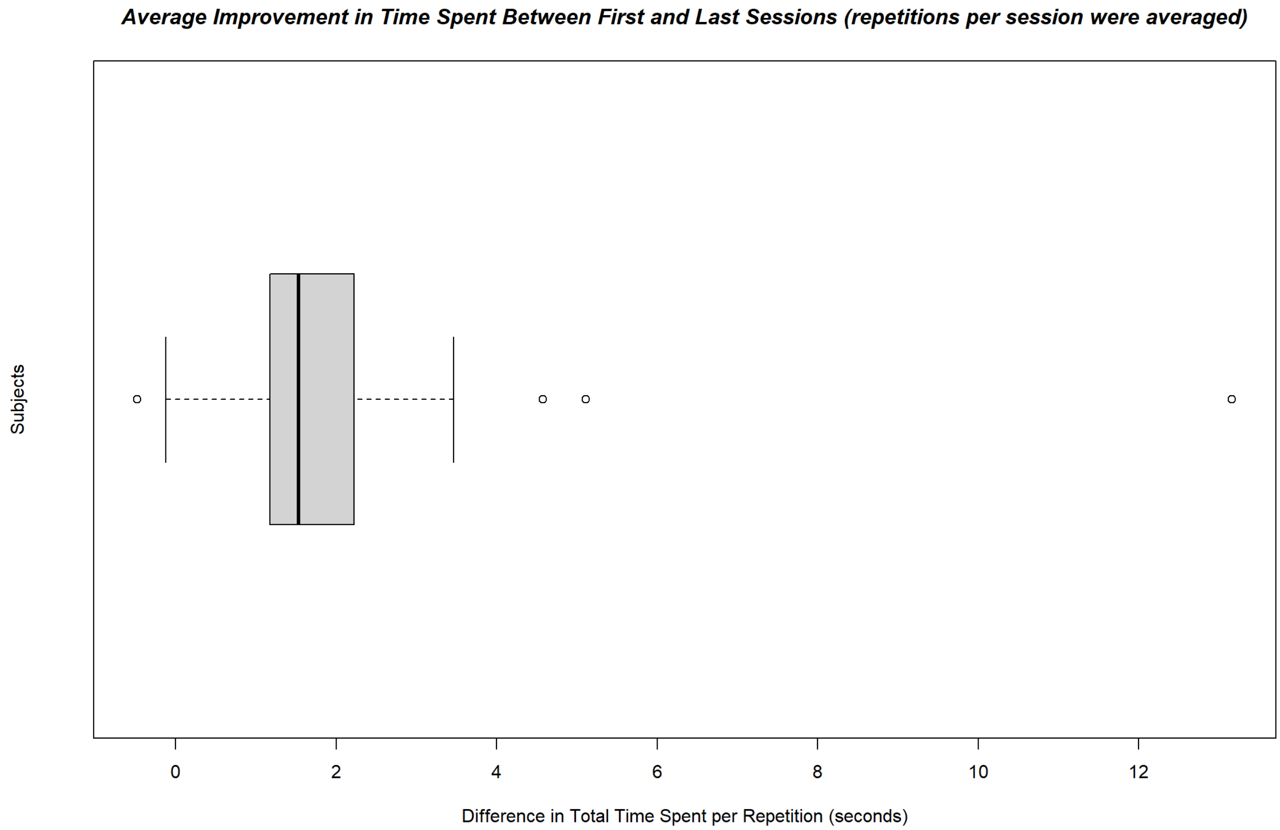
```
#First, in order to explore the dataset, I'm going to create a few new datasets with the original dataset so that I can crea
te visualizations of the data.



#This dataset, PasswordDataEA1.df, sums all of the password input actions to create a TotalTime column. Also, all of the pas
sword input action columns are removed.
PasswordDataEA1.3.df <- PasswordData.df
PasswordDataEA1.3.df$TotalTime <- rowSums(PasswordData.df[ , 4:ncol(PasswordData.df)])
PasswordDataEA1.df <- PasswordDataEA1.3.df[,c("subject","sessionIndex","rep","TotalTime")]


#This dataset, PasswordDataEA2.df, groups the original dataset by subject and sessionIndex and obtains the mean of each pass
word input action under each subject/session group.
PasswordDataEA2.df <- PasswordData.df

PasswordDataEA2.df <- PasswordDataEA2.df %>%
  group_by(subject, sessionIndex) %>%
  summarise(across(H.period:H.Return, mean, .names = "{col}_mean"))
```

```
## `summarise()` has grouped output by 'subject'. You can override using the
## `.groups` argument.
```

