NOTE: We decided to include only 2 graphs in each section (histogram, boxplot, bar graph, ridge density plot, scatter plot) to reduce the length of the report. While doing analysis we looked at all the plots and code for all plots is included in rmd file.

1.Dataset Introduction

The dataset records typing speed of a group of 51 different individuals who have access to the passcode ".tie5Roanl" where each row corresponds to the record of time an individual takes to type the passcode to get access to a system. While an individual has access to the system, no other individual has access to the system. Below is the description of each column in the dataset: The column "X" is row-id ranging from 351 to 20400. The column "subject" contains list of 51 different individuals where each user is assigned with an unique identifier encoded as s002, s003 and so on (note that they are not in sequence from s001 to s051). The column "sessionIndex" is a continuous block of time where an individual has had access to the system.Only one user will be associated with a given session. In our dataset, sessionIndex takes value either 7 or 8 that means we are provided with the information of users participating in session 7 and session 8. The column "rep" is the repetition of the individual passcode entries within a session, which in our dataset ranges from 1 to 50. There are 31 columns that represent the timing information for the passcode. The name of the column encodes the type of timing information. Column names of the form "H.keyname" represents a hold time for the named key, meaning the time from when key was pressed to when it was released. Column names of the form "DD.key1_name.key2_name" represents a keydown-keydown time for the named keys meaning the time from when key1 was pressed to when key2 was pressed. Column names "UD.key1_name.key2_name" represents a keyup-keydown time for the named keys meaning the time from when key1 was released to when key2 was pressed. We observed in our dataset that all column of the type UD.key1_name.key2_name, except some, have negative values. And, the times in the column type "DD.key1_name.key2_name" is the addition of the times in the two columns "H.keyname" and "UD.key1_name.key2_name". For example Consider row 1 observation for user s002: DD.period.t = H.period + UD.period.t 0.3573 = 0.1391 + 0.2182 And this is true for all other observations.

In passcode data analysis we are working with three datasets namely:

1.known.csv: This csv file contains 1776 observations over 35 columns where the individual who entered the passcode is known.

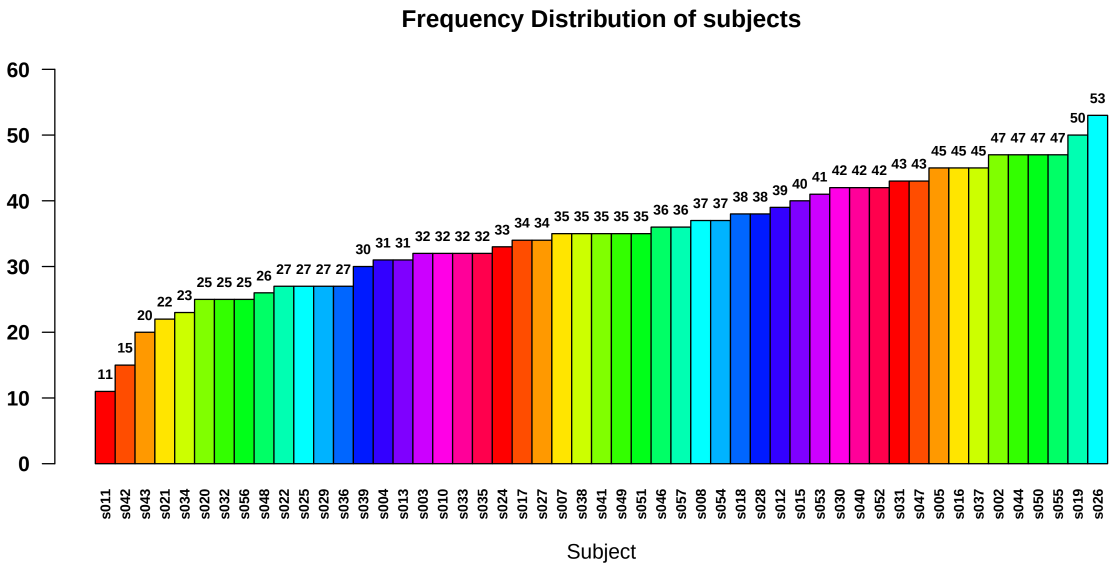
2.Unknown.csv: This csv file contains 1795 observations over 33 columns where the individual who entered the passcode is unknown.

3.Questioned.csv: This csv file contains 12 sessions where we need to predict the individual who had access to the system.

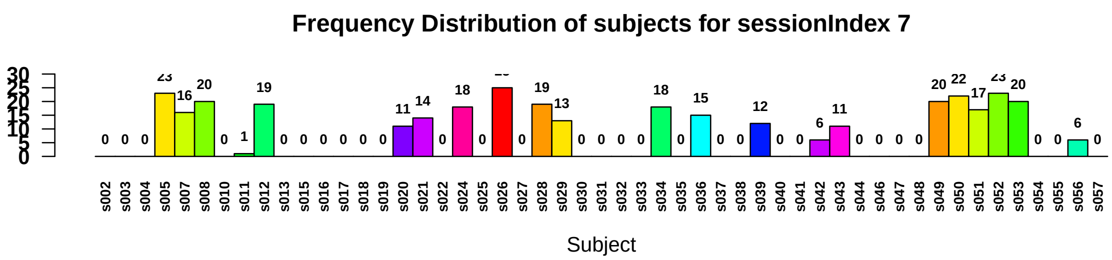
2.Data Exploration

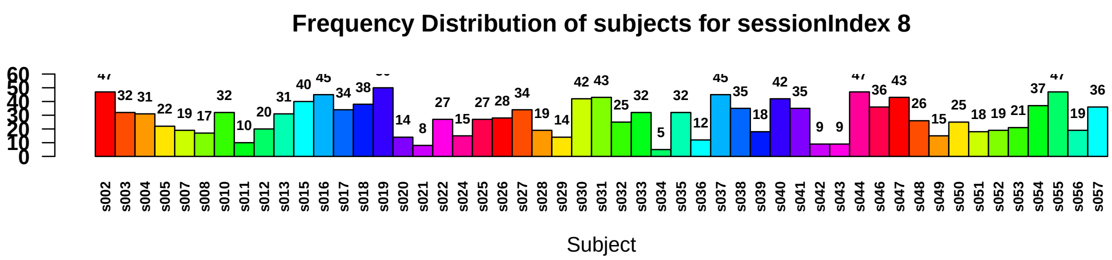
Checking for NA values We checked for missing values and found out that there are no missing values in the dataset.

As mentioned before, there are 51 different individuals recorded in "subject" column who have access to the system, meaning "subject" is a factor variable with 51 levels.The below plot shows the number of observations that belong to each subject and we can notice that the observations are not evenly distributed. The user "s011" logged into system total 9 times which is the least number in our dataset. The user "s026" logged into to system total 43 times which is the maximum in our dataset. There are other users who have same number of occurrences in our dataset,for eg. users s020, s032, s056 logged into system for total 20 times, user s022, s025, s029, s036 logged into system for total 22 times and so on.



Let us look at the subjects and their occurrences in our dataset grouped by session index. As explained before, session index is the time slot when an individual had access to the system. In our dataset we have two session index values: 7 and 8. There are more occurrences for session index 8 compared to session index 7, to be specific 349 observations for session index 7 and 1427 observations for session index 8. One thing to be noted that session index 8 has occurences of all subject values where as session index 7 does not.The following plots gives a clear idea:





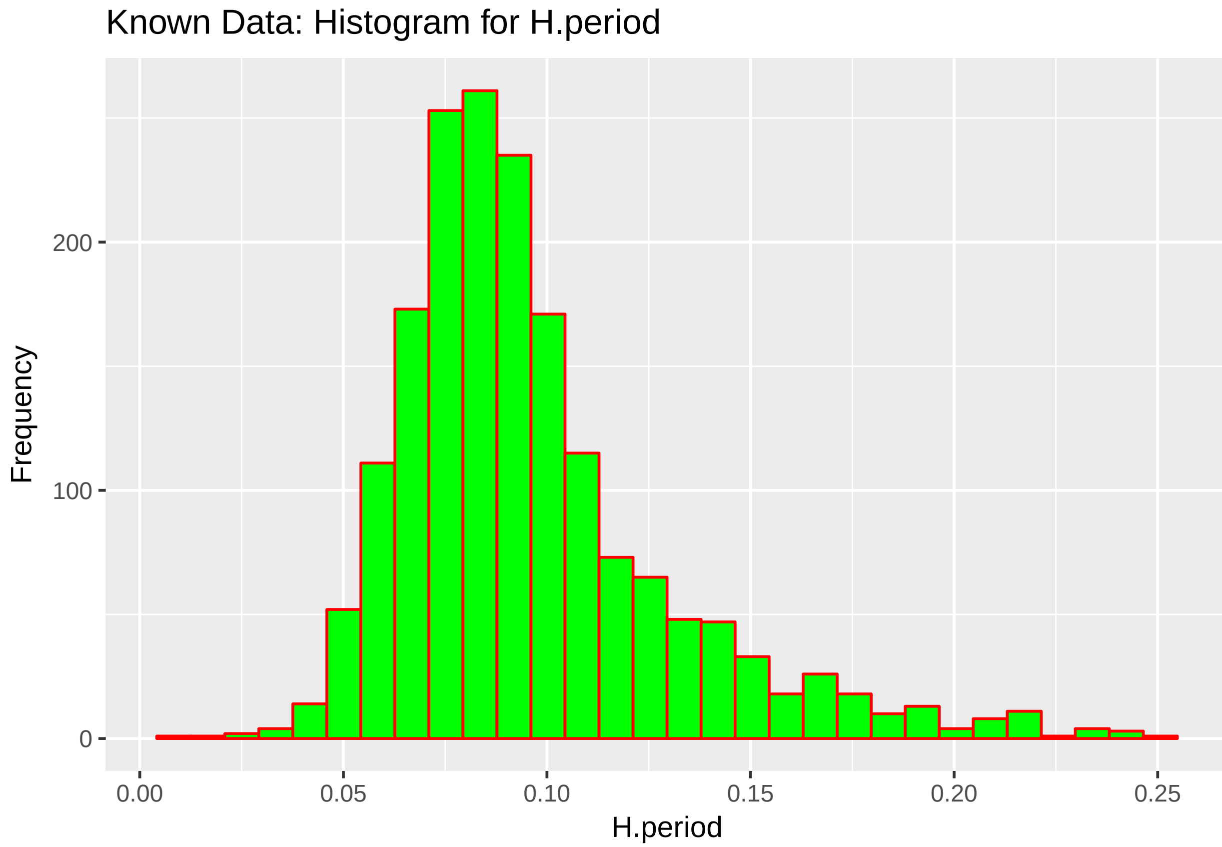
Looking at independent variables

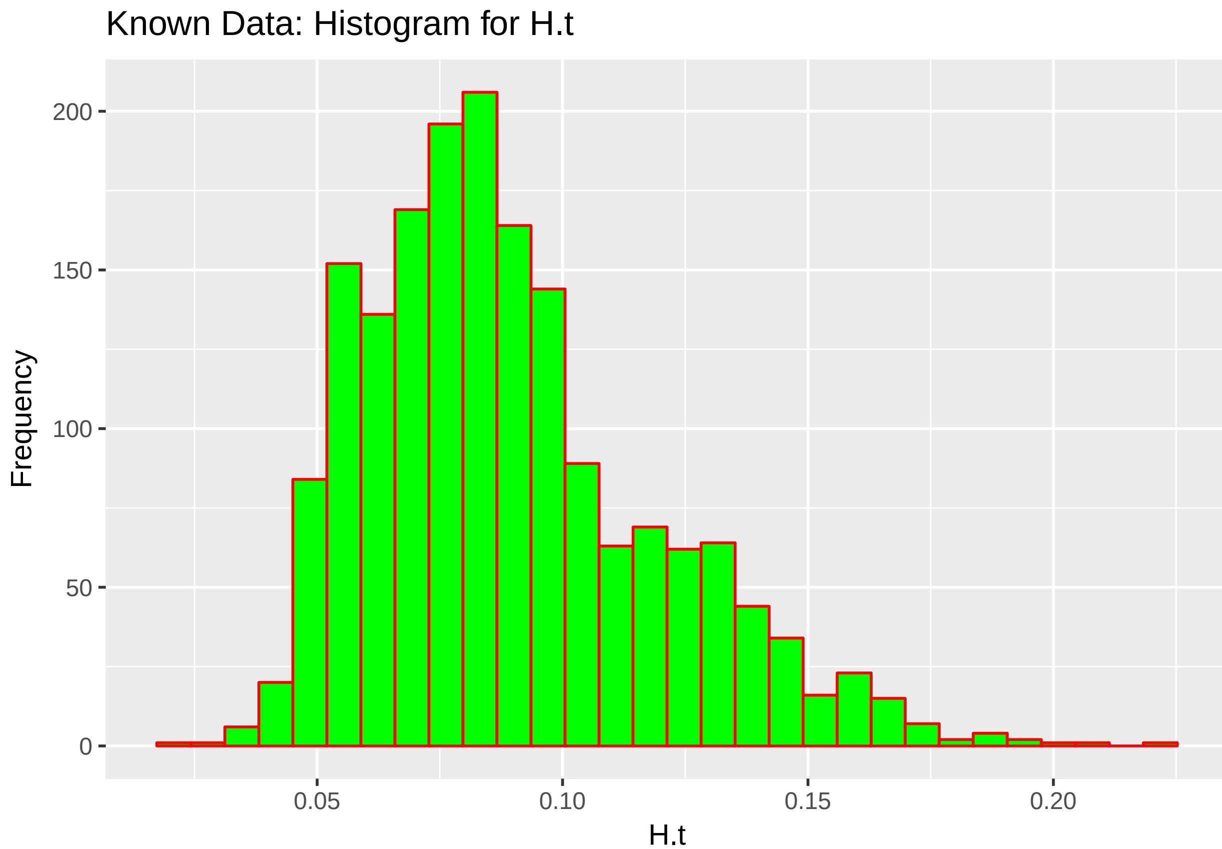
X

x variable is a row id and hence we removed it from our datset.

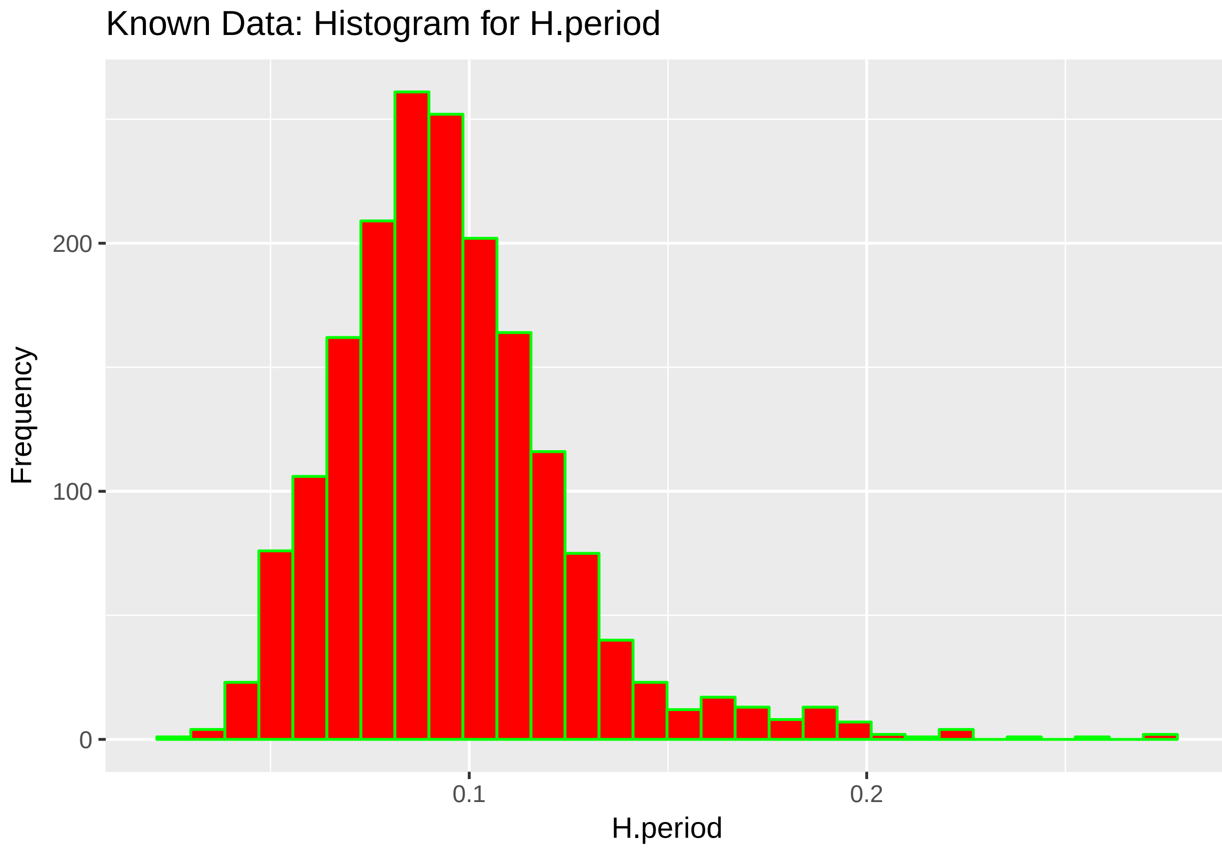
Histogram for H variables from known and unknown data:

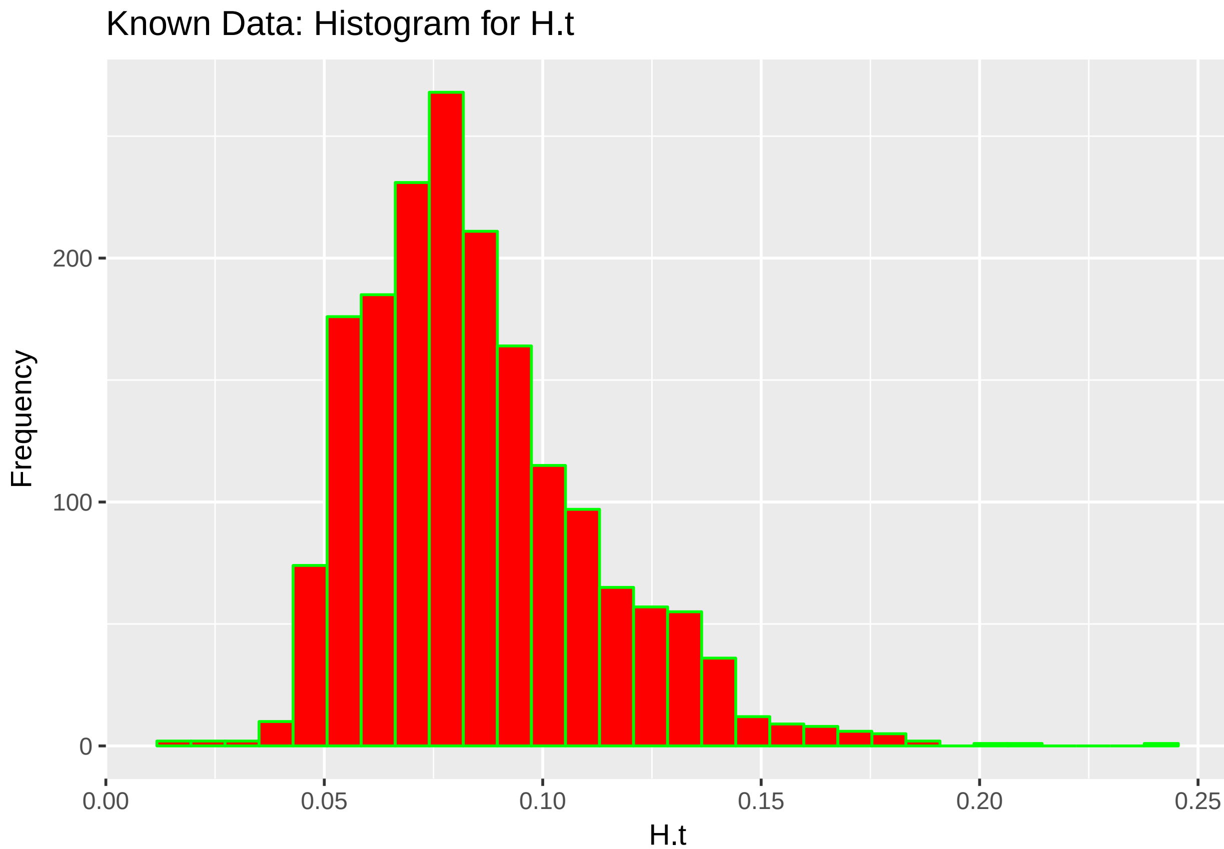
The histogram of the H variables of passcode from known and unknown file has a bell shaped curve meaning their distribution are normal. Few of them are slightly skewed, the reason being most of the values are close to zero and in real world the data follows close to normal distribution.






```
##  [1] "session.id"      "rep"             "H.period"
##  [4] "DD.period.t"     "UD.period.t"     "H.t"
##  [7] "DD.t.i"          "UD.t.i"          "H.i"
## [10] "DD.i.e"          "UD.i.e"          "H.e"
## [13] "DD.e.five"       "UD.e.five"       "H.five"
## [16] "DD.five.Shift.r" "UD.five.Shift.r" "H.Shift.r"
## [19] "DD.Shift.r.o"    "UD.Shift.r.o"    "H.o"
## [22] "DD.o.a"          "UD.o.a"          "H.a"
## [25] "DD.a.n"          "UD.a.n"          "H.n"
## [28] "DD.n.l"          "UD.n.l"          "H.l"
## [31] "DD.l.Return"     "UD.l.Return"     "H.Return"
```


Known Data: Histogram for H.period
Frequency
200
100
0
0.1
0.2
H.period

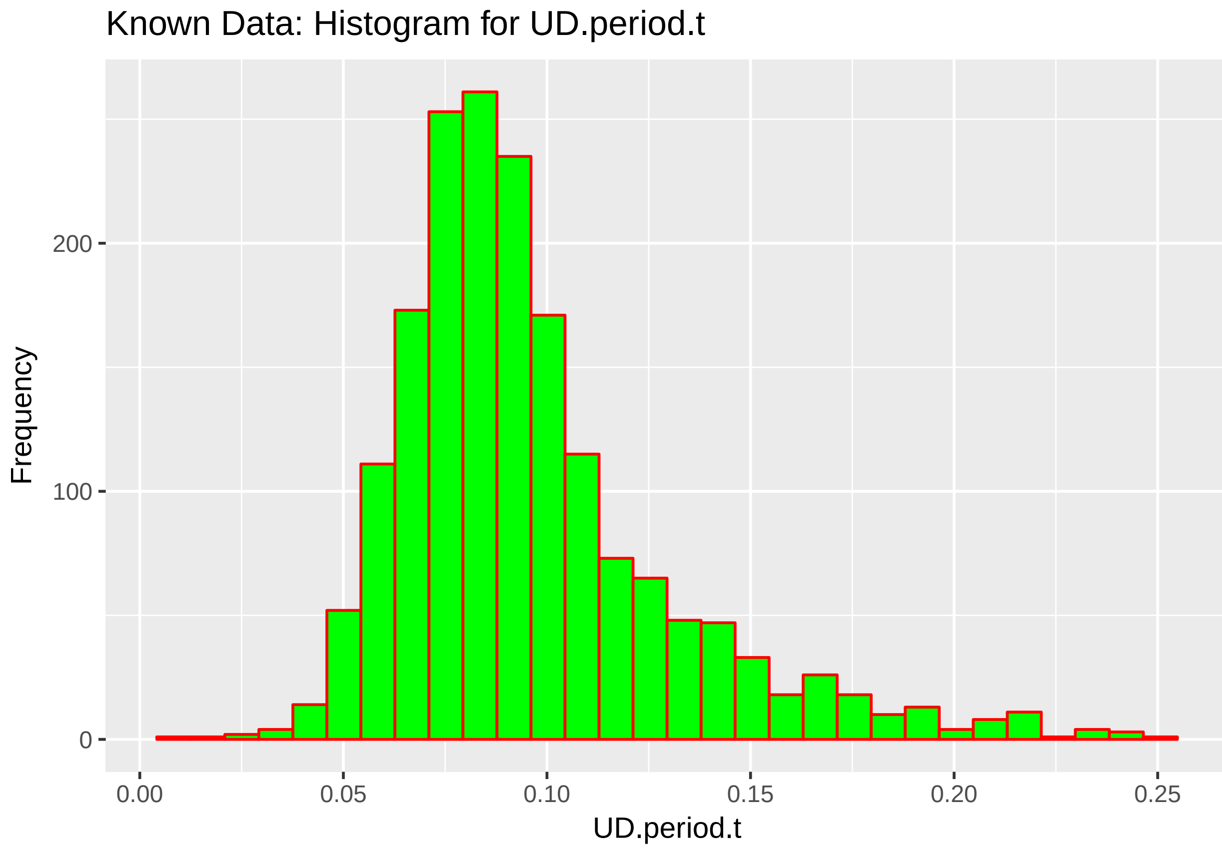


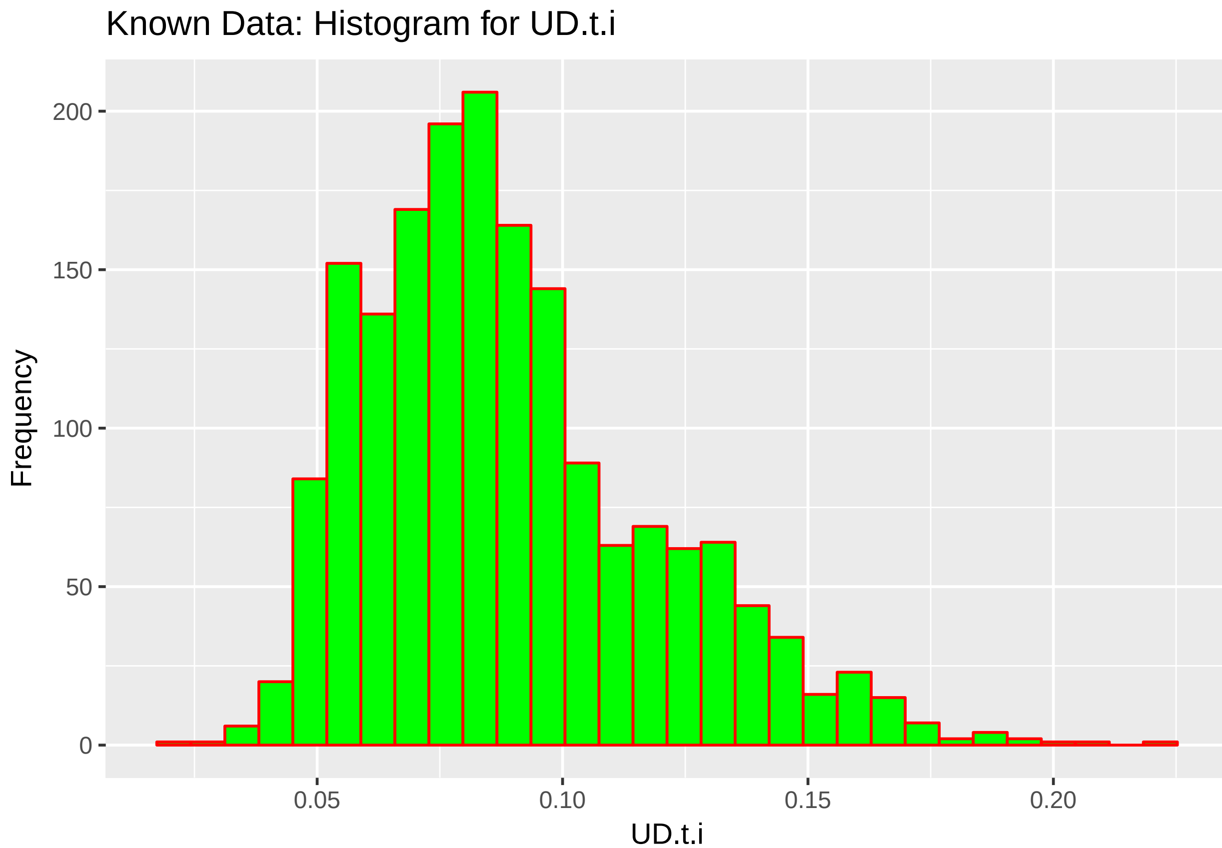
Histogram for UD variables from known and unknown data:

The distribtuion of UD variables from known and unknown file are rightly-skewed. The up-down time of most of the users while typing passcode are takes smaller values.

```
##  [1] "X"               "subject"         "sessionIndex"
##  [4] "rep"             "H.period"        "DD.period.t"
##  [7] "UD.period.t"     "H.t"             "DD.t.i"
## [10] "UD.t.i"          "H.i"             "DD.i.e"
## [13] "UD.i.e"          "H.e"             "DD.e.five"
## [16] "UD.e.five"       "H.five"          "DD.five.Shift.r"
## [19] "UD.five.Shift.r" "H.Shift.r"       "DD.Shift.r.o"
## [22] "UD.Shift.r.o"    "H.o"             "DD.o.a"
## [25] "UD.o.a"          "H.a"             "DD.a.n"
## [28] "UD.a.n"          "H.n"             "DD.n.l"
## [31] "UD.n.l"          "H.l"             "DD.l.Return"
## [34] "UD.l.Return"     "H.Return"
```

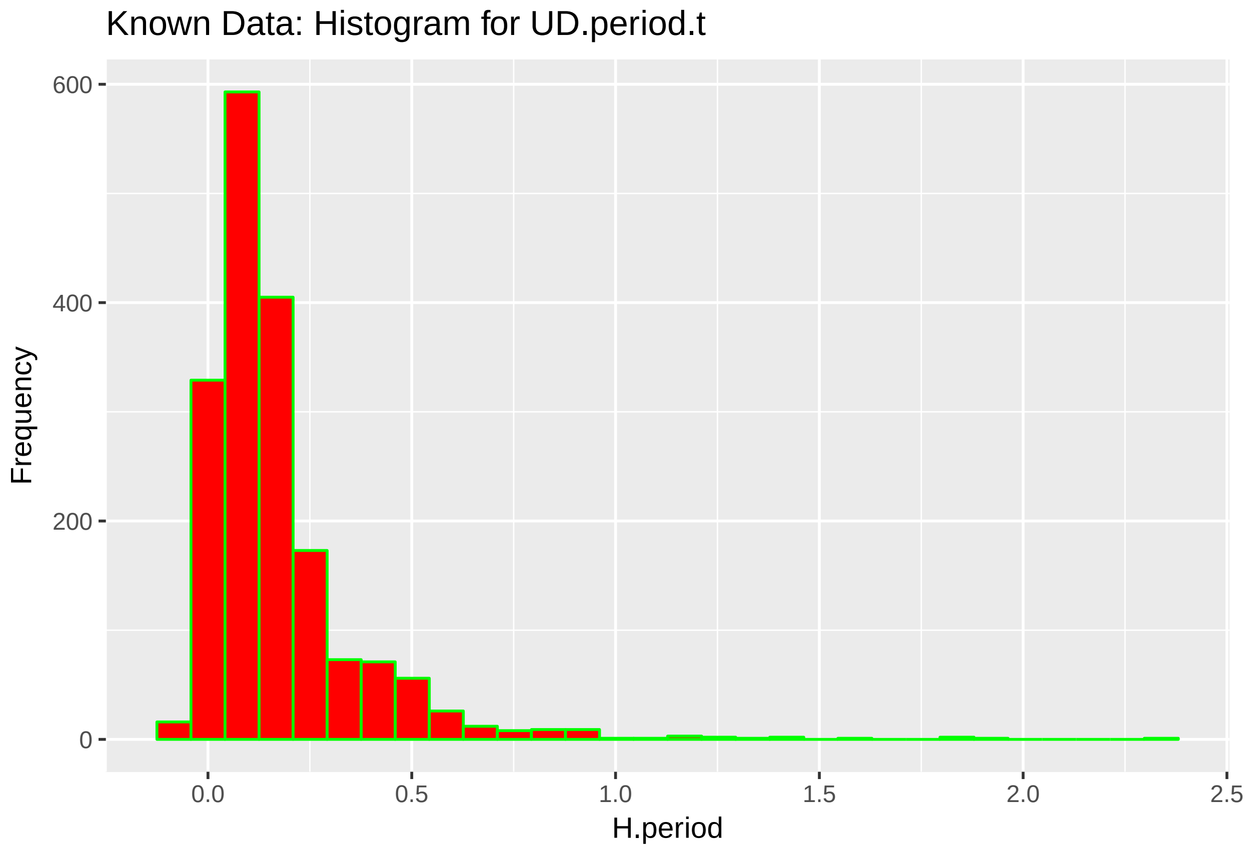
```
##  [1] "UD.period.t"     "UD.t.i"          "UD.i.e"
##  [4] "UD.e.five"       "UD.five.Shift.r" "UD.Shift.r.o"
##  [7] "UD.o.a"          "UD.a.n"          "UD.n.l"
## [10] "UD.l.Return"
```

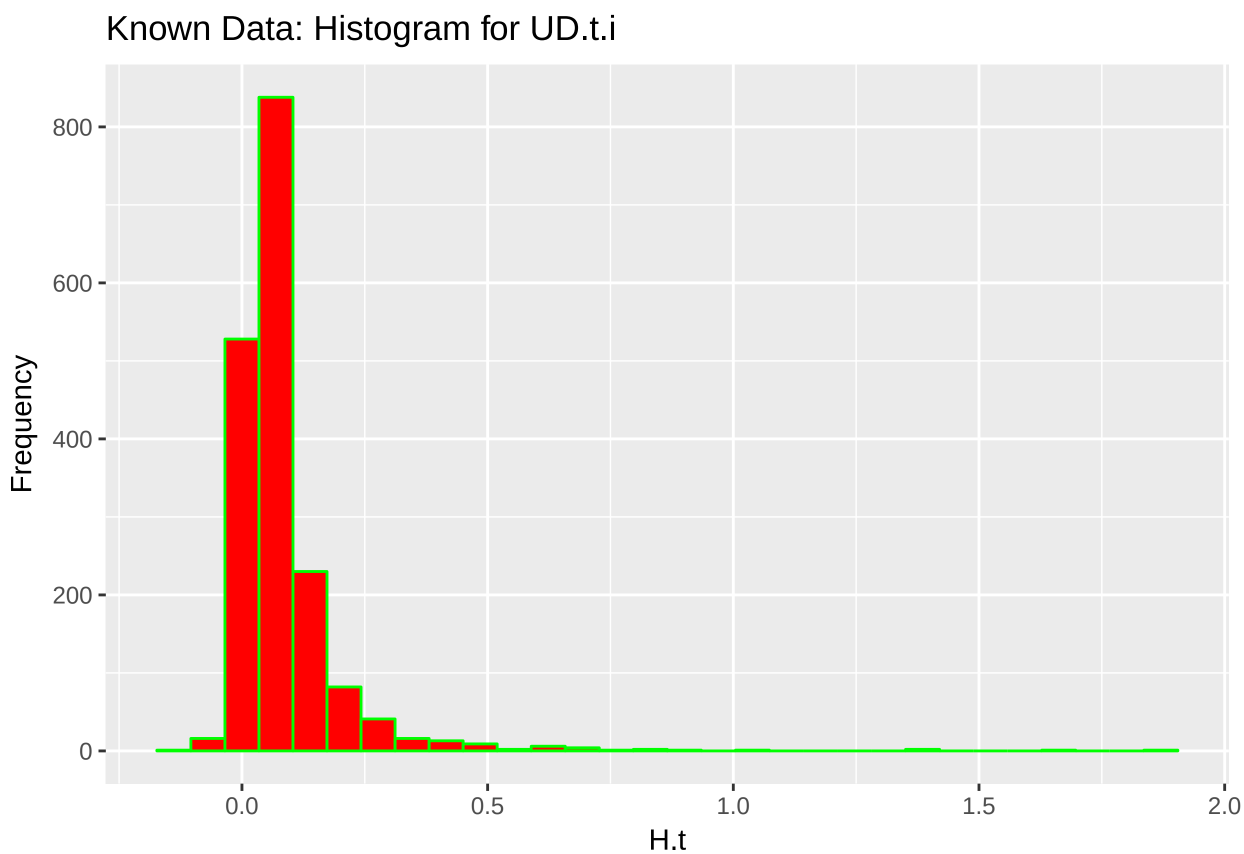

Known Data: Histogram for UD.period.t
Frequency
200
100
0
0.00
0.05
0.10
0.15
0.20
0.25
UD.period.t




```
##  [1] "session.id"      "rep"             "H.period"
##  [4] "DD.period.t"     "UD.period.t"     "H.t"
##  [7] "DD.t.i"          "UD.t.i"          "H.i"
## [10] "DD.i.e"          "UD.i.e"          "H.e"
## [13] "DD.e.five"       "UD.e.five"       "H.five"
## [16] "DD.five.Shift.r" "UD.five.Shift.r" "H.Shift.r"
## [19] "DD.Shift.r.o"    "UD.Shift.r.o"    "H.o"
## [22] "DD.o.a"          "UD.o.a"          "H.a"
## [25] "DD.a.n"          "UD.a.n"          "H.n"
## [28] "DD.n.l"          "UD.n.l"          "H.l"
## [31] "DD.l.Return"     "UD.l.Return"     "H.Return"
```

```
##  [1] "UD.period.t"     "UD.t.i"          "UD.i.e"
##  [4] "UD.e.five"       "UD.five.Shift.r" "UD.Shift.r.o"
##  [7] "UD.o.a"          "UD.a.n"          "UD.n.l"
## [10] "UD.l.Return"
```