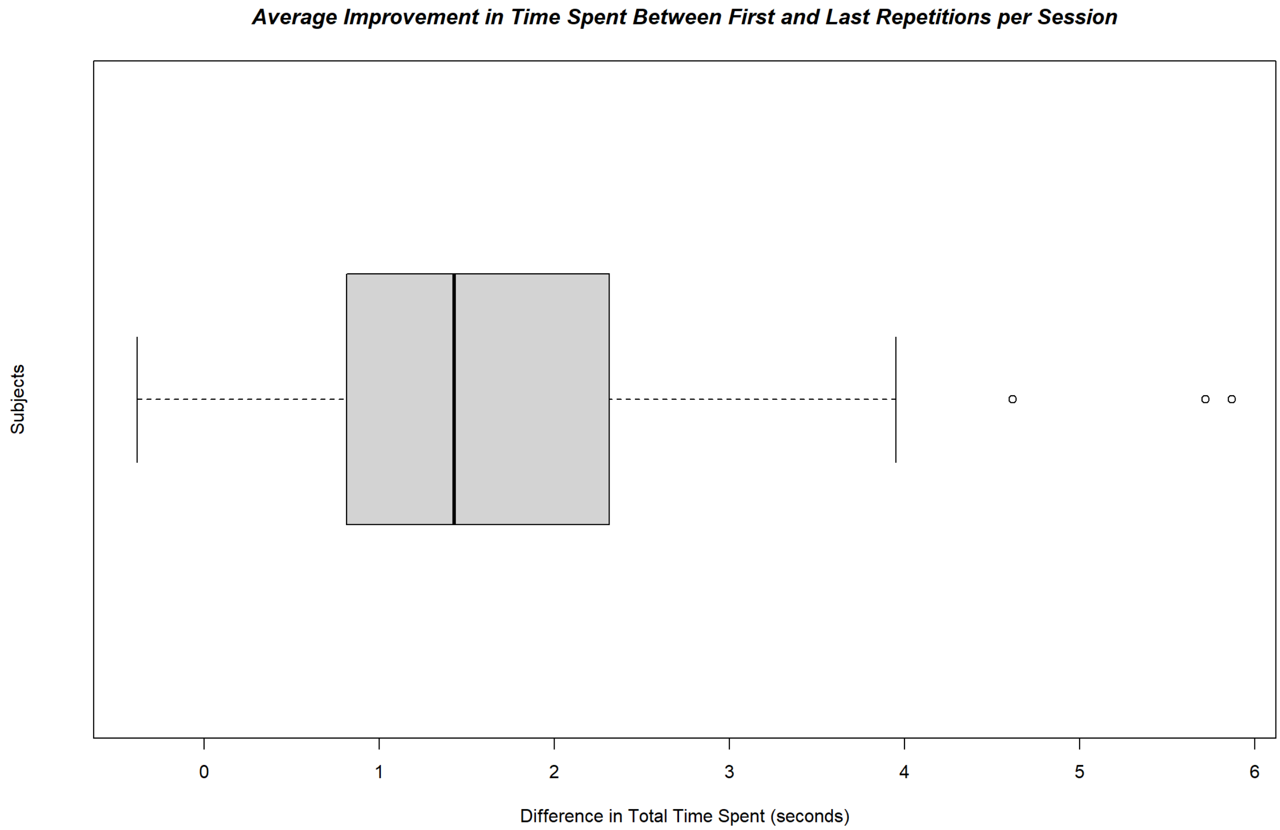
```
#This dataset, PasswordData_long.df, is just a long form version of the original dataset with just 5 columns: subject, sessi
onIndex, rep, action, time_spent
PasswordData_temp.df <- PasswordData.df
PasswordDataCols.df <- colnames(PasswordData.df)
PasswordData_long.df <-  PasswordData_temp.df %>%                                  # Apply pivot_longer function
  pivot_longer(PasswordDataCols.df[4:34], names_to = "action", values_to = "time_spent")


#This dataset, PasswordDataEA1.1.df, takes the dataset I created earlier, PasswordDataEA1.df, which obtained the total times
per repetition, groups it by subject and sessionIndex, and obtains the mean of total times per each subject/session group
PasswordDataEA1.1.df <- PasswordDataEA1.df %>%
  group_by(subject, sessionIndex) %>%
  summarise(Mean_TotalTime = mean(TotalTime))
```

```
## `summarise()` has grouped output by 'subject'. You can override using the
## `.groups` argument.
```

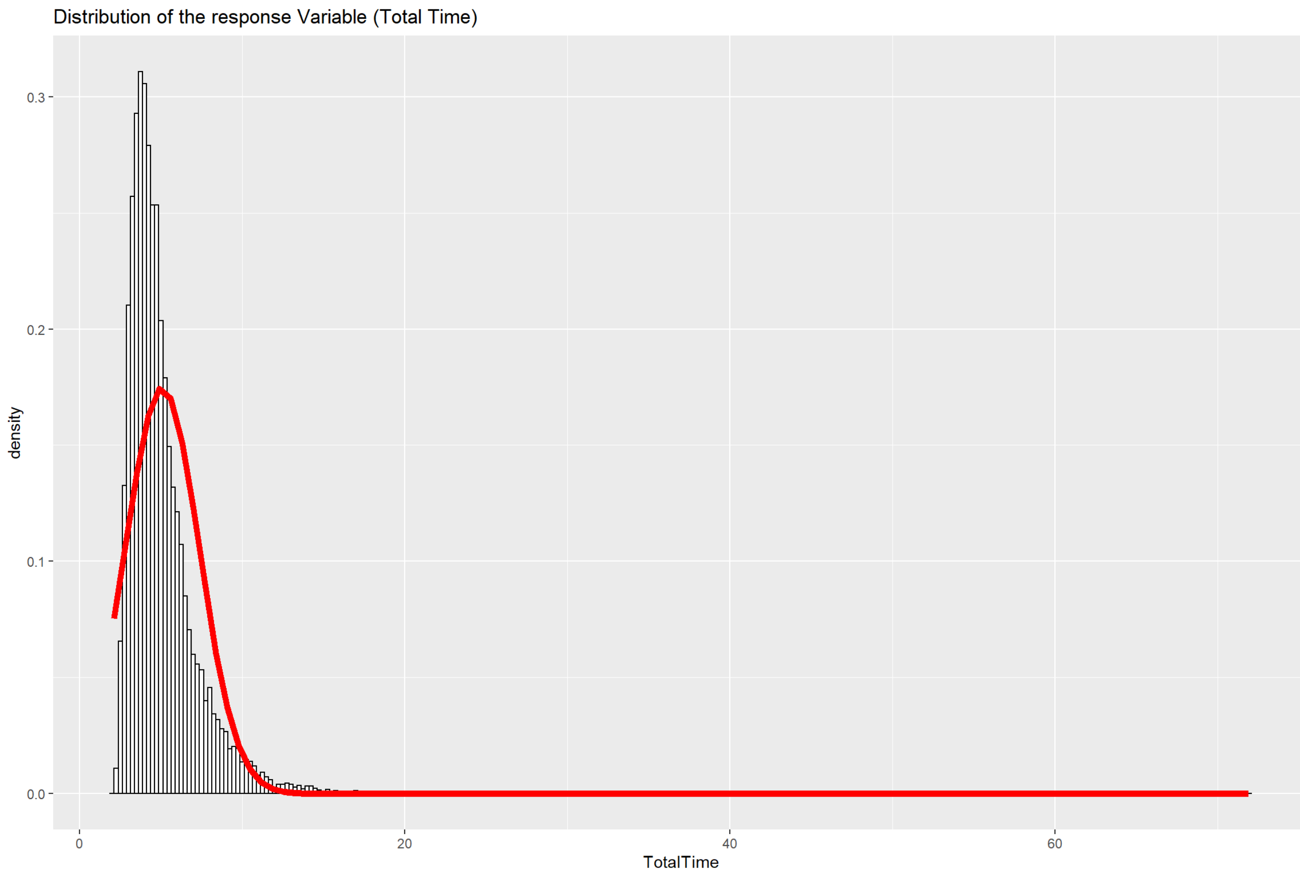
```
#This dataset, PasswordDataEA1.2.df, takes PasswordDataEA1.1.df and obtains the difference in average total time spent per r
epetition between the last session and the first session for each subject
PasswordDataEA1.2.df <- PasswordDataEA1.1.df %>%
  group_by(subject) %>%
  mutate(DiffBtwnLastFirst = Mean_TotalTime[sessionIndex == 1] - Mean_TotalTime[sessionIndex == 8])

PasswordDataEA1.2.df <- unique(PasswordDataEA1.2.df[,c(1,4)])


#This dataset, PasswordDataEA1.3.df, includes subject, sessionIndex, the difference between the first and last repetitions o
f each session
PasswordDataEA1.3.df <- PasswordDataEA1.df %>%
  group_by(subject,sessionIndex) %>%
  mutate(DiffBtwnLastFirst = TotalTime[rep == 1] - TotalTime[rep == 50])

PasswordDataEA1.3.df <- unique(PasswordDataEA1.3.df[,c(1,2,5)])

PasswordDataEA1.3.df <- PasswordDataEA1.3.df %>%
  group_by(subject) %>%
    summarise(DiffBtwnLastFirst = mean(DiffBtwnLastFirst))

#Datasets created

 # PasswordData.df
 #
 # PasswordDataEA1.df
 #
 # PasswordDataEA1.1.df
 #
 # PasswordDataEA1.2.df
 #
 # PasswordDataEA2.df
 #
 # PasswordDataEA1.3.df
 #
 # PasswordData_long.df



#Visualizations
#This box plot shows the difference in total time spent to type the password between the first and last repetition per sessi
on to show what improvement is seen within each session by each subject
boxplot(PasswordDataEA1.2.df$DiffBtwnLastFirst, density = 20,
        legend.text = rownames(PasswordDataEA1.2.df$subject), horizontal = TRUE, xlab = "Difference in Total Time Spent per
Repetition (seconds)", ylab = "Subjects")
title(main = list("Average Improvement in Time Spent Between First and Last Sessions (repetitions per session were average
d)", font = 4))
```




```
#This box plot shows the average difference in total time spent to type the password between the first and last session of e
ach subject
boxplot(PasswordDataEA1.3.df$DiffBtwnLastFirst, density = 20,
        legend.text = rownames(PasswordDataEA1.3.df), horizontal = TRUE, xlab = "Difference in Total Time Spent (seconds)",
ylab = "Subjects")
title(main = list("Average Improvement in Time Spent Between First and Last Repetitions per Session", font = 4))
```




```
#This histogram with overlayed density plot shows the distribution of data and whether the Response Variable makes up a norm
al distribution
ggplot(PasswordDataEA1.df, aes(x=TotalTime)) +
            geom_histogram(aes(y = ..density..), binwidth=.25, colour="black", fill="white") +
            stat_function(fun = dnorm, lwd = 2, col = 'red',
                          args = list(mean = mean(PasswordDataEA1.df$TotalTime), sd = sd(PasswordDataEA1.df$TotalTime)))  +
    labs(title = "Distribution of the response Variable (Total Time)")
```



My exploratory analysis includes box plots showing the average improvement of the subjects between their first and last sessions as well as first and last repetition per session. These plots show that there are outliers in the response variable of the dataset where some subjects committed actions during the typing of the password that were much slower or faster than they usually would, or in some cases, much slower or faster than other subjects. Furthermore, these plots show that on average, there is a change in speed, almost always an improvement, between consecutive sessions and between consecutive repetitions. In fact, between the first and last sessions of the subjects, there is an average improvement of almost 2 seconds. Finally, I created a histogram overlayed with a density plot to check whether the data is normally distributed, which it seems to not be as it's too heavily right-skewed.

For this project, as the outliers can be attributed to the randomness of the subjects, since people often make mistakes when typing ambiguous passwords, and the data is assumed to have been cleaned and verified, I have decided not to remove outliers.

Furthermore, I decided to use penalized quasi-likelihood along with random intercept and random intercept and slope models to model the data. This technique allows for the fitting of data to mixed effect models when the data is not of a normal distribution. It is an approximate inference technique that allows estimation of model parameters without knowledge of the error distribution of the response variable. (Everitt & Hothorn, 2014) More common methods such as restricted maximum likelihood and maximum likelihood require that the data be normally distributed. (Pilowsky, 2018) Penalized quasi-likelihood just requires that the response variable not have a mean less than five and the response variable does not fit a discrete count distribution such as Poisson or Binomial, or that the response variable is not binary. (Pilowsky, 2018) The response variable's mean is greater than 5 and is not binary so this technique should work.

Penalized Quasi-Likelihood:

$$\hat{g}_n \in \arg\max_{g \in \delta} [\hat{Q}_n(F(g)) - \lambda_n^2 J^2(g)]$$

Random intercept and random intercept and slope models are commonly used as opposed to generalized linear models for datasets with repeated measurements. (Everitt & Hothorn, 2014) These models assume that correlation amongst independent repeated measurements on the same unit arises from the shared unobserved variables and that time has a fixed effect. (Everitt & Hothorn, 2014) The difference between random intercept and random intercept and slope models are that random intercept models measure dissimilarity in intercepts while random intercept and slope models measure dissimilarity in intercepts and slopes. (Everitt & Hothorn, 2014)

Random Intercept Model:

$$y_{ij} = \beta_0 + \beta_1 t_j + u_i + \varepsilon_{ij}$$

y_ij = observation made t_j = time i = individual

Random Intercept and Slope Model:

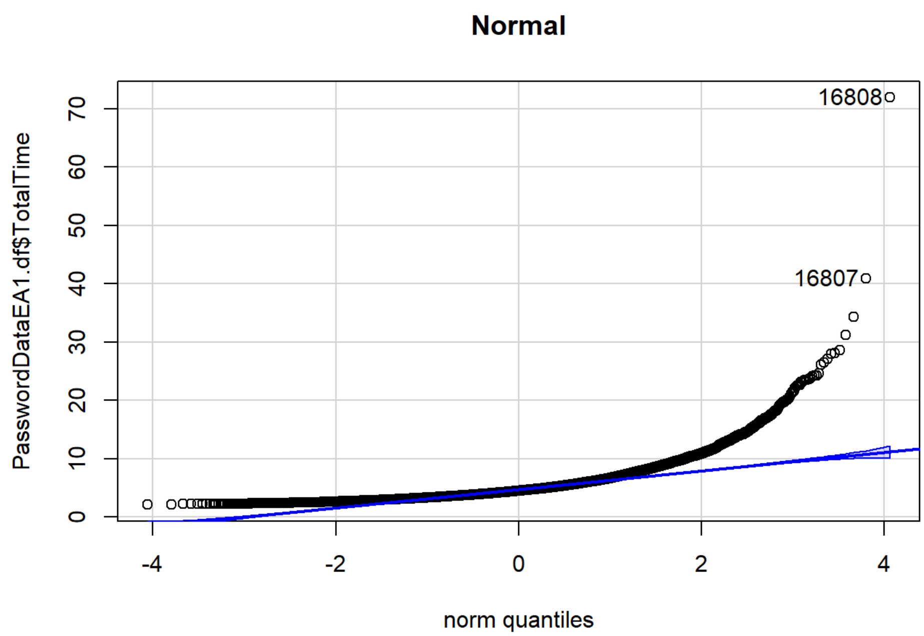
$$y_{ij} = \beta_0 + \beta_1 t_j + u_i + v_i t_j + \varepsilon_{ij}$$

u_i = intercepts v_i = slopes

Sources: (Everitt & Hothorn, 2014) (Mammen & van de Gee, 1997)