Known Data: Histogram for UD.period.t
600
400
200
0
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
H.period

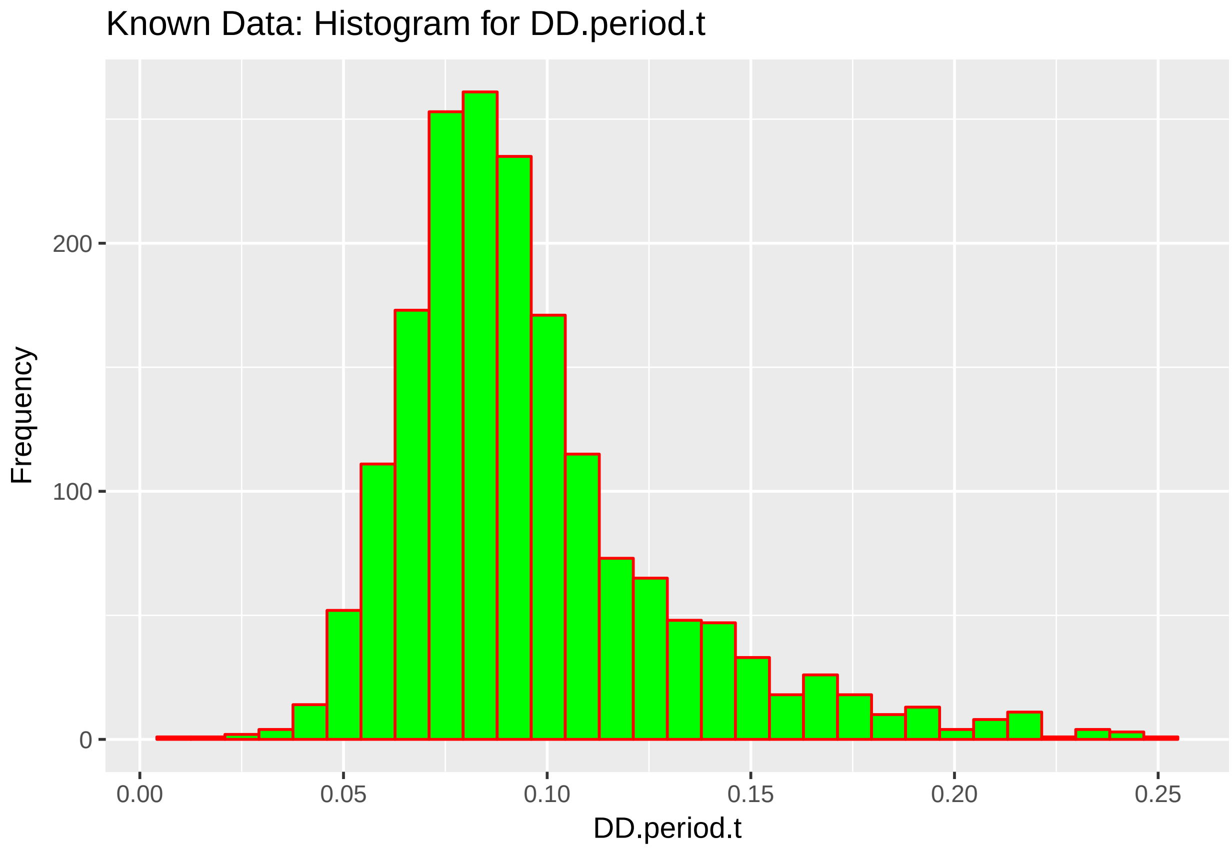


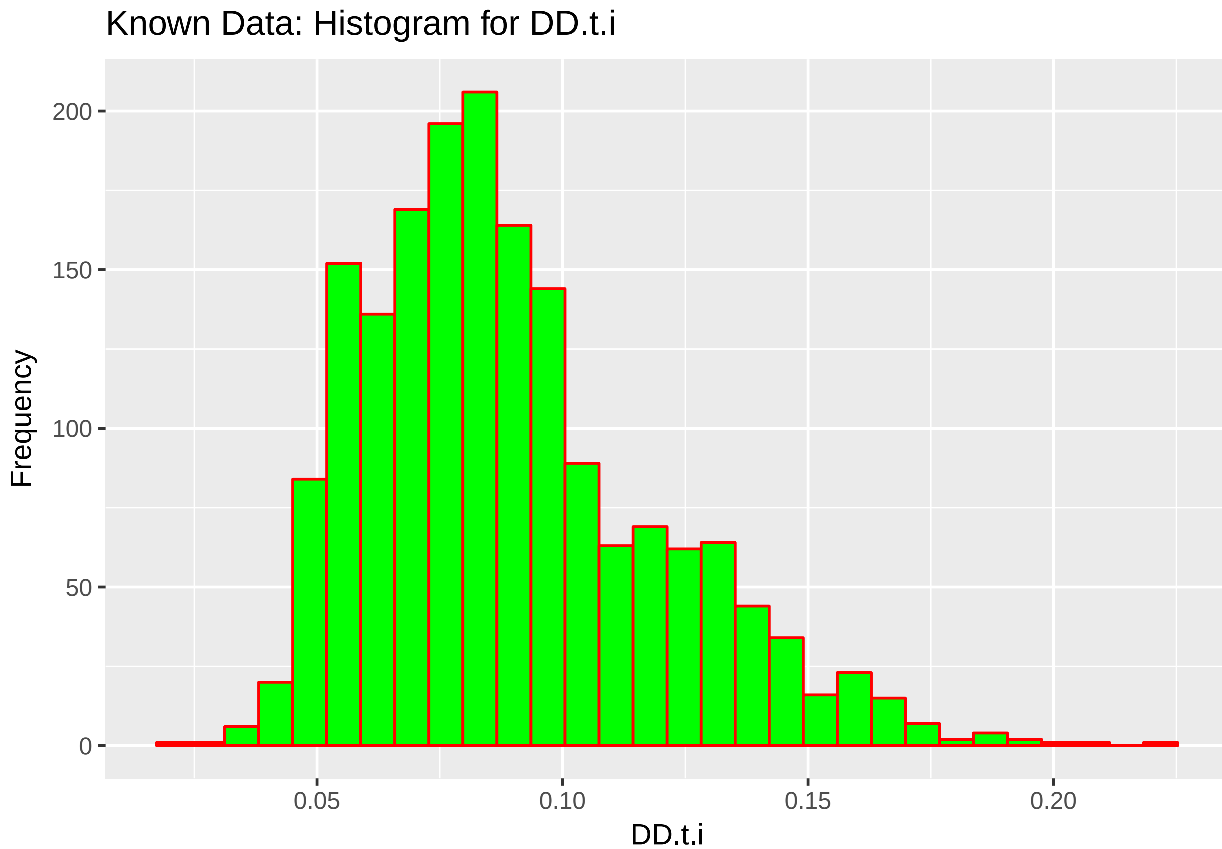
Distribution of DD variables:

Since DD variables are highly skewed, the information they carry are not different than combined information of H variables and UD variables, hence we decided to not include them in our model.

```
##  [1] "X"                "subject"          "sessionIndex"
##  [4] "rep"              "H.period"         "DD.period.t"
##  [7] "UD.period.t"      "H.t"              "DD.t.i"
## [10] "UD.t.i"           "H.i"              "DD.i.e"
## [13] "UD.i.e"           "H.e"              "DD.e.five"
## [16] "UD.e.five"        "H.five"           "DD.five.Shift.r"
## [19] "UD.five.Shift.r" "H.Shift.r"        "DD.Shift.r.o"
## [22] "UD.Shift.r.o"     "H.o"              "DD.o.a"
## [25] "UD.o.a"           "H.a"              "DD.a.n"
## [28] "UD.a.n"           "H.n"              "DD.n.l"
## [31] "UD.n.l"           "H.l"              "DD.l.Return"
## [34] "UD.l.Return"      "H.Return"
```

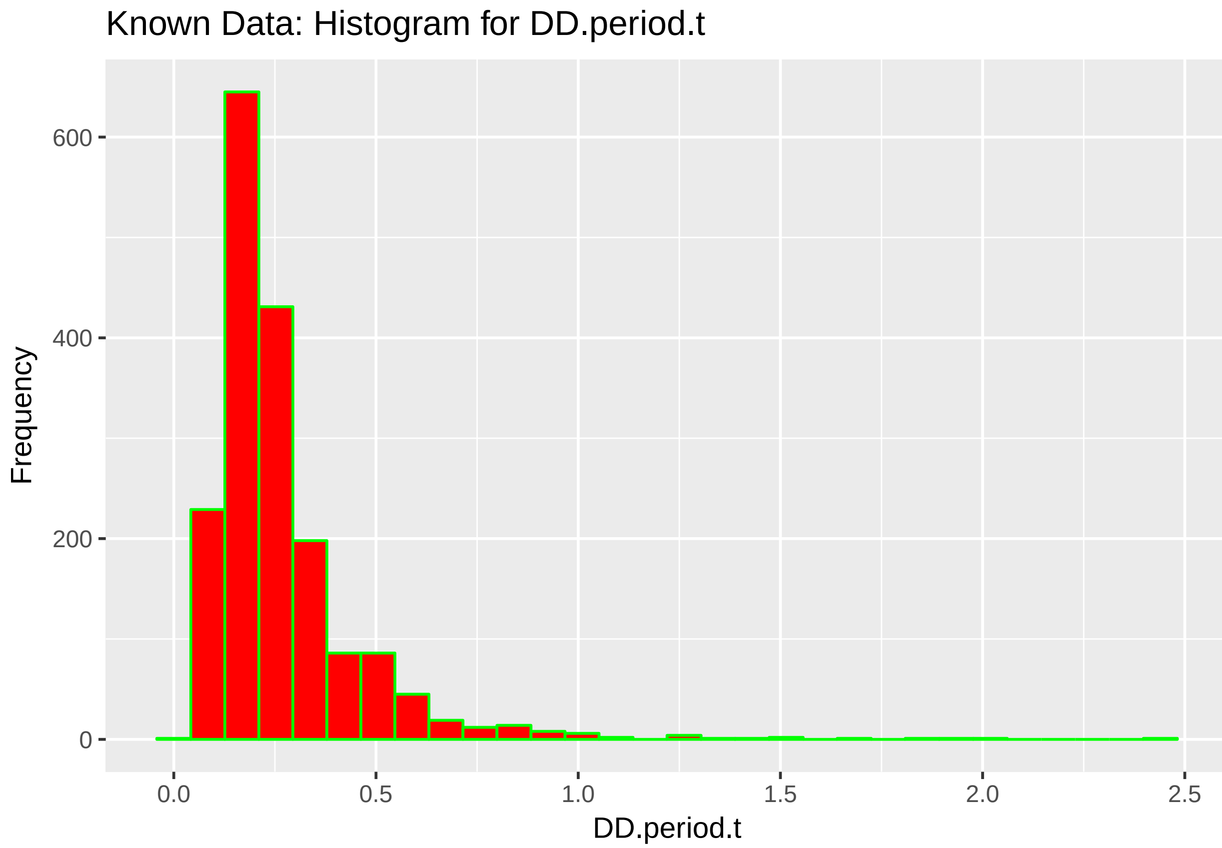
```
##  [1] "UD.period.t"      "UD.t.i"           "UD.i.e"
##  [4] "UD.e.five"        "UD.five.Shift.r" "UD.Shift.r.o"
##  [7] "UD.o.a"           "UD.a.n"           "UD.n.l"
## [10] "UD.l.Return"
```

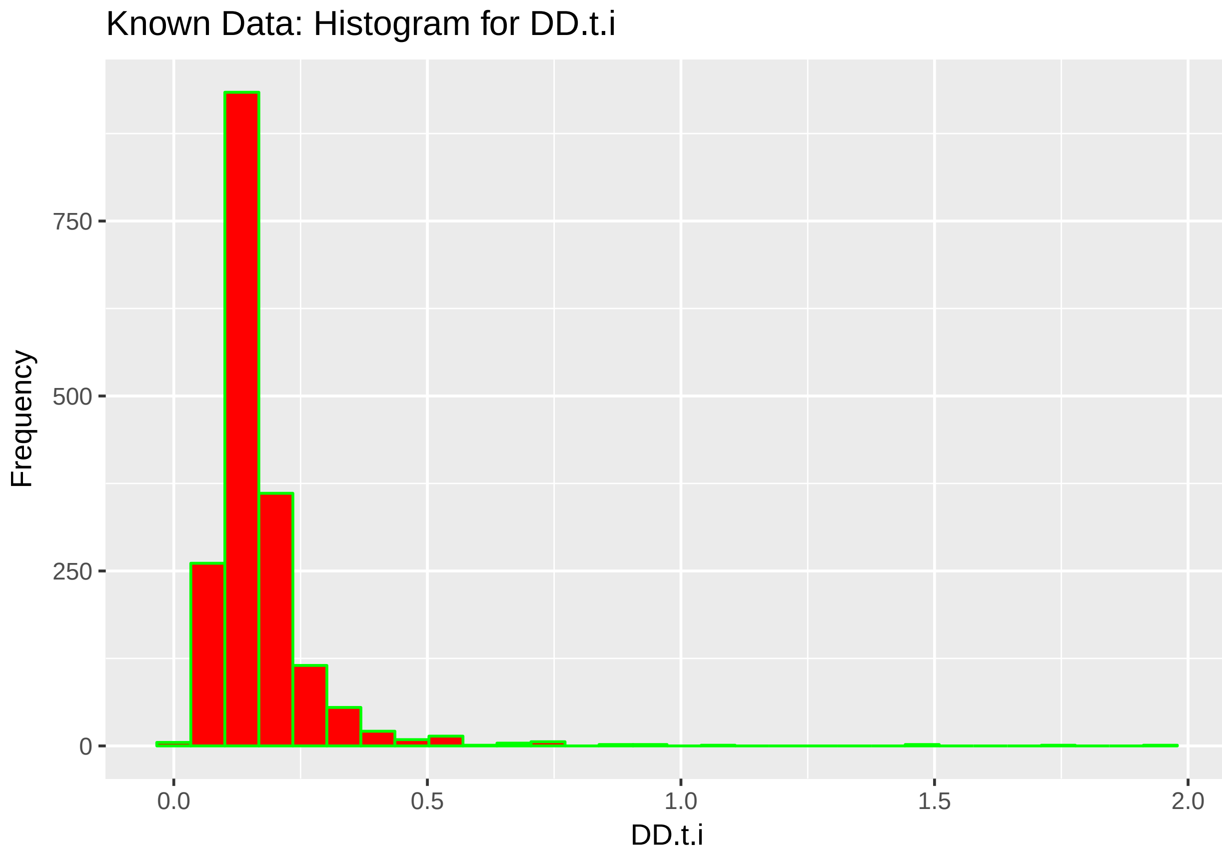

Known Data: Histogram for DD.period.t
Frequency
200
100
0
0.00
0.05
0.10
0.15
0.20
0.25
DD.period.t




```
##  [1] "session.id"      "rep"             "H.period"
##  [4] "DD.period.t"     "UD.period.t"     "H.t"
##  [7] "DD.t.i"          "UD.t.i"          "H.i"
## [10] "DD.i.e"          "UD.i.e"          "H.e"
## [13] "DD.e.five"       "UD.e.five"       "H.five"
## [16] "DD.five.Shift.r" "UD.five.Shift.r" "H.Shift.r"
## [19] "DD.Shift.r.o"    "UD.Shift.r.o"    "H.o"
## [22] "DD.o.a"          "UD.o.a"          "H.a"
## [25] "DD.a.n"          "UD.a.n"          "H.n"
## [28] "DD.n.l"          "UD.n.l"          "H.l"
## [31] "DD.l.Return"     "UD.l.Return"     "H.Return"
```

```
##  [1] "DD.period.t"     "DD.t.i"          "DD.i.e"
##  [4] "DD.e.five"       "DD.five.Shift.r" "DD.Shift.r.o"
##  [7] "DD.o.a"          "DD.a.n"          "DD.n.l"
## [10] "DD.l.Return"
```