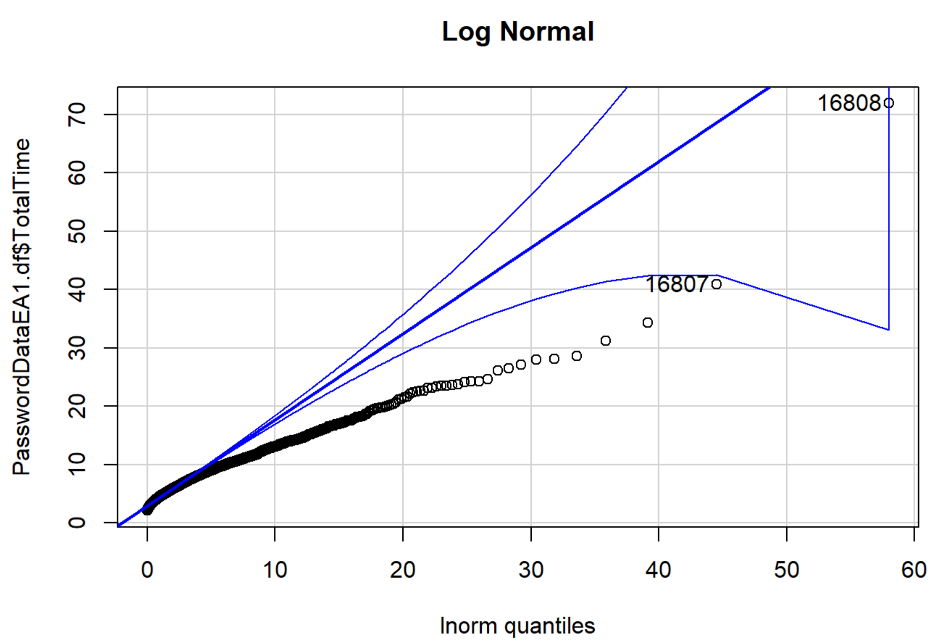
Decide what probability distribution best fits the dataset via quantile comparison plots

```
# normal
qqp(PasswordDataEA1.df$TotalTime, "norm", main= "Normal")
```




```
## [1] 16808 16807
```

```
# lognormal
qqp(PasswordDataEA1.df$TotalTime, "lnorm", main= "Log Normal")
```

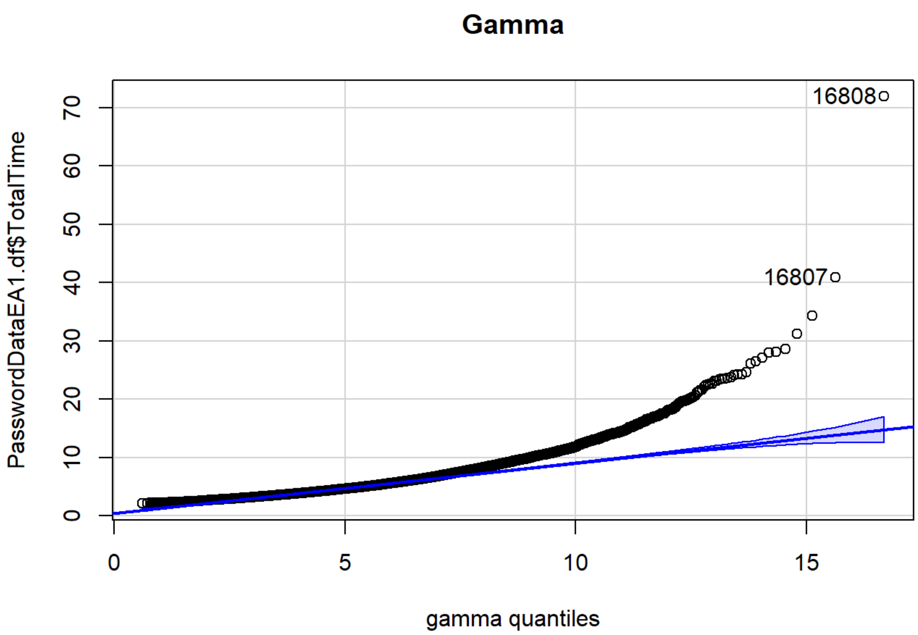


```
## [1] 16808 16807
```

```
# gamma
gamma <- fitdistr(PasswordDataEA1.df$TotalTime, "gamma")
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
qqp(PasswordDataEA1.df$TotalTime, "gamma", shape = gamma$estimate[[1]], rate = gamma$estimate[[2]], main= "Gamma")
```

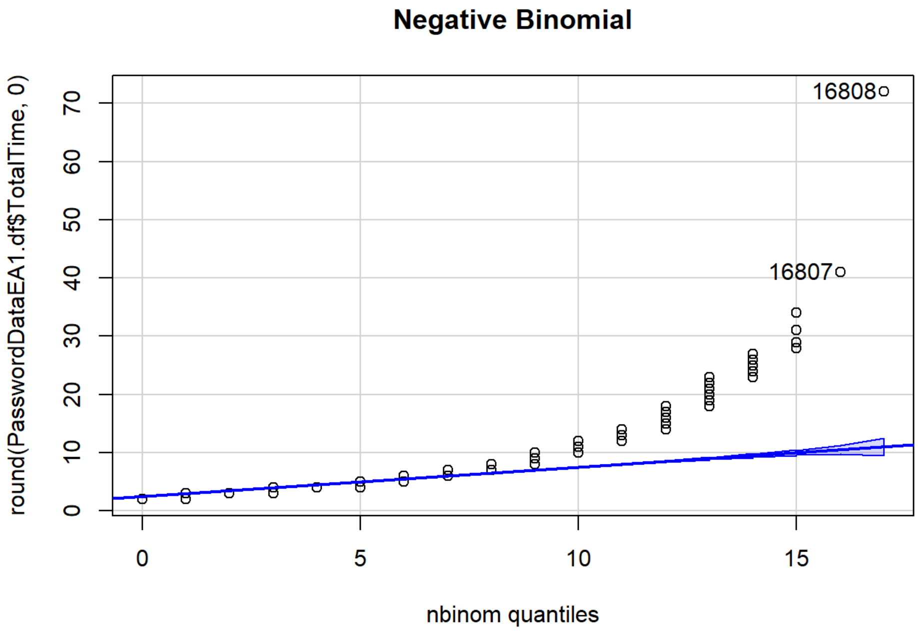



```
## [1] 16808 16807
```

```
# negative binomial
nbinom <- fitdistr(round(PasswordDataEA1.df$TotalTime,0), "Negative Binomial")
```

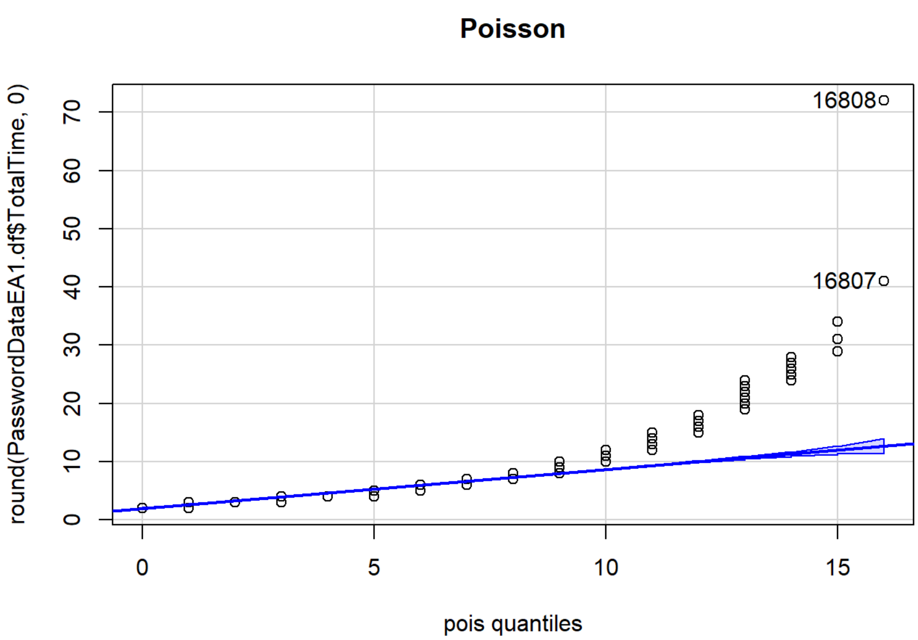
```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
qqp(round(PasswordDataEA1.df$TotalTime,0), "nbinom", size = nbinom$estimate[[1]], mu = nbinom$estimate[[2]], main= "Negative
Binomial")
```




```
## [1] 16808 16807
```

```
# Poisson
poisson <- fitdistr(round(PasswordDataEA1.df$TotalTime,0), "Poisson")
qqp(round(PasswordDataEA1.df$TotalTime,0), "pois", lambda = poisson$estimate, main= "Poisson")
```




```
## [1] 16808 16807
```

Next I utilized quantile comparison plots to check what probability distribution best fits the response variable. I tested with Normal, Lognormal, Gamma, Negative Binomial and Poisson. I decided to fit the dataset to a Gamma distribution as although Poisson seemed like a slightly closer fit, it also would require a transformation of the response variable from continuous to discrete, or in other words rounding to whole numbers.

Statistical Analysis

Here I fitted the dataset to a gamma model via penalized quasi-likelihood. I tested fit with a random intercept model as well as a random intercept and slope model. (Arbor Custom Analytics, 2020) Also, these were tested with identity and inverse link functions. The GLMMPQL function does not output AIC, BIC, or Log Likelihood and thus I tested fit via residuals charts. These charts showed that the random intercept model with the identity link function and the random intercept model with the identity link function are tied or close to tied for having the closest fit to the dataset.

```
#Testing the models
#Random intercept models with a random effect term consisting of a slope and cluster term
#Gamma Distribution with Identity Link Function
PQL1 <- glmmPQL(TotalTime ~ sessionIndex + rep, ~1 | subject, family=Gamma(link=identity), data = PasswordDataEA1.df, verbos
e = FALSE)
summary(PQL1)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: PasswordDataEA1.df
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | subject
##         (Intercept)  Residual
## StdDev:    1.681695 0.2156055
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects:  TotalTime ~ sessionIndex + rep
##                  Value  Std.Error    DF   t-value p-value
## (Intercept)   6.080892 0.23638450 20347  25.72458       0
## sessionIndex -0.186639 0.00293549 20347 -63.58006       0
## rep          -0.007104 0.00046170 20347 -15.38636       0
##  Correlation:
##              (Intr) sssnIn
## sessionIndex -0.061
## rep          -0.051 -0.006
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.0596331 -0.6027234 -0.2148257  0.3582809 26.8555520
##
## Number of Observations: 20400
## Number of Groups: 51
```

```
#Gamma Distribution with Inverse Link Function
PQL2 <- glmmPQL(TotalTime ~ sessionIndex + rep, ~1 | subject, family=Gamma(link=inverse), data = PasswordDataEA1.df, verbose
= FALSE)
summary(PQL2)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: PasswordDataEA1.df
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | subject
##         (Intercept)  Residual
## StdDev:   0.0699077 0.4294586
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects:  TotalTime ~ sessionIndex + rep
##                   Value   Std.Error    DF   t-value p-value
## (Intercept)  0.08132964 0.009851592 20347   8.25548       0
## sessionIndex 0.02501722 0.000200294 20347 124.90233       0
## rep          0.00130749 0.000011767 20347 111.11277       0
##  Correlation:
##              (Intr) sssnIn
## sessionIndex -0.090
## rep          -0.084  0.854
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -17.21917631  -0.50295919  -0.04113114   0.32990583   2.34407934
##
## Number of Observations: 20400
## Number of Groups: 51
```