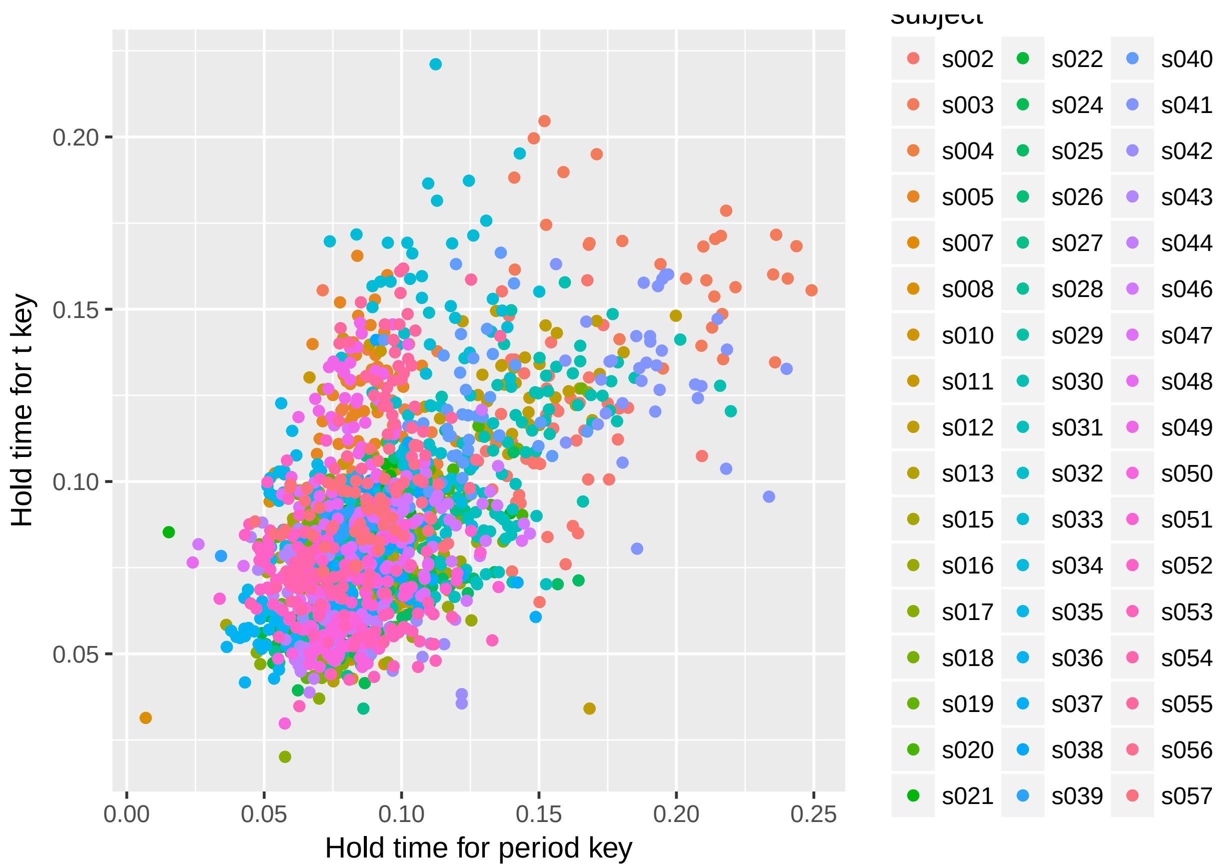

Known Data: Histogram for DD.period.t
Frequency
600
400
200
0
0.0
0.5
1.0
1.5
2.0
2.5
DD.period.t



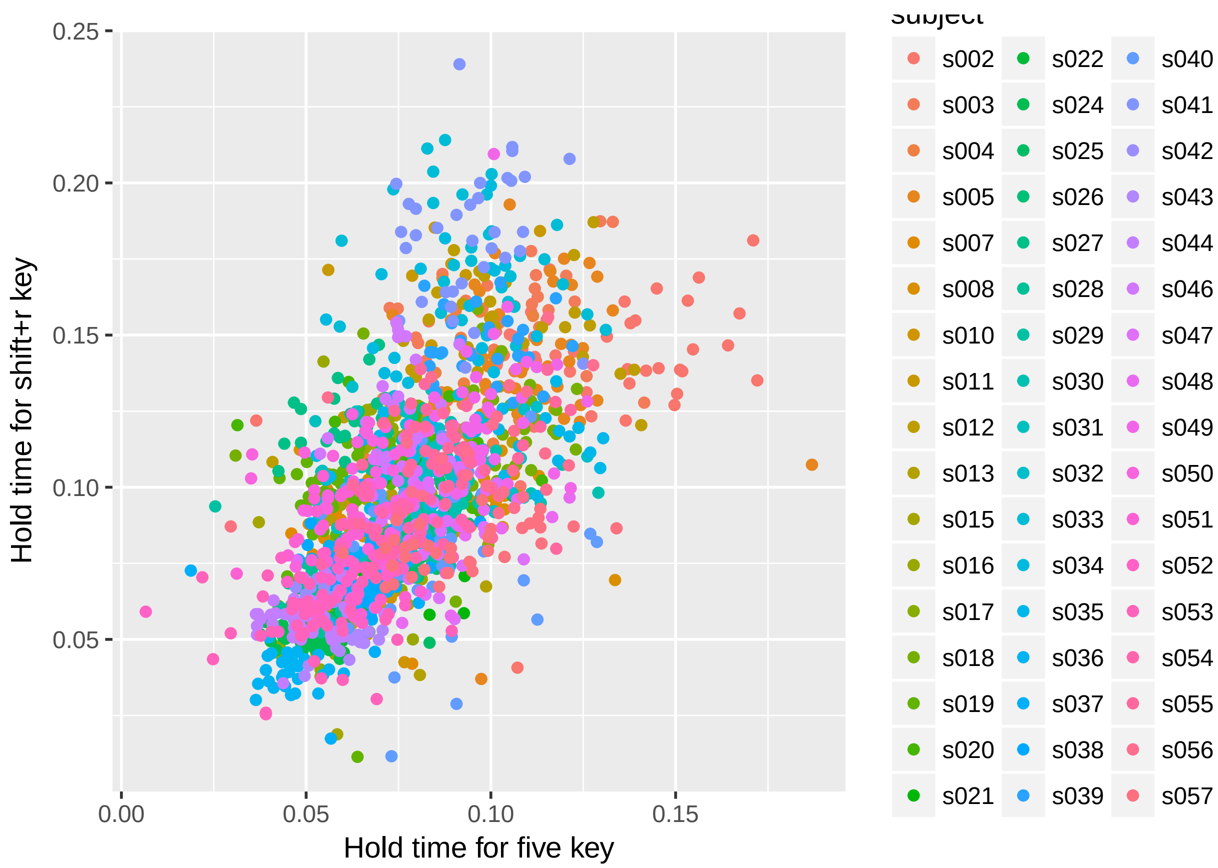
rep

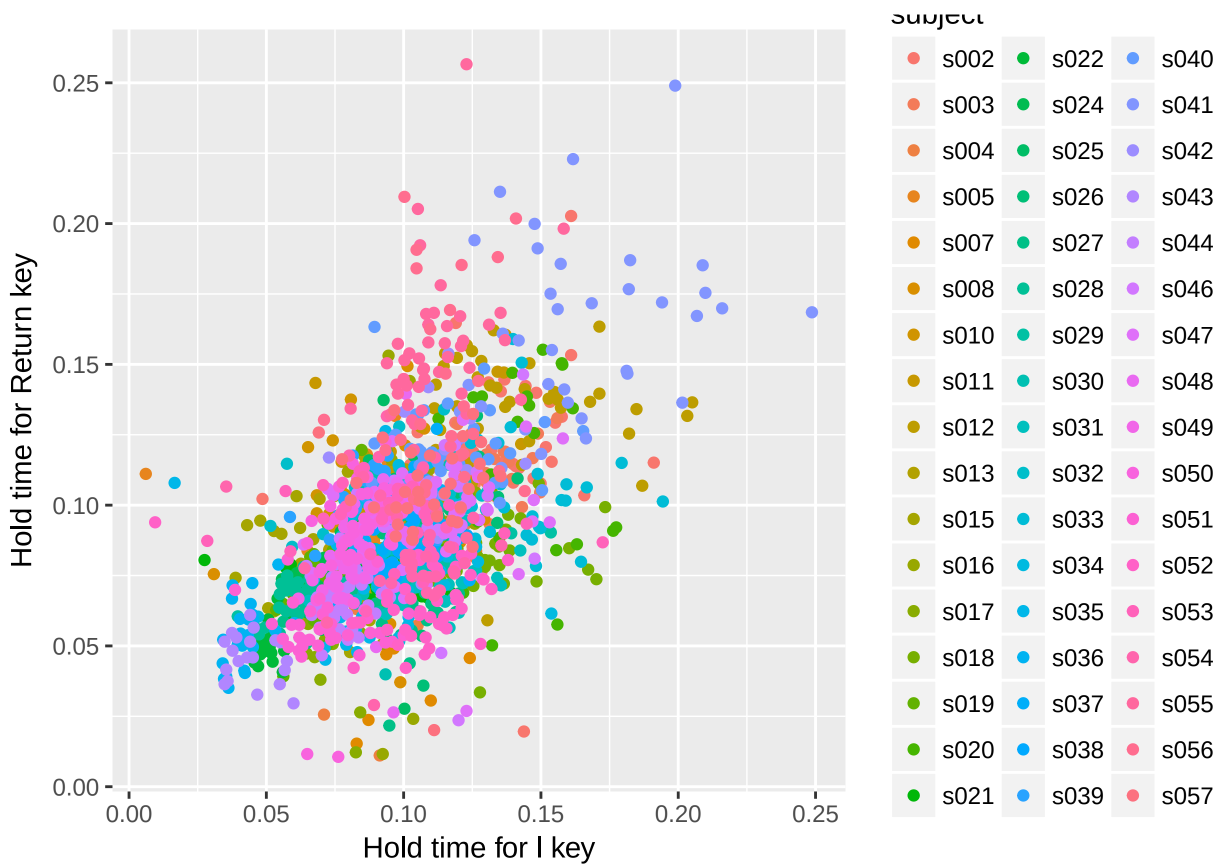
rep is the count of an individual's passcode entries within a session. It takes values from 1 repetition to maximum 50 repetition. There can be different possibilities for a user to enter passcode multiple times in a session: the user types a passcode and leaves the system for some time,the system gets locked if not used for a particular time, so the next time user wants to use the system within same session he needs to enter the passcode again. Other possibility is, user enters wrong passcode and tried multiple times when is a session or the system is designed in a way that it asks user to enter passcode randomly. Since rep variable doesn't carry useful information in predicting user in a session based on typing speed also we are not provided with clear details about rep variable in the document, therefore we decided to not include this variable in our model building process.

Let us look at the three scatter plots for all users : hold time for period key vs hold time of t key, hold time of five key vs hold time of capital r key and hold time of l key vs hold time of return key. We can see a strong positive linear relatisnship among the variables and some observations belonging to different classes overlap. The 51 different users are not easily distuniguishable.

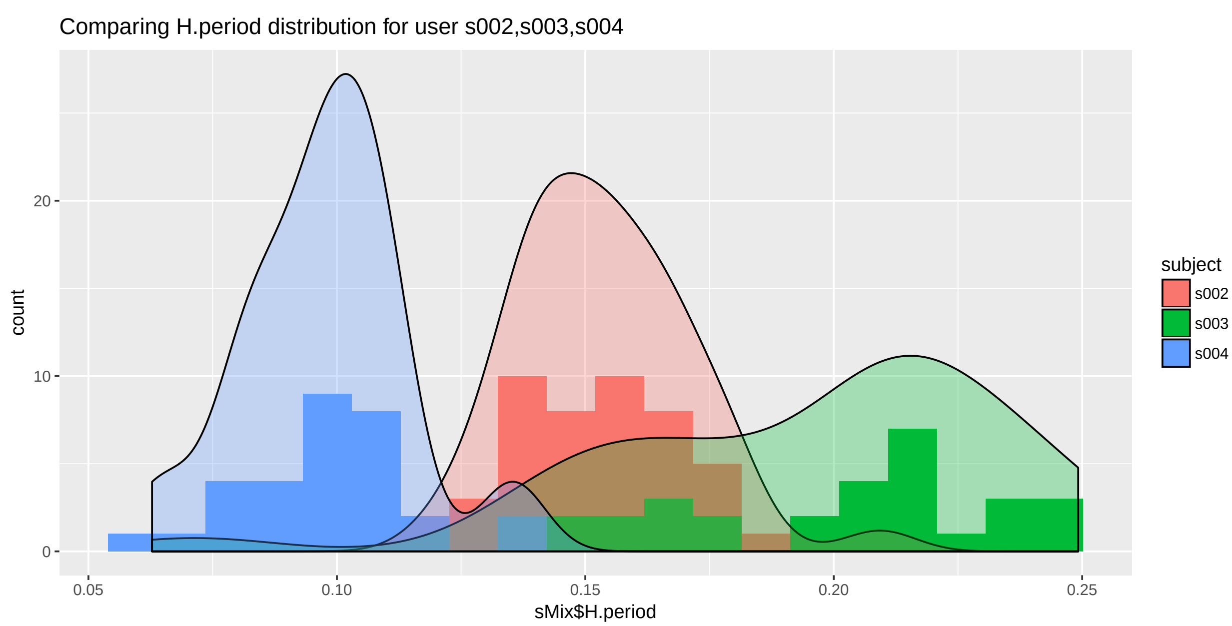


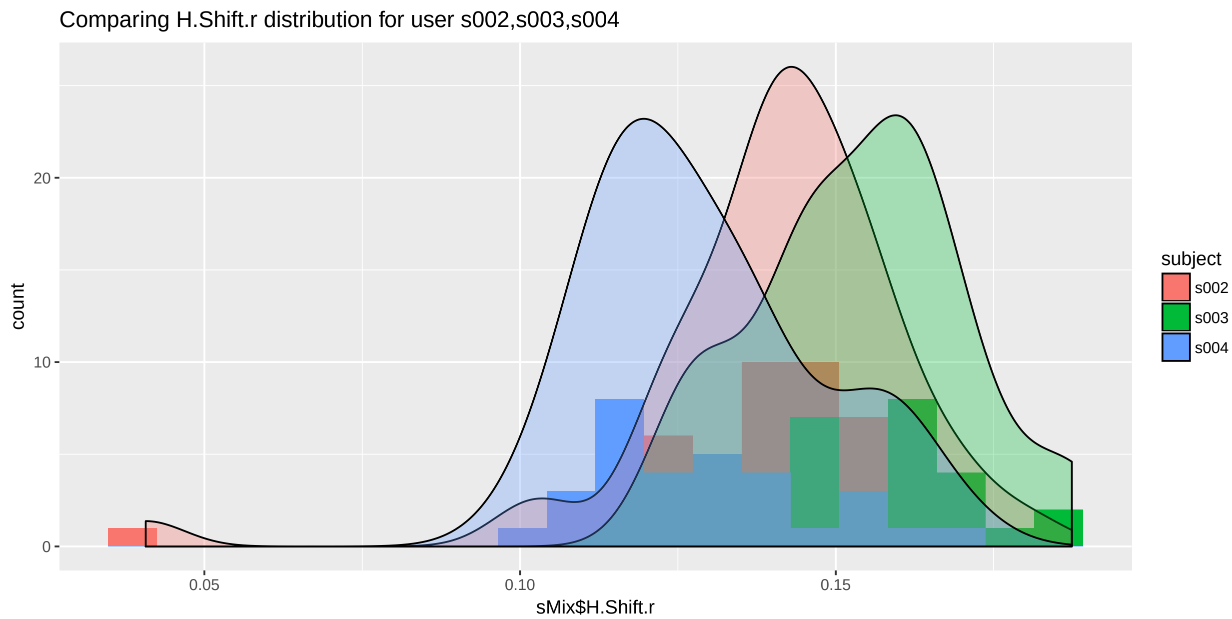
Subject
s002
s003
s004
s005
s007
s008
s010
s011
s012
s013
s015
s016
s017
s018
s019
s020
s021
s022
s024
s025
s026
s027
s028
s029
s030
s031
s032
s033
s034
s035
s036
s037
s038
s039
s040
s041
s042
s043
s044
s046
s047
s048
s049
s050
s051
s052
s053
s054
s055
s056
s057
Hold time for t key
Hold time for period key
0.20
0.15
0.10
0.05
0.00
0.05
0.10
0.15
0.20
0.25


Subject
s002
s003
s004
s005
s007
s008
s010
s011
s012
s013
s015
s016
s017
s018
s019
s020
s021
s022
s024
s025
s026
s027
s028
s029
s030
s031
s032
s033
s034
s035
s036
s037
s038
s039
s040
s041
s042
s043
s044
s046
s047
s048
s049
s050
s051
s052
s053
s054
s055
s056
s057
0.25
0.20
0.15
0.10
0.05
0.00
0.05
0.10
0.15
Hold time for shift+r key
Hold time for five key