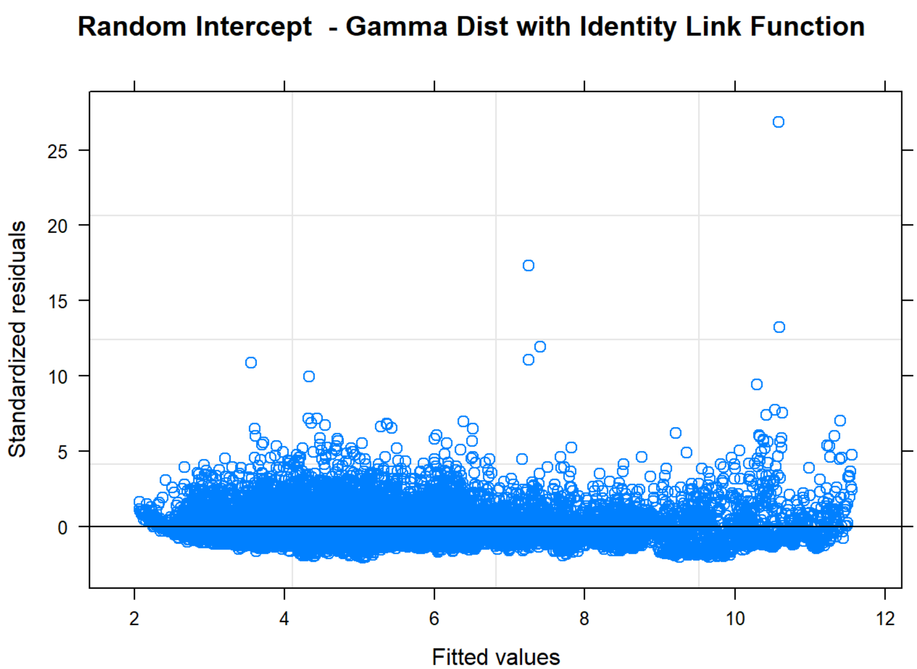
```
#Random Intercept and Slope models with random slope term for rep
#Gamma Distribution with Identity Link Function
PQL3 <- glmmPQL(TotalTime ~ sessionIndex + rep,~1+rep|subject, family=Gamma(link=identity), data = PasswordDataEA1.df, verbo
se = FALSE)
summary(PQL3)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: PasswordDataEA1.df
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 + rep | subject
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev     Corr
## (Intercept) 1.95131724 (Intr)
## rep         0.01173108 -0.904
## Residual    0.21255575
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects:  TotalTime ~ sessionIndex + rep
##                  Value  Std.Error    DF   t-value p-value
## (Intercept)   6.191267 0.27407585 20347  22.58961       0
## sessionIndex -0.186821 0.00289161 20347 -64.60783       0
## rep          -0.011385 0.00171970 20347  -6.62033       0
##  Correlation:
##              (Intr) sssnIn
## sessionIndex -0.052
## rep          -0.875 -0.001
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.0665392 -0.6019761 -0.2115863  0.3536210 25.6716237
##
## Number of Observations: 20400
## Number of Groups: 51
```

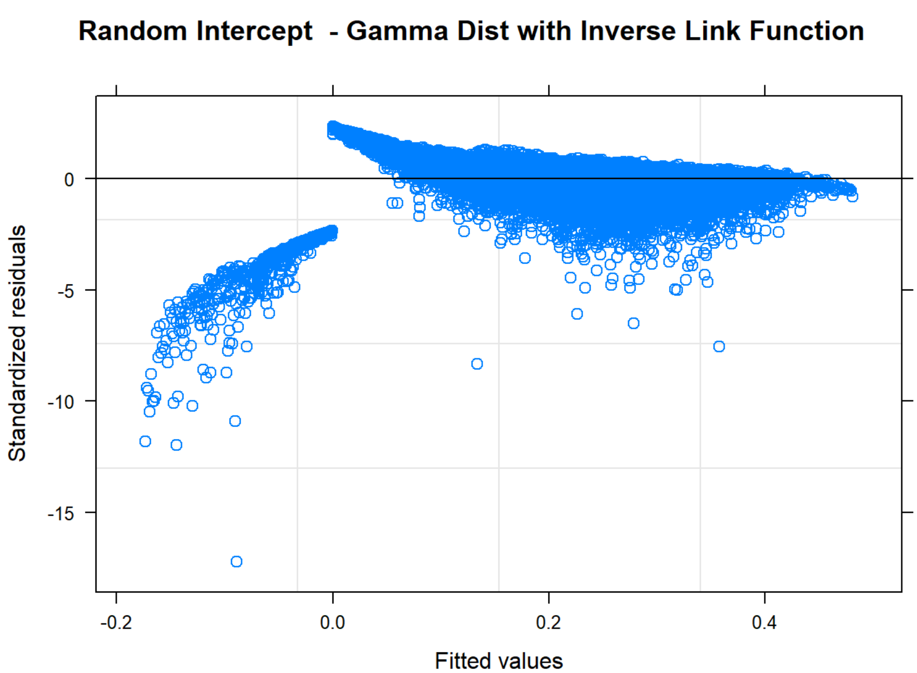
```
#Gamma Distribution with Inverse Link Function
PQL4 <- glmmPQL(TotalTime ~ sessionIndex + rep,~1+rep|subject, family=Gamma(link=inverse), data = PasswordDataEA1.df, verbos
e = FALSE)
summary(PQL4)$Value
```

```
## NULL
```

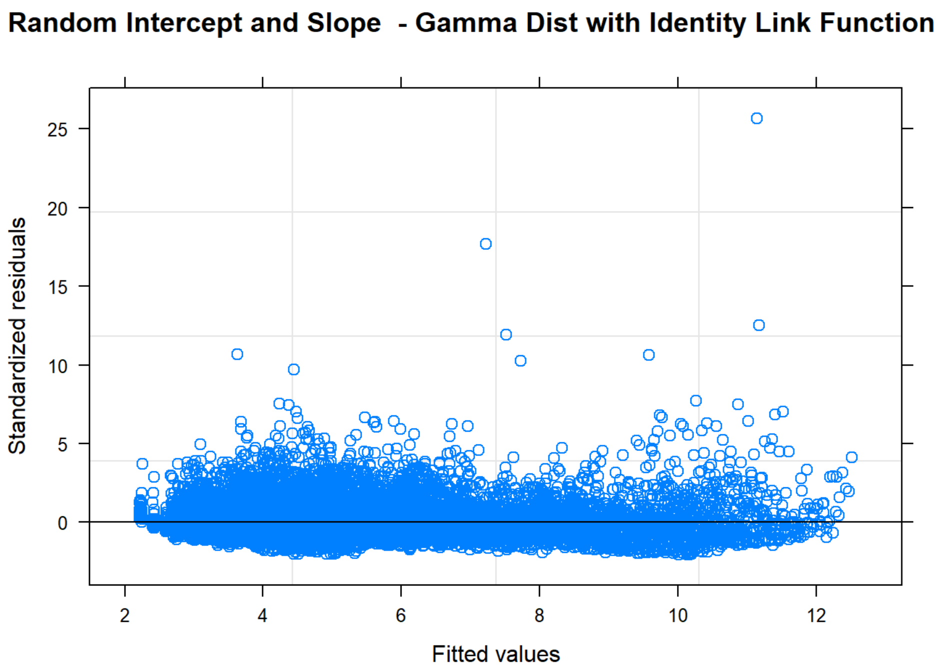
```
#Plot the residuals to check which models best fit the data
plot(PQL1, main = "Random Intercept  - Gamma Dist with Identity Link Function")
```



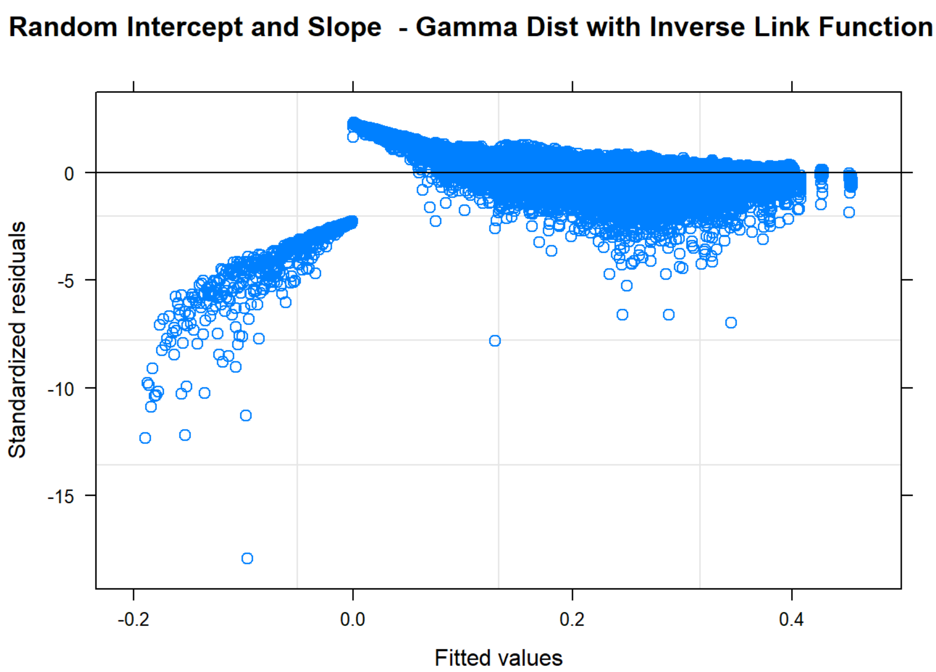

```
plot(PQL2, main = "Random Intercept  - Gamma Dist with Inverse Link Function")
```




```
plot(PQL3, main = "Random Intercept and Slope  - Gamma Dist with Identity Link Function")
```




```
plot(PQL4, main = "Random Intercept and Slope  - Gamma Dist with Inverse Link Function")
```



```
#Results Table of final model results with a highlight on the chosen models
fixed.effect <- c('TotalTime ~ sessionIndex + rep','TotalTime ~ sessionIndex + rep','TotalTime ~ sessionIndex + rep','TotalT
ime ~ sessionIndex + rep')
random.effect <- c('1 | subject','1 | subject','1+rep|subject','1+rep|subject')
family <- c('Gamma(link=identity)','Gamma(link=inverse)','Gamma(link=identity)','Gamma(link=inverse)')
fixed.vars <- c('sessionIndex / rep', 'sessionIndex / rep', 'sessionIndex / rep', 'sessionIndex / rep')
coefficient.estimate <- c('-0.186639 / -0.007104', '0.02501722 / 0.00130749', '-0.186821    / -0.011385', '0.02638535 / 0.00
050069')
std.error <- c('0.00293549 / 0.00046170','0.000200294 / 0.000011767','0.00289161 / 0.00171970','0.000222314 / 0.000086832')
t_value <- c('63.58006 /    -15.38636', '124.90233 / 111.11277','   -64.60783 / -6.62033','118.68508 / 5.76625')
p_value <- c('0 / 0', '0 / 0', '0 / 0', '0 / 0')
# Join the variables to create a data frame
df2 <- data.frame(fixed.effect,random.effect,family,fixed.vars,coefficient.estimate,std.error,t_value,p_value)

#Utilize datatable and formattable to highlight a row
datatable(df2) %>% formatStyle(
  'family',
  target = 'row',
  backgroundColor = styleEqual(c("Gamma(link=identity)"), c('lime'))
)
```

Show 10 entries Search:

	fixed.effect	random.effect	family	fixed.vars	coefficient.estimate	std.error	t_value	p_value
1	TotalTime ~ sessionIndex + rep	1 | subject	Gamma(link=identity)	sessionIndex / rep	-0.186639 / -0.007104	0.00293549 / 0.00046170	63.58006 / -15.38636	0 / 0
2	TotalTime ~ sessionIndex + rep	1 | subject	Gamma(link=inverse)	sessionIndex / rep	0.02501722 / 0.00130749	0.000200294 / 0.000011767	124.90233 / 111.11277	0 / 0
3	TotalTime ~ sessionIndex + rep	1+rep|subject	Gamma(link=identity)	sessionIndex / rep	-0.186821 / -0.011385	0.00289161 / 0.00171970	-64.60783 / -6.62033	0 / 0
4	TotalTime ~ sessionIndex + rep	1+rep|subject	Gamma(link=inverse)	sessionIndex / rep	0.02638535 / 0.00050069	0.000222314 / 0.000086832	118.68508 / 5.76625	0 / 0