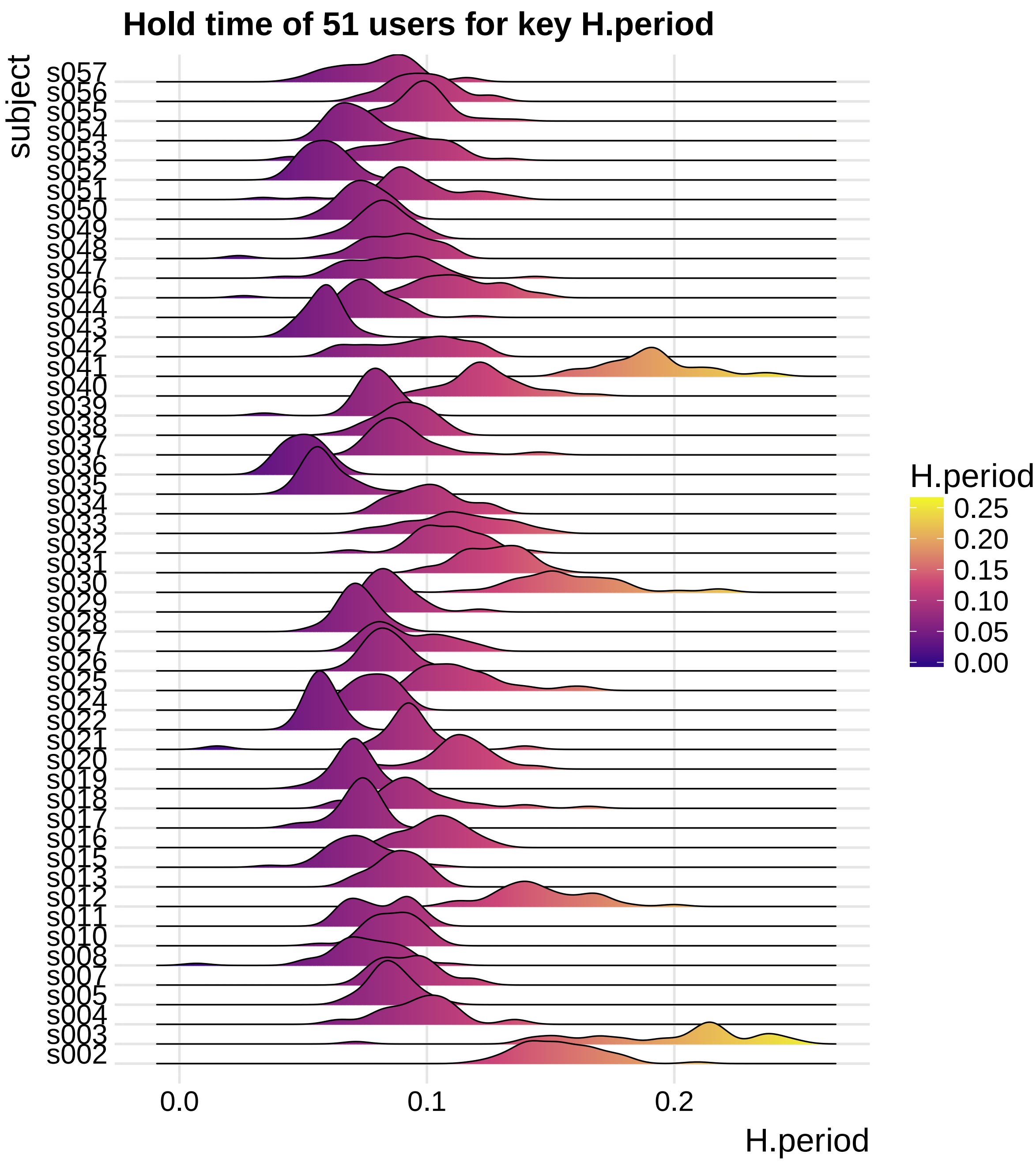
Let us look at the hold time distribution of keys period,t,Shift+r, five, a ,Return for three different users s002, s003, s004. All plots below show gaussian distribution of hold times which is a good indication and we can clearly see some difference in hold time speed of different users. Some graphs show significant difference like hold time for keys period, Shift+r,a and return key.

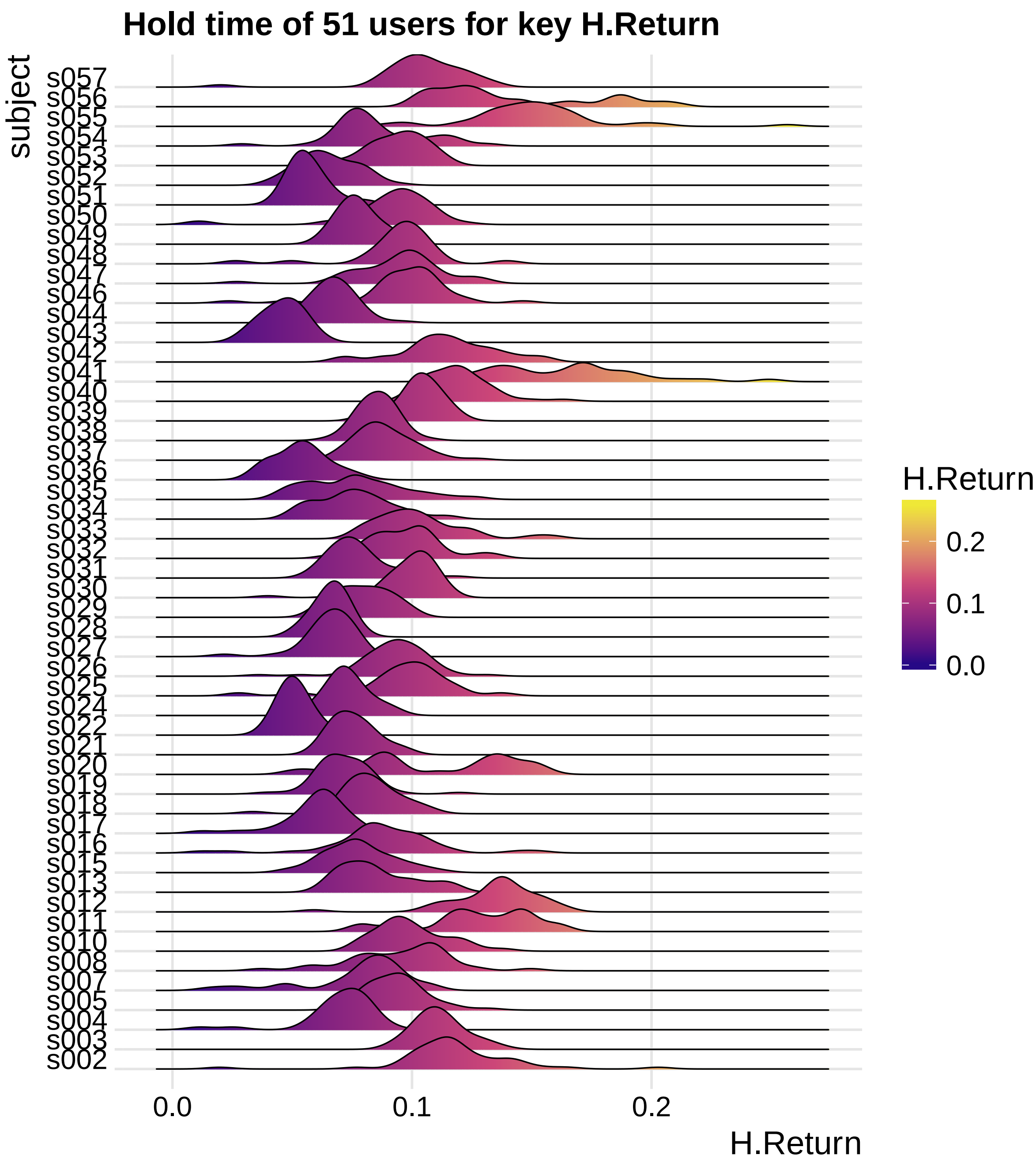




From above graphs we can clearly say that there is difference in typing speed of 3 users. In our dataset we have 51 different users. To look at the difference in distribution of hold times for passcode of all 51 users, we used density ridge plots. We saw in previous scatter plot that for few users timing were overlapping but we are not sure if they have the same timing. From following plots we can clearly see that none of the users have same time distribution and we can say that all 51 users have different time speed for typing passcode.

NOTE : We looked at the distribution plots of all keys in passcode for 51 different users and decided to include only 2 plots in the report. Code for all plots are in rmd file.


Hold time of 51 users for key H.period
subject
s057
s056
s055
s054
s053
s052
s051
s050
s049
s048
s047
s046
s044
s043
s042
s041
s040
s039
s038
s037
s036
s035
s034
s033
s032
s031
s030
s029
s028
s027
s026
s025
s024
s022
s021
s020
s019
s018
s017
s016
s015
s013
s012
s011
s010
s008
s007
s005
s004
s003
s002
0.0
0.1
0.2
H.period
H.period
0.25
0.20
0.15
0.10
0.05
0.00



Next we want to see if the sessionIndex makes any difference in the typing speed of 51 different users. We performed t tests of H variables (since they have normal distribution, we assumed equal variance between the two groups, reps in each group are small) to see if the hold time of keys in passcode is different for a user in session 7 and session 8. As explained before, out of 51 users only 22 users have information in both the sessions hence we will look for only these 22 users. Below is the hypothesis testing stated:

Null hypothesis: The true mean of a user in session 7 and session 8 are same.

Alternative hypothesis: The true mean of a user in session 7 and session 8 are different.

```
##          user005              user7
```

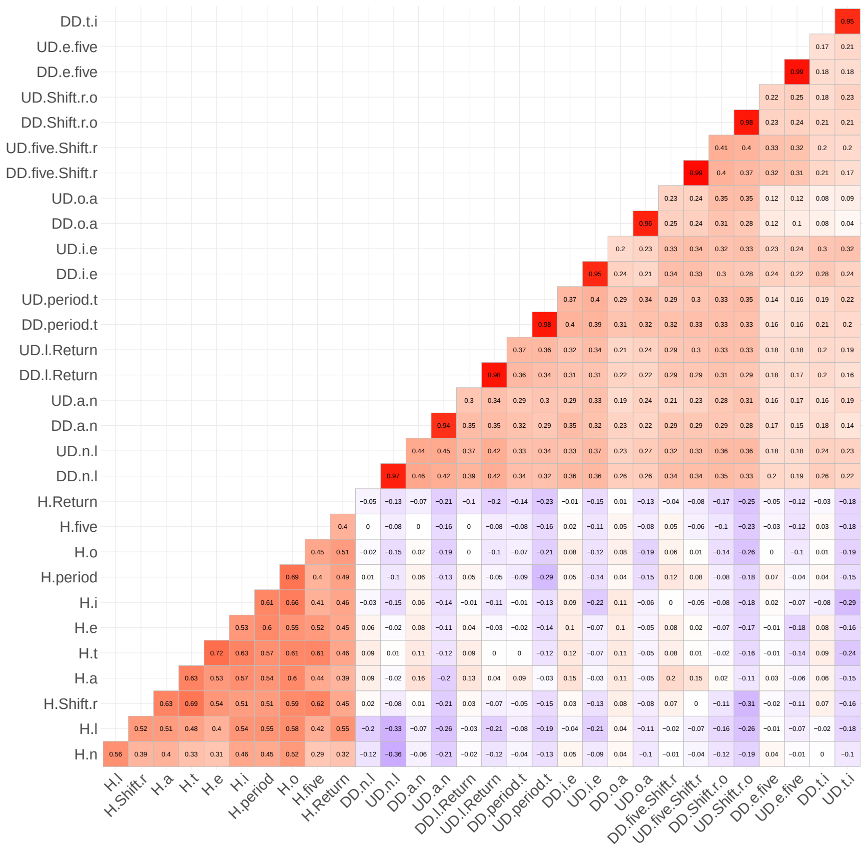
```
## H.period  "0.225579834416413"  "0.0708745587538702"
## H.t       "0.545272645204014"  "0.572557460555183"
## H.i       "0.476874256989693"  "0.0645041952352453"
## H.e       "0.021307921198305"  "0.577592793967863"
## H.five    "0.0924234540223777" "0.509625732677777"
## H.Shift.R "0.306290817106458"  "0.434301719766681"
## H.o       "0.0527723432570757" "0.381313646237267"
## H.a       "0.540384306124343"  "0.247488611171574"
## H.n       "0.279285596854102"  "0.573592323013656"
## H.l       "0.0145530560866159" "0.00135371518612777"
## H.Return  "0.833146451991467"  "0.31824673899244"
##           user8                  user12
## H.period  "0.401363950778124"    "0.333914890073025"
## H.t       "0.139962571246605"    "0.2866039231881"
## H.i       "0.0134458980339771"   "0.0734554273963594"
## H.e       "0.812484220387221"    "0.207009300737539"
## H.five    "0.876418994616681"    "0.154206395273636"
## H.Shift.R "9.82159936634696e-06" "0.817933222590262"
## H.o       "0.00413147536316708"  "0.467881782527705"
## H.a       "0.334021173414902"    "0.699722274707588"
## H.n       "0.00047081789411014"  "0.435506212167137"
## H.l       "0.0463089227299904"   "0.0853198565691602"
## H.Return  "2.18624820878992e-05" "0.609600138471278"
##           user20                 user21
## H.period  "0.00844321461338779"  "0.617655349248006"
## H.t       "0.00162579780525935"  "0.892034810273689"
## H.i       "8.42290761764362e-05" "0.48658576883861"
## H.e       "9.52093647610005e-05" "0.838600690437507"
## H.five    "0.368844455540884"    "0.255342006414334"
## H.Shift.R "0.0116106046162763"   "0.0101974626807535"
## H.o       "9.63330022373983e-05" "0.666246025886427"
## H.a       "1.34198664346111e-09" "0.715201029667283"
## H.n       "0.0592615467594185"   "0.954180656467369"
## H.l       "0.680544021991278"    "0.545242093499037"
## H.Return  "0.000542824417687539" "0.0855340189261078"
##           user24                 user26
## H.period  "1.65823102897691e-05" "0.00352475381397672"
## H.t       "0.189375339061276"    "0.000254560671205553"
## H.i       "0.148053770480619"    "0.0113793245601963"
## H.e       "0.00758490362298845"  "0.00529999515582406"
## H.five    "0.724475636080838"    "0.000356557781545018"
## H.Shift.R "0.343773727824329"    "0.0285097458158995"
## H.o       "0.143579471882819"    "0.000337029920072124"
## H.a       "0.410567997362882"    "0.0499654584138109"
## H.n       "0.378405752240207"    "4.67130042178878e-05"
## H.l       "0.000180407476444732" "0.0361175742239147"
## H.Return  "0.0404869489922391"   "2.32022320147722e-05"
##           user28                 user29
## H.period  "0.166661973620788"    "0.810744140336191"
## H.t       "0.0550667357568428"   "0.894315610148951"
## H.i       "0.000711812474928928" "0.111389141503794"
## H.e       "0.847163362508696"    "0.745851937866652"
## H.five    "0.323317374418668"    "0.030873394042799"
## H.Shift.R "0.430445257732535"    "0.0839103090988839"
```

```
## H.o       "0.162842545081353"    "0.473780770516468"
## H.a       "0.0862600616241837"   "0.27065513975624"
## H.n       "0.000102608201388674" "0.232729794464214"
## H.l       "0.00632890919860983"  "0.0630656311053334"
## H.Return  "0.562727944295733"    "0.0330853053124272"
##           user34                 user36
## H.period  "0.128712238556508"    "1.01620476994585e-06"
## H.t       "0.475706469383037"    "0.146212980569398"
## H.i       "0.408814462127697"    "0.0135582142372756"
## H.e       "0.946886215006021"    "0.532597596116292"
## H.five    "0.852986886758542"    "0.0295664398959654"
## H.Shift.R "0.185055475139817"    "0.02038919381454"
## H.o       "0.00772899813547368"  "0.0239448533855768"
## H.a       "0.489089064009973"    "0.822091558085858"
## H.n       "0.550513665289851"    "0.042140904440006"
## H.l       "0.980235118187839"    "0.0161130405048637"
## H.Return  "0.0766366981477668"   "0.000895162562500594"
##           user39                 user42                 user43
## H.period  "0.850255215335797"    "0.0646206504117537"   "0.57133489283432"
## H.t       "0.224381430915735"    "0.353229656310427"    "0.604635180671272"
## H.i       "0.0444722481932356"   "0.00379306757300097"  "0.495183872620964"
## H.e       "0.951015682785717"    "0.371261404364565"    "0.514575719273243"
## H.five    "0.660125679932164"    "0.35568141353991"     "0.128655233351432"
## H.Shift.R "0.00497412535234979"  "0.0179873840735458"   "0.261090630324435"
## H.o       "0.00480774793516476"  "0.230416303753862"    "0.686560099634664"
## H.a       "0.760693758620665"    "0.025073281839468"    "0.59476226806888"
## H.n       "0.532461259664645"    "0.703658414012703"    "0.234484365067517"
## H.l       "0.479574624730272"    "0.903009450237153"    "0.882438455918992"
## H.Return  "0.870791191642201"    "0.326937778915909"    "0.0580067971553835"
##           user49               user50               user51
## H.period  "0.286719454245317"  "0.241955598797504"  "0.197131165803971"
## H.t       "0.85046045184996"   "0.694385931659781"  "0.187092313868757"
## H.i       "0.108352104058063"  "0.288225127744256"  "2.27216647878969e-05"
## H.e       "0.571752835493471"  "0.675431243798576"  "0.851727916326838"
## H.five    "0.313712207235443"  "0.0146981215717976" "0.0169020857603508"
## H.Shift.R "0.217575704188229"  "0.677842756274673"  "0.349890630839619"
## H.o       "0.805238816330714"  "0.567196139185435"  "0.00867665816667034"
## H.a       "0.89687850094998"   "0.0128310968591037" "6.96833322103789e-05"
## H.n       "0.173833532838457"  "0.9319657581373"    "0.0860166000651058"
## H.l       "0.206706886409095"  "0.154536857809193"  "0.00432087690481967"
## H.Return  "0.22883184819255"   "0.654336973083915"  "0.41112727943859"
##           user52                 user53                 user56
## H.period  "0.466628782321676"    "0.385895674077047"    "0.177236940890561"
## H.t       "0.0627798813073075"   "0.457725910881009"    "0.779770872670086"
## H.i       "0.0352768051730583"   "0.613285279287619"    "0.0896661490318603"
## H.e       "0.000231249531536023" "0.115191833272705"    "0.745836719250522"
## H.five    "0.0030753237890513"   "0.474972710674768"    "0.350143450261018"
## H.Shift.R "0.00019356971662815"  "0.0348872477893859"   "0.28838942168223"
## H.o       "0.00536982300136739"  "0.91434447794166"     "0.952611378644901"
## H.a       "0.424000319014745"    "0.521742867892809"    "0.396817054596021"
## H.n       "0.186752472747405"    "0.771885001540263"    "0.385718665854819"
## H.l       "0.634069011499597"    "0.508921162134625"    "0.946364246744355"
## H.Return  "0.794953232631625"    "0.932185085132435"    "0.158845725728041"
##
```

```
## H.period  "4"
## H.t       "4"
## H.i       "4"
## H.e       "4"
## H.five    "4"
## H.Shift.R "4"
## H.o       "4"
## H.a       "4"
## H.n       "4"
## H.l       "4"
## H.Return  "4"
```

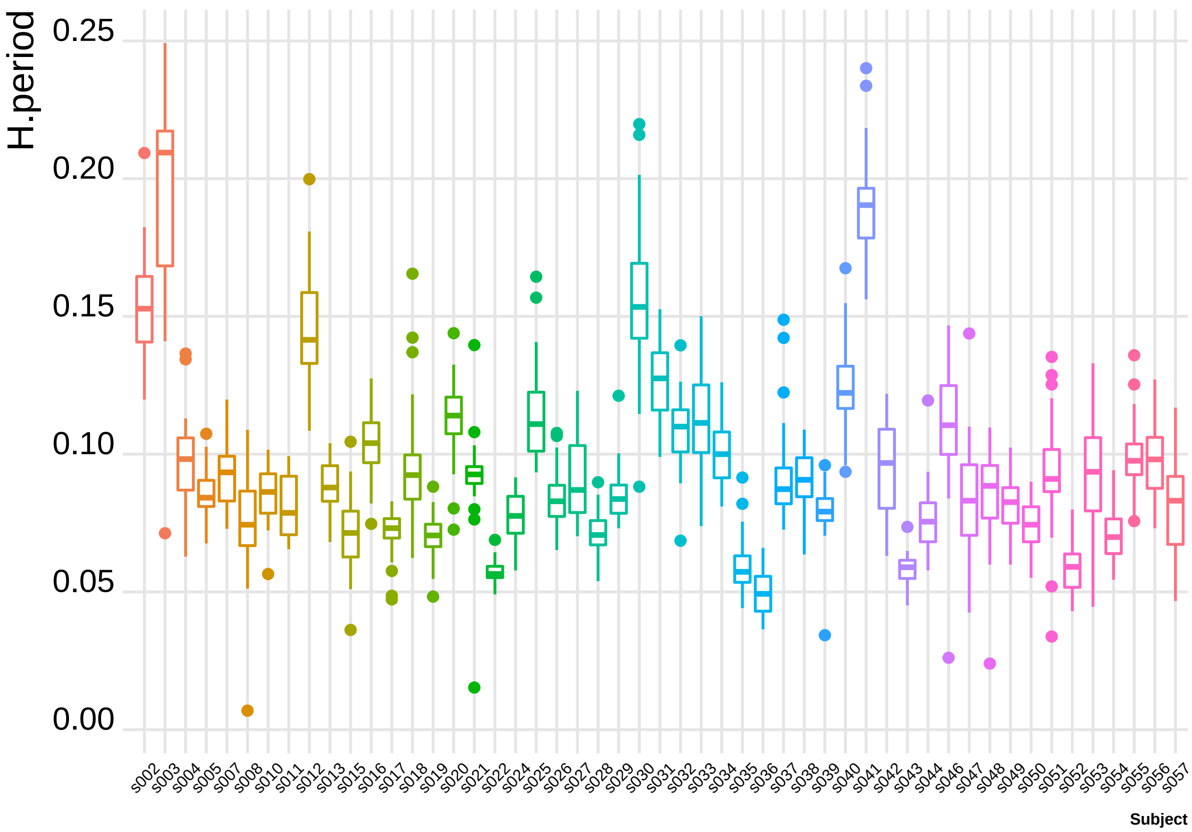
Two data samples are independent if they come from unrelated populations and the samples does not affect each other. Since most of the p-values are greater than 0.05, we fail to reject the null hypothesis and conclude that sessionIndex 7 and 8 does not make any difference in typing speed of users. Since session 7 and session 8 does not make any difference in typing measurements we decided to not include this variable in our model building.

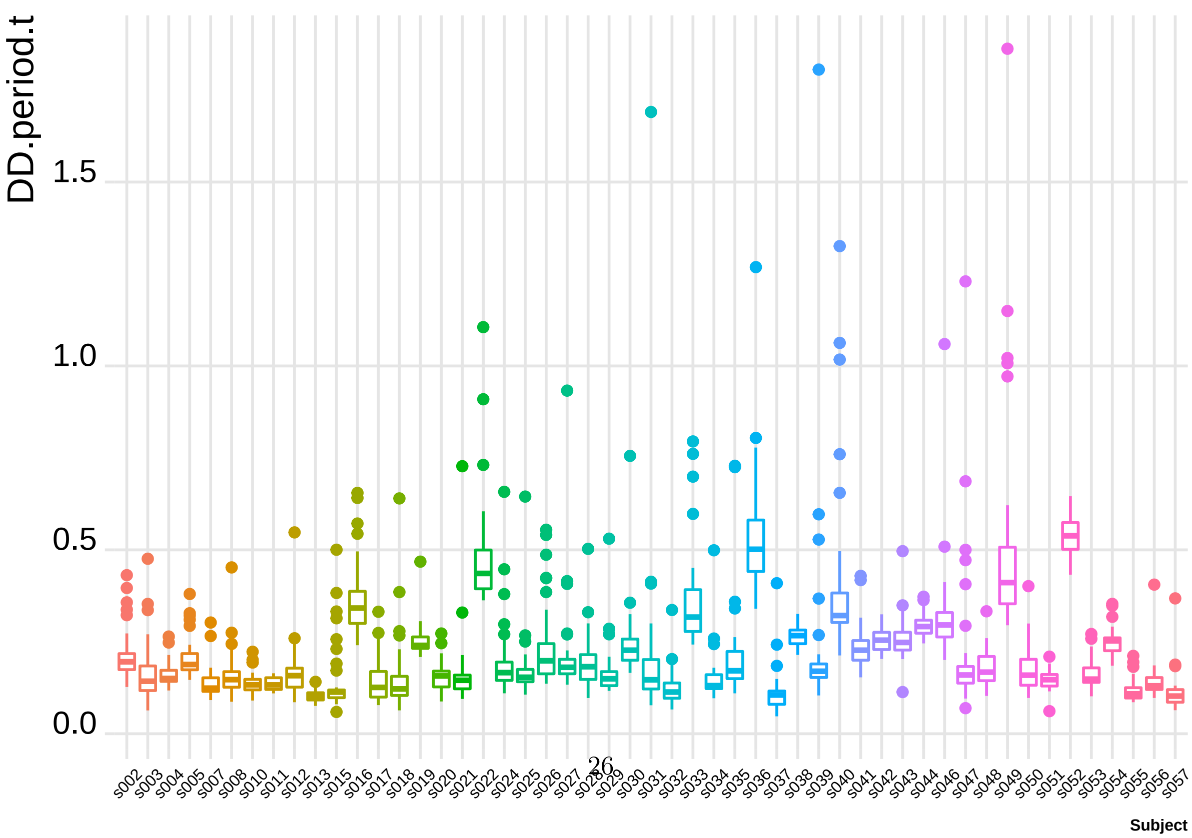
1: Our main goal was to design a good classifier. So, before jumping into directly fitting all the predictors into different classifier methods. We decided to check the correlation between predictors. And, hence the following chart obtained.

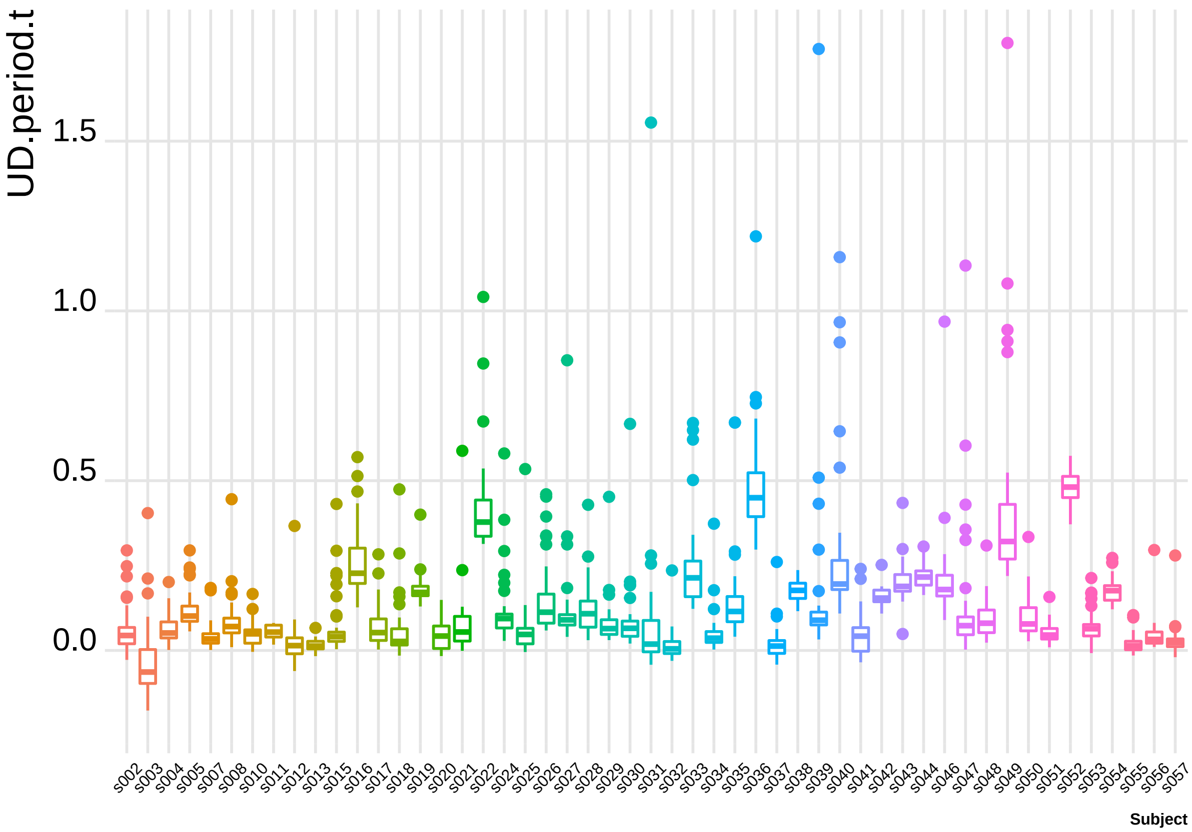


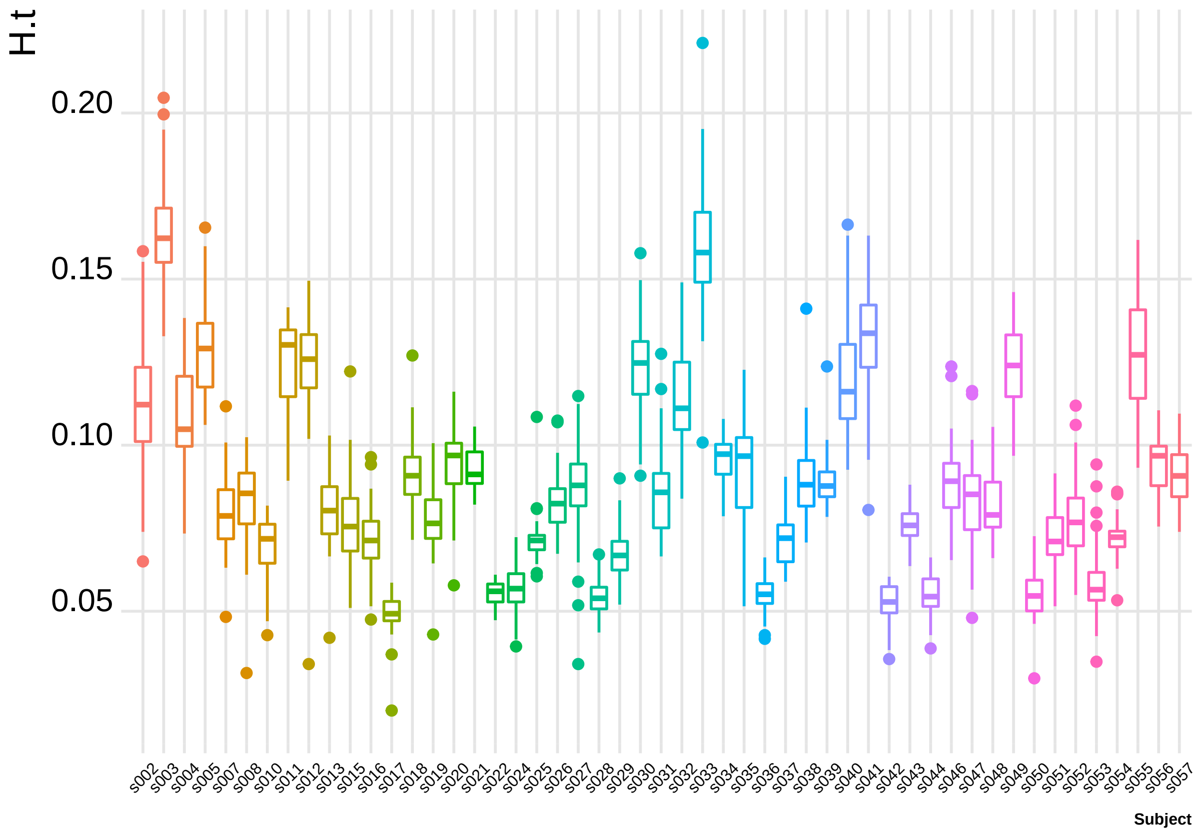
By looking at the above correlation chart, we can easily see that some predictors are highly correlated with each other. For example, the correlation coefficient between **DD.period.t** and **UD.period.t** is 0.98, hence these are highly correlated predictors. In fact, there are other 9 pairs of predictors which are behaving the same nature. Hence, multicolinearity is expected if all the predictors are used to build a classifier.

2: Another method we deployed to obtain importance of predictors is we produced box plots between one predictor versus all the classes on a single plot. And, hence we produced 31 those plots. As to show those 31 boxplots in this report we felt unnecessary, and hence we decided to show following 4 plots.






UD.period.t
1.5
1.0
0.5
0.0
s002 s003 s004 s005 s007 s008 s010 s011 s012 s013 s015 s016 s017 s018 s019 s020 s021 s022 s024 s025 s026 s027 s028 s029 s030 s031 s032 s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s046 s047 s048 s049 s050 s051 s052 s053 s054 s055 s056 s057
Subject



By looking at the above plots we can clearly see that there is a clear distinction between the most of the users to their typing speeds when they were helding down the password keys **period** and **t**. However, there is not clear distinction between the most of the users on such that the time between pressing down the **.** key to the time to press down the **t** key, and the time between the **.** key coming up to the time to press down the **t** key. We obtained the same patterns as mentioned in preceding sentences. Hence, we somehow convienced that those **H** predictors would play the significant roles to build a classifier.

3 Obtaining Classifiers

We used createDataPartition method from caret package which creates balanced splits of the data. The random sampling occurs within each class and it preserves the overall class distribution of the data. Below table shows 51 classes and count of observations in each class for original data, train data and split data.

i: Multinomial logistics regression

Multinomial logistic regression is a classification method used for multiclass classification problems where the dependent variable is nominal which in our case is subject variable having 51 classes. We are trying to make predictions based on linear combinations of the observed features including all the available variables, only H variables and combination of H and UD variables. We preented Model accuracy and class specific accuracy in our analysis. The multinomial logistics Model accuracy is presented in Table 1 below:

Table 1: Accuracy for GLM Model with H predictors only:

Model	Methods	Accuracy

Model	Methods	Accuracy
Glm using multinom function	Train	0.848987108655617
Glm using multinom function	Test	0.776811594202899
Glm using multinom function	5-fold	0.782113

The model accuracy on test dataset achieved is 77.68116%. We tried 5-fold cross validation approach to look for any improvements in model performance. The model accuracy improved slightly, and it is reported as 78.2113 %. The class S054 have the lowest class specific accuracy of 42.86%, originally it had 14 reps and it predicted only 6. Moreover, we can see most of the class specific accuracy is low. Hence we can conclude that multinomial logistics model performance tends to decrease for the response variable with more than 2 classes. Next we will look how LDA performs, since it is a popular method for multi-class classification problems with more than 2 classes.

ii : LDA

Linear Discriminate Analysis (LDA) is a method that uses a linear combination of the observed variables to separate the observations into 2 (or more) classes. A key feature of the LDA is the assumption that the distribution of each individual class type is normal. The method for linear discriminate analysis was conducted using the "lda" function from the "MASS" library in R. The same basic structure was used for model building and accuracy estimating. The accuracy results are presented in Table 2 below.