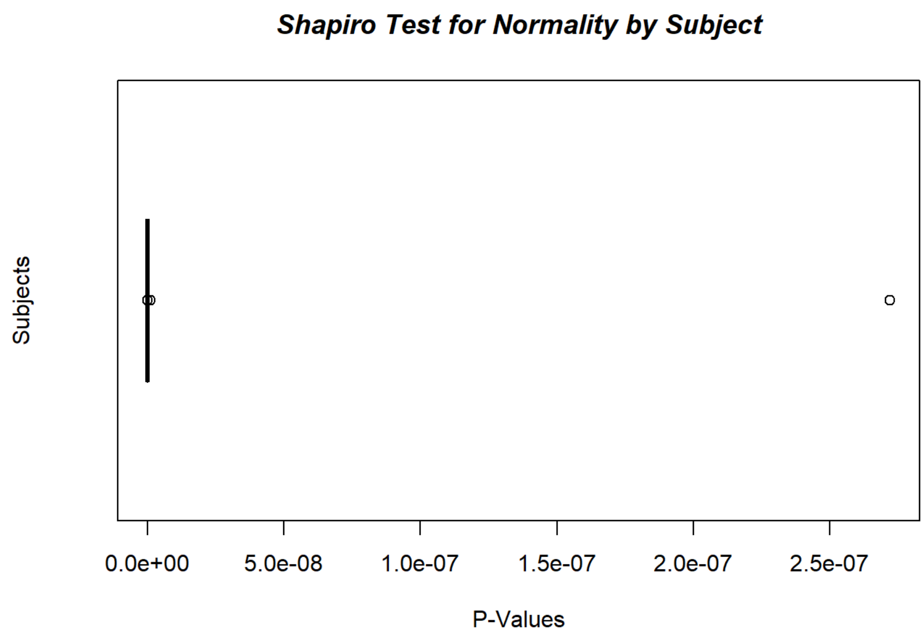
Showing 1 to 4 of 4 entries Previous 1 Next

The results of these two models show that repetitions and consecutive sessions do influence password typing speed. In fact, the p value obtained shows that the null hypothesis has not been disproven. Results show that total time taken to type the password decreases as repetitions increase and to a greater extent as session increases. In other words, people tend to type a password faster the more they type it in a given day and when repeatedly typing the password on consecutive days.

Post Hoc Analysis

For the post hoc analysis, I decided to further confirm whether the response variable is not normally distributed and compare the fixed effects to determine if there's an interaction between them. I tested this via a Shapiro-Wilkins test for normality grouped by subject as this test has a limit of 3500 for observations, and by manipulating the closest fitting model to have a fixed variable of session:repetition (session:repetition).

```
# Run a Shapiro-Wilk test for normality on the dataset grouped by subject
ShapiroTestResults <- PasswordDataEA1.df %>%
  group_by(subject) %>%
  do(tidy(shapiro.test(.$TotalTime)))


#View the results in a boxplot to get an idea of where the data is centered at
boxplot(ShapiroTestResults$p.value, density = 20,
        legend.text = rownames(ShapiroTestResults$subject), horizontal = TRUE, xlab = "P-Values", ylab = "Subjects")
title(main = list("Shapiro Test for Normality by Subject", font = 4))
```




```
#Check for an interaction term between sessionIndex and rep
posthocPQL1 <- glmmPQL(TotalTime ~ sessionIndex*rep + sessionIndex +rep, ~1 | subject, family=Gamma(link=identity), data = P
asswordDataEA1.df, verbose = FALSE)

#View Results
summary(posthocPQL1)
```

```
## Linear mixed-effects model fit by maximum likelihood
##   Data: PasswordDataEA1.df
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | subject
##         (Intercept)  Residual
## StdDev:      1.6813 0.2132885
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects:  TotalTime ~ sessionIndex * rep + sessionIndex + rep
##                      Value  Std.Error    DF   t-value p-value
## (Intercept)       6.487096 0.23783764 20346  27.27531       0
## sessionIndex     -0.268579 0.00602308 20346 -44.59160       0
## rep              -0.022724 0.00109991 20346 -20.66014       0
## sessionIndex:rep  0.003146 0.00020085 20346  15.66136       0
##  Correlation:
##                  (Intr) sssnIn rep
## sessionIndex     -0.128
## rep              -0.124  0.806
## sessionIndex:rep  0.112 -0.876 -0.910
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.0340800 -0.6005026 -0.2080043  0.3603056 26.5097010
##
## Number of Observations: 20400
## Number of Groups: 51
```

```
#Set up a formatted table for the results
fixed.effect <- c('TotalTime ~ sessionIndex*rep + sessionIndex + rep','TotalTime ~ sessionIndex*rep + sessionIndex + rep','T
otalTime ~ sessionIndex*rep + sessionIndex + rep')
random.effect <- c('1 | subject','1 | subject','1 | subject')
family <- c('Gamma(link=identity)','Gamma(link=identity)','Gamma(link=identity)')
fixed.vars <- c('sessionIndex','rep','sessionIndex:rep')
value <- c('-0.268579', '-0.022724', '0.003146')
std.error <- c('0.00602308', '0.00109991', '0.00020085')
t_value <- c('-44.59160','-20.66014','15.66136')
p_value <- c('0','0','0')
# Join the variables to create a data frame
df2 <- data.frame(fixed.effect,random.effect,family,fixed.vars,value,std.error,t_value,p_value)
#Highlight relevant row
datatable(df2) %>% formatStyle(
  'fixed.vars',
  target = 'row',
  backgroundColor = styleEqual(c("sessionIndex:rep"), c('lime'))
)
```

Show 10 **entries** **Search:**

	fixed.effect	random.effect	family	fixed.vars	value	std.error	t_value	p_value
1	TotalTime ~ sessionIndex*rep + sessionIndex + rep	1 | subject	Gamma(link=identity)	sessionIndex	-0.268579	0.00602308	-44.59160	0
2	TotalTime ~ sessionIndex*rep + sessionIndex + rep	1 | subject	Gamma(link=identity)	rep	-0.022724	0.00109991	-20.66014	0
3	TotalTime ~ sessionIndex*rep + sessionIndex + rep	1 | subject	Gamma(link=identity)	sessionIndex:rep	0.003146	0.00020085	15.66136	0

Showing 1 to 3 of 3 entries Previous 1 Next

The results of my post-hoc analysis were that the data is definitely not of a normal distribution and that there is an interaction between session and repetition although it has a very small effect on the response variable compared to that of session and repetition individually.

Conclusions

The random effects model I created that fits the data the best indicates that the time taken to type a password, changes over time as one retypes it throughout the day and on consecutive days. Thus, I've failed to reject the null hypothesis and I have confirmed that at least within the boundaries of this sample and to a statistically significant extent, a person's typing dynamics change over time, short and long term. Also, the results of the post-hoc analysis showed that the data is extremely likely to not be normally distributed and that there is an interaction between session and repetition.