Test accuracy for LDA model with H predictors only:

Table 2: Accuracy Table for LDA having only H predictors:

Model	Method	Accuracy
LDA having only H predictors	Train	0.818600368324125
LDA having only H predictors	Test	0.815942028985507
LDA having only H predictors	LOOCV	0.810380645177326
LDA having only H predictors	5-fold	0.811373873873874

From Table 2, we observed that the test accuracy (81.6%) performs slightly better than LOOVC (81%) and k-fold (81.1%). With regards to the subject specific accuracy, the model performs well on the subject (s0011) with the lowest rep of 4. The model correctly predicted 3 out of 4 reps (75%) in s0011. This shows that the model is performing well in terms of predicting small reps

iii: LDA Model with H and UD predictors

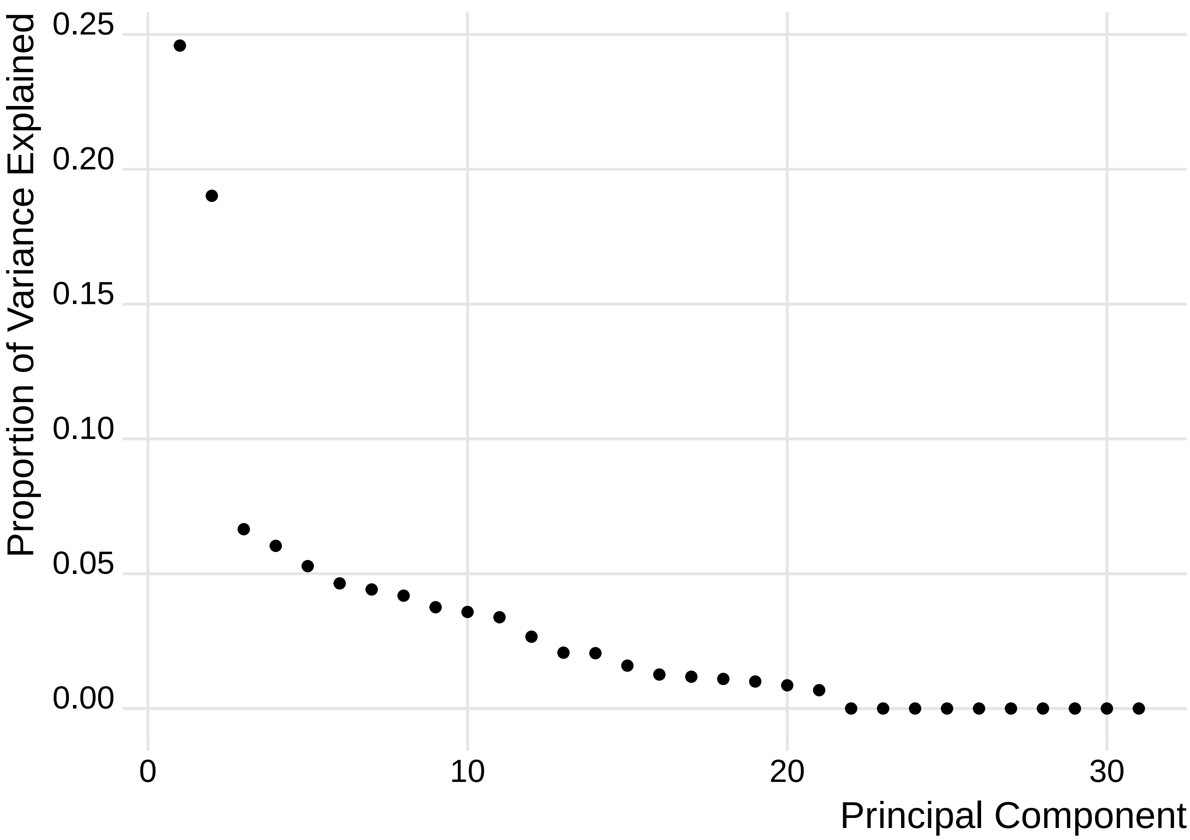
Table 3: Accuracy for LDA having only H and UD predictors:

Model	Method	Accuracy
LDA having only H and UD predictors	Train	0.901473296500921
LDA having only H and UD predictors	Test	0.879710144927536
LDA having only H and UD predictors	LOOCV	0.880673834502745
LDA having only H and UD predictors	5-fold	0.873873873873874

Table 3 above displayed the accuracy rate of LDA model with H and UD predictors only. It showed that the LOOCV is performing well in term of overall accuracy (88.1%) as compared to test accuracy and K-fold approach. The subject specific accuracy in Appendix showed that most of the subject specific accuracy are very high with some correctly predicting all the reps within a subject especially for s010, s017, s022, s024, s025 obtaining 100% subject specific accuracy rate. This shows that the model is performing well.

iv: Implementation of PCA on LDA

The scree plot for PCA can be obtained as:



By looking at the scree plot we can say that 21 principle componests have explained almost 100% variablity on the data. Hence using 21 principle components for further analysis.

```
## [1] 0.09363296
```

```
## [1] 0.1426554
```

Table 4: Test accuracy for PCA on LDA model

Model	Method	Accuracy
Applying PCA on LDA	Train	0.0936329588014981
Applying PCA on LDA	Test	0.142655367231638

V : Bagging and Random Forest using all predictors

Vi: Bagging

Bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree on individual copy, and then combining all the trees in order to create a single predictive model. In bagging,each tree is built on a bootstrap data set which is independent of the other trees. Generally, it considers m = p (number of predictors in the model). Implementing a bagging model on the train dataset and assessing the training and testing accuracy, k-fold and loocv yields the output shown in Table 4 below.

Table 5: Accuracy for bagging can be given as:

Model	Method	Accuracy
Bagging having all predictors	Train	1
Bagging having all predictors	Test	0.921739130434783
Bagging having all predictors	LOOCV	0.91441441
Bagging having all predictors	5-fold	0.939708510982033

Table 4 displayed the accuracy rate of model with bagging. The k-fold approach appears to be performing well in term of overall accuracy (93.8%) as compared to test accuracy (92.17%) and LOOCV approach (91.44%). The subject specific accuracy in the bagging model seems to be performing better than the LD model, with greater proportion of the subject specific accuracy being 100%. This model performs better than LDA model with H and UD predictors.

Vi: Random Forest

Random Forest is an ensemble model that constructs a number of decision trees at training time and output the class that is the mode of the classes output by individual trees. It basically averages the predictions made by tree models to make predictions. Random Forest gives more improved prediction compared to boosting. It considers the important variables first by building randomly different tree models. It considers m = sqrt(p) in the place of mtry for random forests of classification trees, where mtry is number of predictors in the model. Implementing a random forest model on the train dataset and assessing the training and testing accuracy of the model yields the output shown in Table below.

Table 6: Accuracy for Random forest model:

Model	Method	Accuracy
Rf having all predictors	Train	1
Rf having all predictors	Test	0.957971014492754
Rf having all predictors	K-fold	0.782094362962576
Rf having all predictors	LOOCV	0.96171171

In Table 5 above, it can be observed that loocv method records the highest accuracy rate of 96.2% compared to test accuracy (96%). K-fold performed below test and loocv with 78.2% accuracy rate. The subject specific accuracy in the random forest model seems to be performing well with majority of the subjects correctly predicting the reps in each subject.

Vii: Boosting

Unlike bagging, in boosting the subset creation is not random and depends upon the performance of the previous models: every new subsets contains the elements that were (likely to be) misclassified by previous models. Hence, in boosting, the trees are grown sequentially: that is each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.

```
## [1] 0.0893186
```

```
## [1] 0.1768116
```

Table 7: Test accuracy for boosting model

Model	Method	Accuracy
Bagging having all predictors	Train	0.0893186003683242
Bagging having all predictors	Test	0.176811594202899

In Table 6, overall accuracy for test data is 78%, which is quite lower than random forest model and bagging model. The class S054 have the lowest class specific accuracy of 42.86%, originally it had 14 reps and it predicted only 6. Moreover, we can see most of the class specific accuracy is low. Hence we can conclude that multinomial logistics model performance tends to decrease for the response variable with more than 2 classes. Next we will look how LDA performs, since it is a popular method for multi-class classification problems with more than 2 classes.

Viii :SVM Full model

Table 8: Test accuracy for SVM full model

Models	Methods	Accuracy
SVM using linear kernel	Train Accuray	97.4234
SVM using linear kernel	Test Accuracy	85.29412
SVM using poly kernel	Train Accuray	97.4234
SVM using poly kernel	Test Accuracy	85.29412
SVM using radial kernel	Train Accuray	99.3036
SVM using radial kernel	Test Accuracy	88.82353

Ix: SVM having H and UD:

SVM with kernel value radial performed well compared to SVM kernel values linear and poly. There is not much difference between model accuracy of SVM linear and SVM radial, hence let us look at the class specific accuracy of predicted data using both model.

Table 9: Test accuracy for SVM having H and UD:

Models	Methods	Accuracy
SVM using radial kernel	2.99442896935933	Train
SVM using radial kernel	2.94117647058824	Test set

Selection of model:

Base on model performance (accuracy rate), our recommended model is the random forest. This is because the model possesses high overall predictive accuracy (96.2%) and subject specific prediction of individual reps are very high. The subject specific accuracy in the random forest model have majority predicting 100% of the reps in each subject. Also, considering subjects with smaller reps, the model predicted greater proportion for subjects with smaller reps.

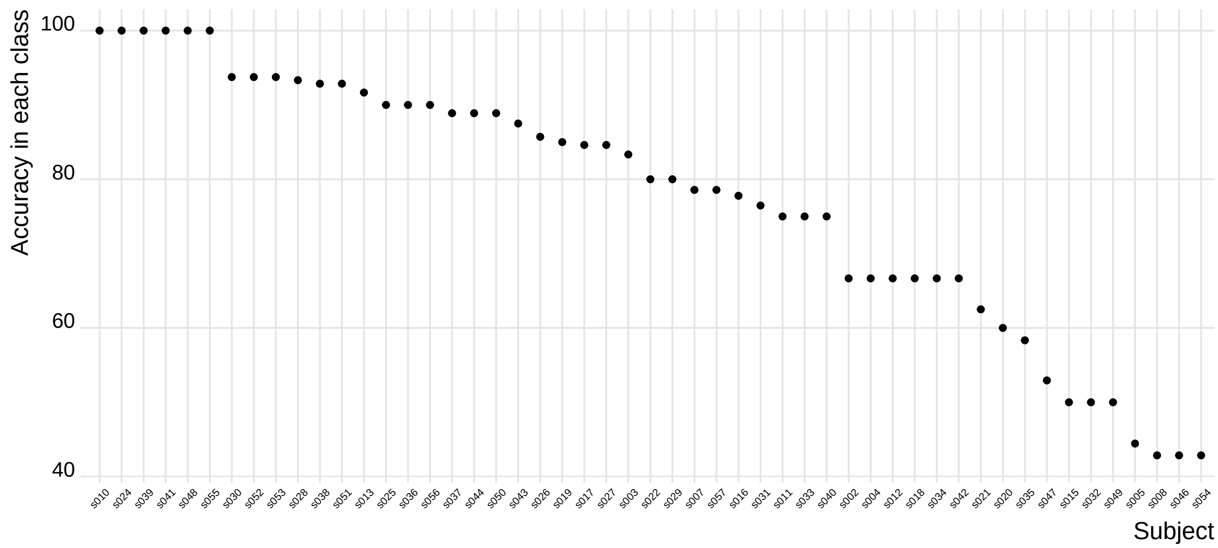
Appendice

Class specific performances on test data using Multinom() function i.e GLM metthod:

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	18	12	66.66667
s003	12	10	83.33333
s004	12	8	66.66667
s005	18	8	44.44444
s007	14	11	78.57143
s008	14	6	42.85714
s010	12	12	100.00000
s011	4	3	75.00000
s012	15	10	66.66667
s013	12	11	91.66667
s015	16	8	50.00000
s016	18	14	77.77778
s017	13	11	84.61538
s018	15	10	66.66667
s019	20	17	85.00000
s020	10	6	60.00000
s021	8	5	62.50000
s022	10	8	80.00000
s024	13	13	100.00000
s025	10	9	90.00000
s026	21	18	85.71429
s027	13	11	84.61538
s028	15	14	93.33333
s029	10	8	80.00000
s030	16	15	93.75000
s031	17	13	76.47059
s032	10	5	50.00000
s033	12	9	75.00000
s034	9	6	66.66667
s035	12	7	58.33333
s036	10	9	90.00000
s037	18	16	88.88889
s038	14	13	92.85714
s039	12	12	100.00000
s040	16	12	75.00000
s041	14	14	100.00000
s042	6	4	66.66667
s043	8	7	87.50000
s044	18	16	88.88889

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s046	14	6	42.85714
s047	17	9	52.94118
s048	10	10	100.00000
s049	14	7	50.00000
s050	18	16	88.88889
s051	14	13	92.85714
s052	16	15	93.75000
s053	16	15	93.75000
s054	14	6	42.85714
s055	18	18	100.00000
s056	10	9	90.00000
s057	14	11	78.57143

Accuracy in each class vs subject for GLM

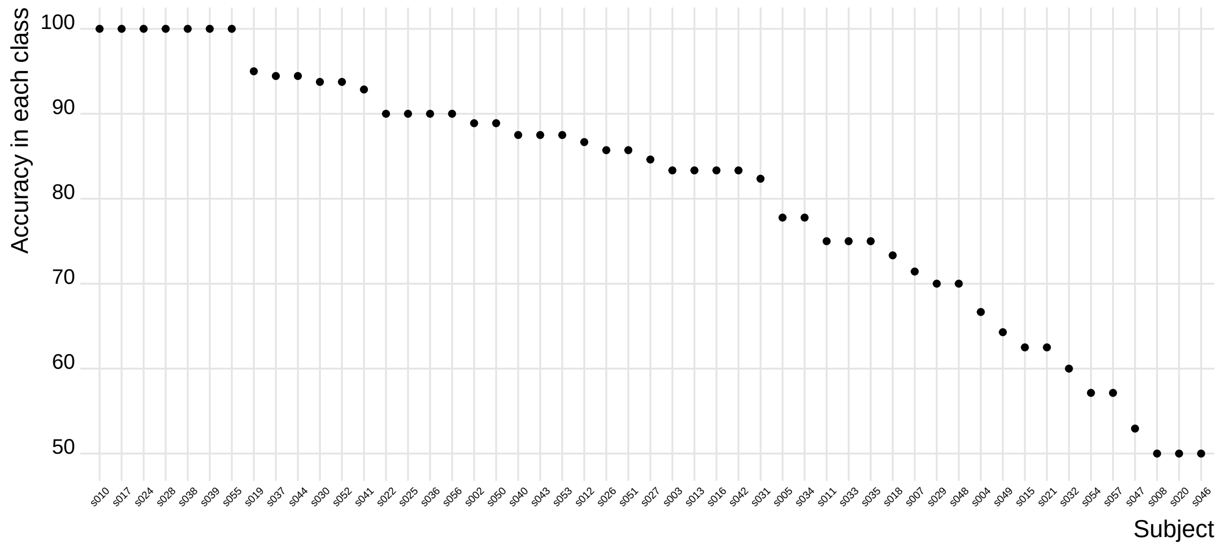


Class specific performances on test data for LDA having only H predictors:

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	18	16	88.88889
s003	12	10	83.33333
s004	12	8	66.66667
s005	18	14	77.77778
s007	14	10	71.42857
s008	14	7	50.00000
s010	12	12	100.00000
s011	4	3	75.00000
s012	15	13	86.66667
s013	12	10	83.33333
s015	16	10	62.50000
s016	18	15	83.33333
s017	13	13	100.00000
s018	15	11	73.33333

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s019	20	19	95.00000
s020	10	5	50.00000
s021	8	5	62.50000
s022	10	9	90.00000
s024	13	13	100.00000
s025	10	9	90.00000
s026	21	18	85.71429
s027	13	11	84.61538
s028	15	15	100.00000
s029	10	7	70.00000
s030	16	15	93.75000
s031	17	14	82.35294
s032	10	6	60.00000
s033	12	9	75.00000
s034	9	7	77.77778
s035	12	9	75.00000
s036	10	9	90.00000
s037	18	17	94.44444
s038	14	14	100.00000
s039	12	12	100.00000
s040	16	14	87.50000
s041	14	13	92.85714
s042	6	5	83.33333
s043	8	7	87.50000
s044	18	17	94.44444
s046	14	7	50.00000
s047	17	9	52.94118
s048	10	7	70.00000
s049	14	9	64.28571
s050	18	16	88.88889
s051	14	12	85.71429
s052	16	15	93.75000
s053	16	14	87.50000
s054	14	8	57.14286
s055	18	18	100.00000
s056	10	9	90.00000
s057	14	8	57.14286

Accuracy in each class vs subject for LDA only H predictors

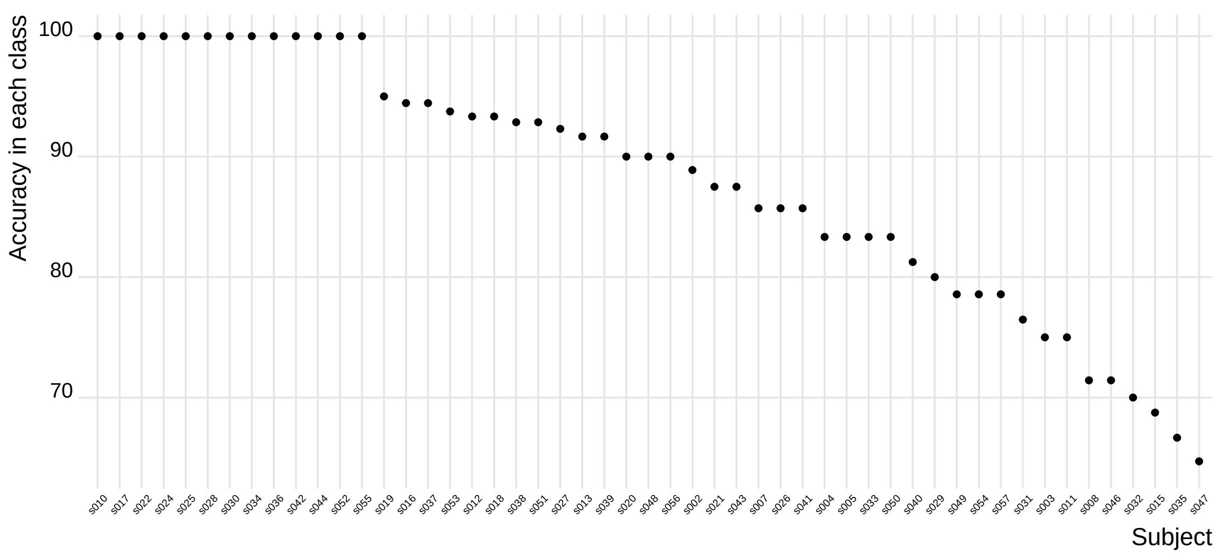


Class specific performances on test data for LDA having only H and UD predictors:

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	18	16	88.88889
s003	12	9	75.00000
s004	12	10	83.33333
s005	18	15	83.33333
s007	14	12	85.71429
s008	14	10	71.42857
s010	12	12	100.00000
s011	4	3	75.00000
s012	15	14	93.33333
s013	12	11	91.66667
s015	16	11	68.75000
s016	18	17	94.44444
s017	13	13	100.00000
s018	15	14	93.33333
s019	20	19	95.00000
s020	10	9	90.00000
s021	8	7	87.50000
s022	10	10	100.00000
s024	13	13	100.00000
s025	10	10	100.00000
s026	21	18	85.71429
s027	13	12	92.30769
s028	15	15	100.00000
s029	10	8	80.00000
s030	16	16	100.00000
s031	17	13	76.47059
s032	10	7	70.00000
s033	12	10	83.33333
s034	9	9	100.00000
s035	12	8	66.66667
s036	10	10	100.00000

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s037	18	17	94.44444
s038	14	13	92.85714
s039	12	11	91.66667
s040	16	13	81.25000
s041	14	12	85.71429
s042	6	6	100.00000
s043	8	7	87.50000
s044	18	18	100.00000
s046	14	10	71.42857
s047	17	11	64.70588
s048	10	9	90.00000
s049	14	11	78.57143
s050	18	15	83.33333
s051	14	13	92.85714
s052	16	16	100.00000
s053	16	15	93.75000
s054	14	11	78.57143
s055	18	18	100.00000
s056	10	9	90.00000
s057	14	11	78.57143

Accuracy in each class vs subject for LDA having only H and UD predictors:

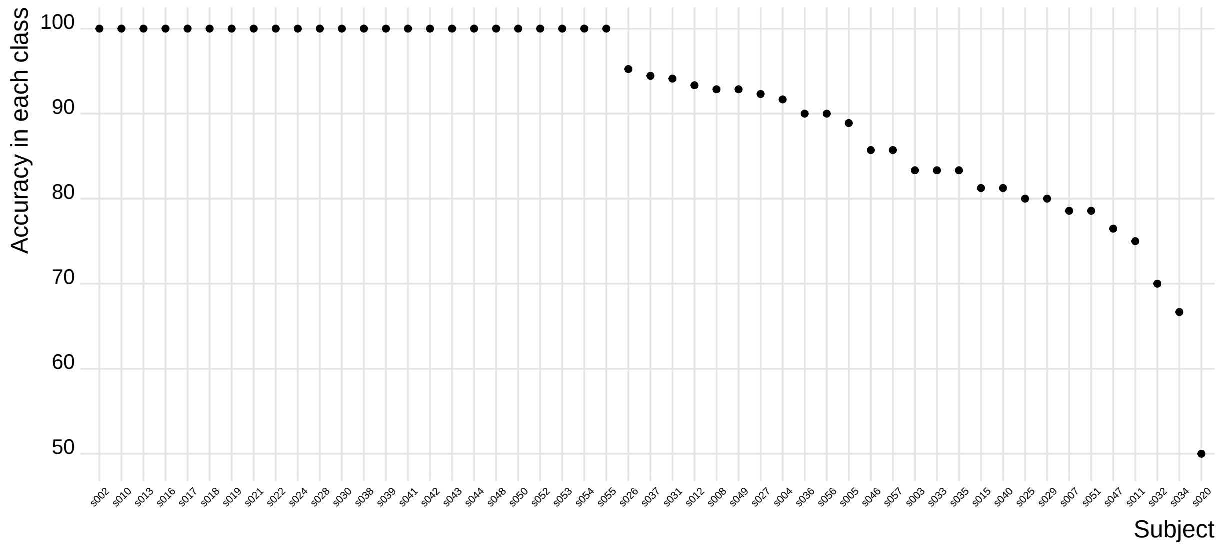


Class specific performances on test data for bagging can be given by:

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	18	18	100.00000
s003	12	10	83.33333
s004	12	11	91.66667
s005	18	16	88.88889
s007	14	11	78.57143
s008	14	13	92.85714

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s010	12	12	100.00000
s011	4	3	75.00000
s012	15	14	93.33333
s013	12	12	100.00000
s015	16	13	81.25000
s016	18	18	100.00000
s017	13	13	100.00000
s018	15	15	100.00000
s019	20	20	100.00000
s020	10	5	50.00000
s021	8	8	100.00000
s022	10	10	100.00000
s024	13	13	100.00000
s025	10	8	80.00000
s026	21	20	95.23810
s027	13	12	92.30769
s028	15	15	100.00000
s029	10	8	80.00000
s030	16	16	100.00000
s031	17	16	94.11765
s032	10	7	70.00000
s033	12	10	83.33333
s034	9	6	66.66667
s035	12	10	83.33333
s036	10	9	90.00000
s037	18	17	94.44444
s038	14	14	100.00000
s039	12	12	100.00000
s040	16	13	81.25000
s041	14	14	100.00000
s042	6	6	100.00000
s043	8	8	100.00000
s044	18	18	100.00000
s046	14	12	85.71429
s047	17	13	76.47059
s048	10	10	100.00000
s049	14	13	92.85714
s050	18	18	100.00000
s051	14	11	78.57143
s052	16	16	100.00000
s053	16	16	100.00000
s054	14	14	100.00000
s055	18	18	100.00000
s056	10	9	90.00000
s057	14	12	85.71429

Accuracy in each class vs subject for bagging:

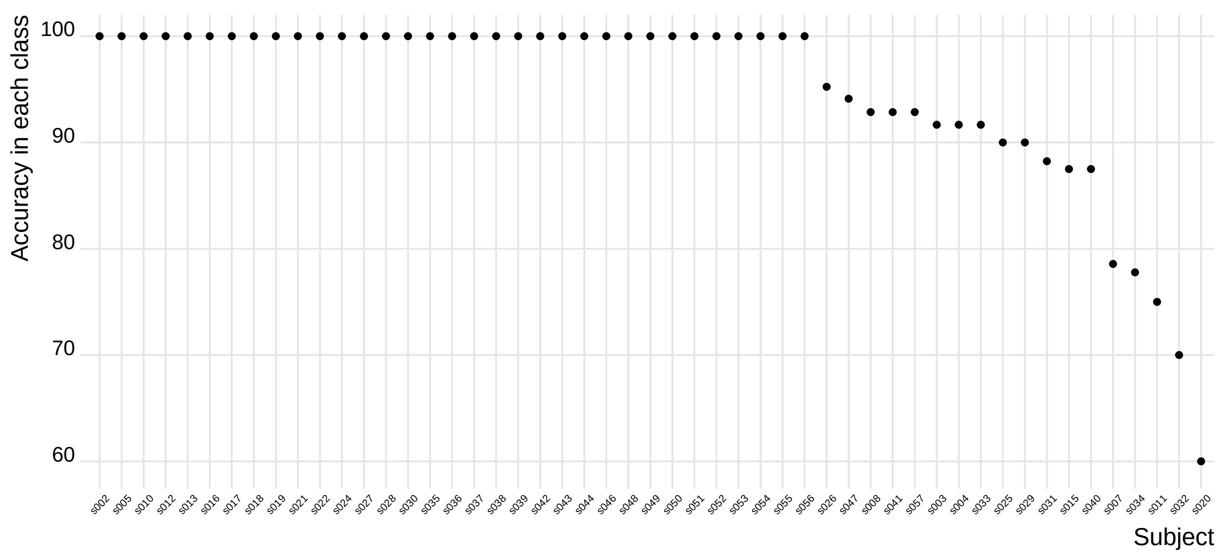


Class specific performances on test data for random forest can be given by:

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	18	18	100.00000
s003	12	11	91.66667
s004	12	11	91.66667
s005	18	18	100.00000
s007	14	11	78.57143
s008	14	13	92.85714
s010	12	12	100.00000
s011	4	3	75.00000
s012	15	15	100.00000
s013	12	12	100.00000
s015	16	14	87.50000
s016	18	18	100.00000
s017	13	13	100.00000
s018	15	15	100.00000
s019	20	20	100.00000
s020	10	6	60.00000
s021	8	8	100.00000
s022	10	10	100.00000
s024	13	13	100.00000
s025	10	9	90.00000
s026	21	20	95.23810
s027	13	13	100.00000
s028	15	15	100.00000
s029	10	9	90.00000
s030	16	16	100.00000
s031	17	15	88.23529
s032	10	7	70.00000
s033	12	11	91.66667
s034	9	7	77.77778
s035	12	12	100.00000
s036	10	10	100.00000

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s037	18	18	100.00000
s038	14	14	100.00000
s039	12	12	100.00000
s040	16	14	87.50000
s041	14	13	92.85714
s042	6	6	100.00000
s043	8	8	100.00000
s044	18	18	100.00000
s046	14	14	100.00000
s047	17	16	94.11765
s048	10	10	100.00000
s049	14	14	100.00000
s050	18	18	100.00000
s051	14	14	100.00000
s052	16	16	100.00000
s053	16	16	100.00000
s054	14	14	100.00000
s055	18	18	100.00000
s056	10	10	100.00000
s057	14	13	92.85714

Accuracy in each class vs subject for random forest can be given by:

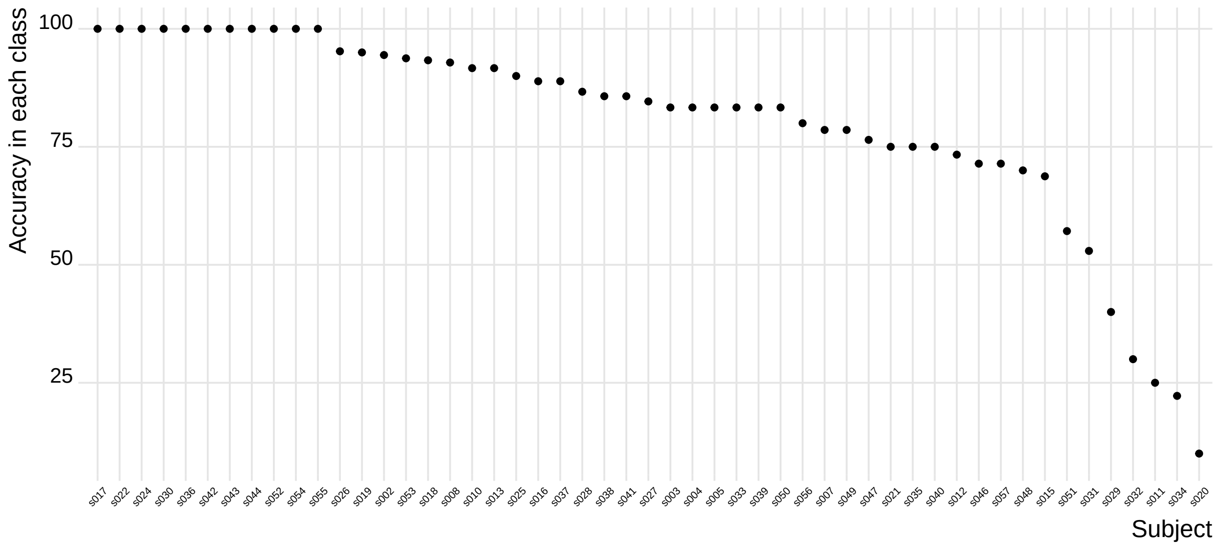


Class specific performances on test data for boosting can be given by:

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	18	17	94.44444
s003	12	10	83.33333
s004	12	10	83.33333
s005	18	15	83.33333
s007	14	11	78.57143
s008	14	13	92.85714

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s010	12	11	91.66667
s011	4	1	25.00000
s012	15	11	73.33333
s013	12	11	91.66667
s015	16	11	68.75000
s016	18	16	88.88889
s017	13	13	100.00000
s018	15	14	93.33333
s019	20	19	95.00000
s020	10	1	10.00000
s021	8	6	75.00000
s022	10	10	100.00000
s024	13	13	100.00000
s025	10	9	90.00000
s026	21	20	95.23810
s027	13	11	84.61538
s028	15	13	86.66667
s029	10	4	40.00000
s030	16	16	100.00000
s031	17	9	52.94118
s032	10	3	30.00000
s033	12	10	83.33333
s034	9	2	22.22222
s035	12	9	75.00000
s036	10	10	100.00000
s037	18	16	88.88889
s038	14	12	85.71429
s039	12	10	83.33333
s040	16	12	75.00000
s041	14	12	85.71429
s042	6	6	100.00000
s043	8	8	100.00000
s044	18	18	100.00000
s046	14	10	71.42857
s047	17	13	76.47059
s048	10	7	70.00000
s049	14	11	78.57143
s050	18	15	83.33333
s051	14	8	57.14286
s052	16	16	100.00000
s053	16	15	93.75000
s054	14	14	100.00000
s055	18	18	100.00000
s056	10	8	80.00000
s057	14	10	71.42857

Accuracy in each class vs subject for boosting can be given by:

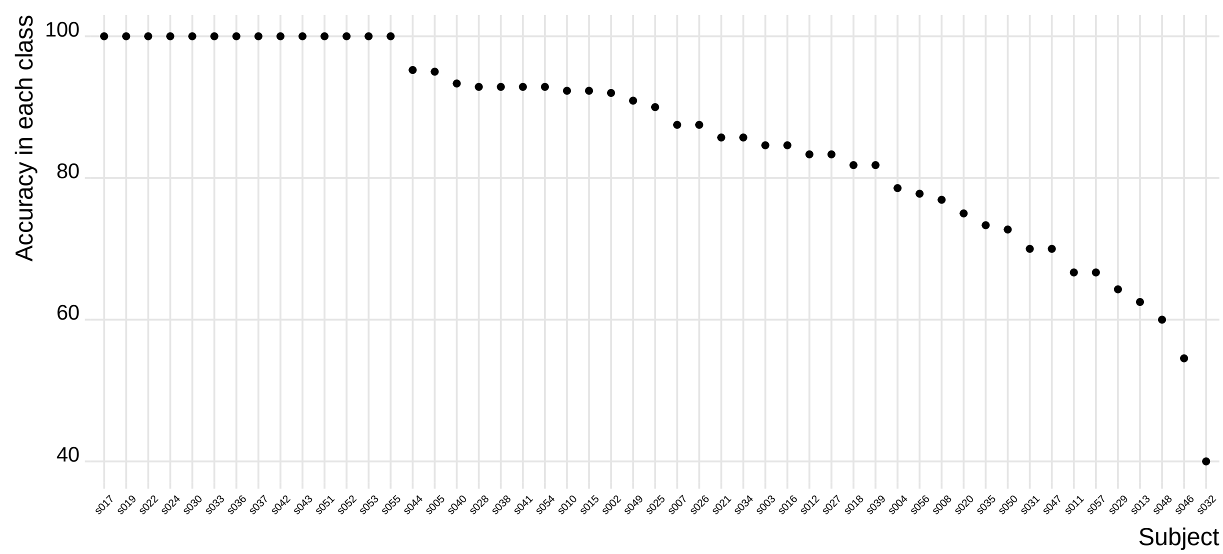


class cpecific performance of PCA on LDA

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s002	25	23	92.00000
s003	13	11	84.61538
s004	14	11	78.57143
s005	20	19	95.00000
s007	16	14	87.50000
s008	13	10	76.92308
s010	13	12	92.30769
s011	3	2	66.66667
s012	18	15	83.33333
s013	8	5	62.50000
s015	13	12	92.30769
s016	13	11	84.61538
s017	14	14	100.00000
s018	11	9	81.81818
s019	20	20	100.00000
s020	12	9	75.00000
s021	7	6	85.71429
s022	11	11	100.00000
s024	12	12	100.00000
s025	10	9	90.00000
s026	24	21	87.50000
s027	12	10	83.33333
s028	14	13	92.85714
s029	14	9	64.28571
s030	16	16	100.00000
s031	20	14	70.00000
s032	10	4	40.00000
s033	11	11	100.00000
s034	7	6	85.71429
s035	15	11	73.33333

	Number of repetitions in each class on test data	Correctly predicted by the model	Accuracy in each class
s036	8	8	100.00000
s037	15	15	100.00000
s038	14	13	92.85714
s039	11	9	81.81818
s040	15	14	93.33333
s041	14	13	92.85714
s042	9	9	100.00000
s043	5	5	100.00000
s044	21	20	95.23810
s046	22	12	54.54545
s047	20	14	70.00000
s048	10	6	60.00000
s049	11	10	90.90909
s050	22	16	72.72727
s051	10	10	100.00000
s052	15	15	100.00000
s053	17	17	100.00000
s054	14	13	92.85714
s055	19	19	100.00000
s056	9	7	77.77778
s057	18	12	66.66667

Accuracy in each class vs subject for pca + lda



class specific performance of SVM

```
## [1] "Number of repetitions in each class on test data"
## [2] "Correctly predicted by the model"
```

	Reps of test data	Correctly predicted by the model	Accuracy in each class
s002	18	0	0.00000
s003	12	0	0.00000

	Reps of test data	Correctly predicted by the model	Accuracy in each class
s004	12	0	0.00000
s005	18	0	0.00000
s007	14	0	0.00000
s008	14	0	0.00000
s010	12	0	0.00000
s011	4	0	0.00000
s012	15	0	0.00000
s013	12	0	0.00000
s015	16	0	0.00000
s016	18	0	0.00000
s017	13	0	0.00000
s018	15	0	0.00000
s019	20	0	0.00000
s020	10	0	0.00000
s021	8	0	0.00000
s022	10	0	0.00000
s024	13	0	0.00000
s025	10	0	0.00000
s026	21	10	47.61905
s027	13	0	0.00000
s028	15	0	0.00000
s029	10	0	0.00000
s030	16	0	0.00000
s031	17	0	0.00000
s032	10	0	0.00000
s033	12	0	0.00000
s034	9	0	0.00000
s035	12	0	0.00000
s036	10	0	0.00000
s037	18	0	0.00000
s038	14	0	0.00000
s039	12	0	0.00000
s040	16	0	0.00000
s041	14	0	0.00000
s042	6	0	0.00000
s043	8	0	0.00000
s044	18	0	0.00000
s046	14	0	0.00000
s047	17	0	0.00000
s048	10	0	0.00000
s049	14	0	0.00000
s050	18	0	0.00000
s051	14	0	0.00000
s052	16	0	0.00000
s053	16	0	0.00000
s054	14	0	0.00000
s055	18	0	0.00000
s056	10	0	0.00000
s057	14	0	0.00000