Assignment 1: ALDA

Abhilasha Kumar

September 6, 2017

1 Reading the Data

```
> cell = read.csv("cell_finaldata.csv", header = TRUE, sep = ",")
```

2 Formatting the Data

The data is originally in long format. Each participant has differing number of items, and different number of assessments for each month. I'm trying to convert this data into wide format such that each participant has all its data into one row. First, I will select only the main variables, to simplify the dataset.

```
> cell_judgments = cell[, c(75, 29,  97,98,4, 82,83,85,86, 57, 103)]
> colnames(cell_judgments) = c("ID", "Days", "Accuracy", "Month", "Messages",
+                                  "RecallType", "RecallBeforeHint", "Vividness",
+                              "GuessedMonth", "Memorability", "NumItems")
> ## next we need to create an "itemno" variable, so that conversion to wide format is made easier
>
> N = nrow(cell_judgments)
> by_id = split(cell, cell_judgments$ID)
> newcomplete <- matrix(NA, nrow = N, ncol = 4)
> for(i in seq(along = by_id)) {
+
+   id = names(by_id)[i];
+   #created dataset for each subject
+
+   id_subjectdata = cell_judgments[cell_judgments$ID == id,]
+   newcomplete = id_subjectdata[,c('ID', 'Days')]
+
+   for(j in 1:nrow(id_subjectdata)){
+     newcomplete[j,"ItemNo"] =  j
+   }
+
+   newcomplete$Memorablity = id_subjectdata$Memorability
+   newcomplete$Accuracy = id_subjectdata$Accuracy
+   newcomplete$Messages = id_subjectdata$Messages
+   newcomplete$RecallBeforeHint = id_subjectdata$RecallBeforeHint
+   newcomplete$Vividness = id_subjectdata$Vividness
+   newcomplete$GuessedMonth = id_subjectdata$GuessedMonth
+   newcomplete$NumItems = id_subjectdata$NumItems
+   newcomplete$Month = id_subjectdata$Month
+   newcomplete$GuessedMonth = id_subjectdata$GuessedMonth
+
+   if (i == 1)
+
+    result = newcomplete
```

```
+  else
+
+   result = rbind(result, newcomplete)
+
+ }
> cell_withitems <- result
> cell_withitems <- as.data.frame(cell_withitems)
> cell_withitems$Month = cell_withitems$Month + 1
> cell_withitems$TimeJudgmentDistance = abs(cell_withitems$Month - cell_withitems$GuessedMonth)
> print(head(cell_withitems))
```

```
   ID Days ItemNo Memorablity Accuracy Messages
1 103   17      1  11.0000000        1       19
2 103   14      2   7.4285717        1        6
3 103   68      3  30.7142870        1       11
4 103  279      4  21.0000000        0        3
5 103   52      5  30.0000000        1        4
6 103  203      6   0.5714285        0        2
  RecallBeforeHint Vividness GuessedMonth NumItems Month
1                1        10            1       30     1
2                3        10            1       30     1
3                1         9            3       30     3
4                5         5            9       30    10
5                1        10            2       30     2
6                4         9            7       30     7
  TimeJudgmentDistance
1                    0
2                    0
3                    0
4                    1
5                    0
6                    0
```

```
> ## this dataset has all the necessary variables required for further analysis, in LONG format.
> #write.csv(cell_withitems, file = "cell_withitems_complete.csv")
```

The dataset has two dependent variables, accuracy in naming the email recipient, and the distance of the month estimate from the actual month in which the email was written. All other measures are independent variables.

3 Long to Wide to Long

Long to Wide

```
> library(tidyr)
> ## first, we pick the variables we are mainly interested in converting to wide format
> ## these are: item number, and accuracy for that item
>
> cell_long = cell_withitems[,c("ID", "ItemNo", "Accuracy")]
> cell_wide = cell_long %>%
+   spread(ItemNo, Accuracy)
```

Wide to Long

Next, we try to convert this WIDE dataset back to LONG format:

```
> library(dplyr)
> cell_long2 = cell_wide %>%
+   gather(-ID, key = "ItemNo", value = "Accuracy") %>%
+   arrange(ID)
> #cell_long2
```

Notice that there are some missing values in this dataset – this is because not all participants have the same number of items, and the spread() function creates all columns for all participants, even if they have missing values.

4 Creating the Wave and Date Variable

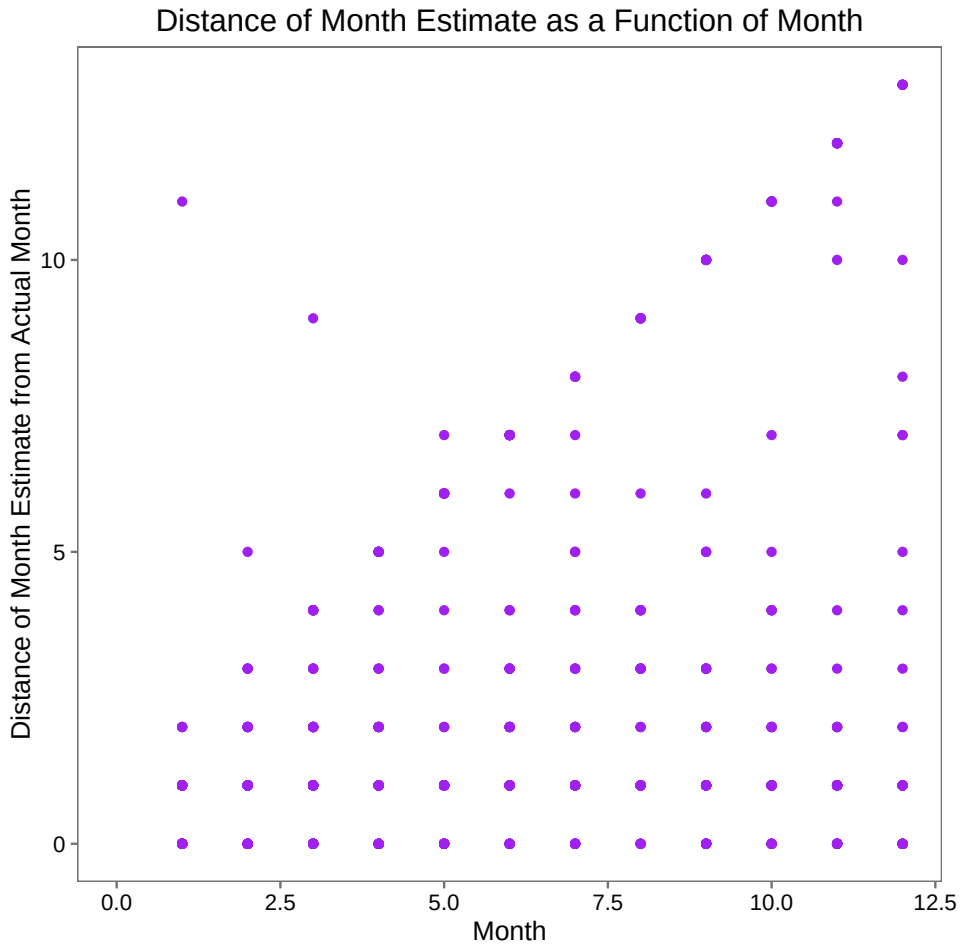
We will use the dataset cell$_{w}ithitemsforallfutureanalysis, asitcontainsalltheinformation.Wedonothavea"date"variabl$

5 Graphing the Data

In this section, we use different time metrics – days and months to graph the data. For graphing purposes, we will use the continuous dependent variable, TimeJudgmentDistance, since accuracy is dichotomous.

Collapsed across Subjects

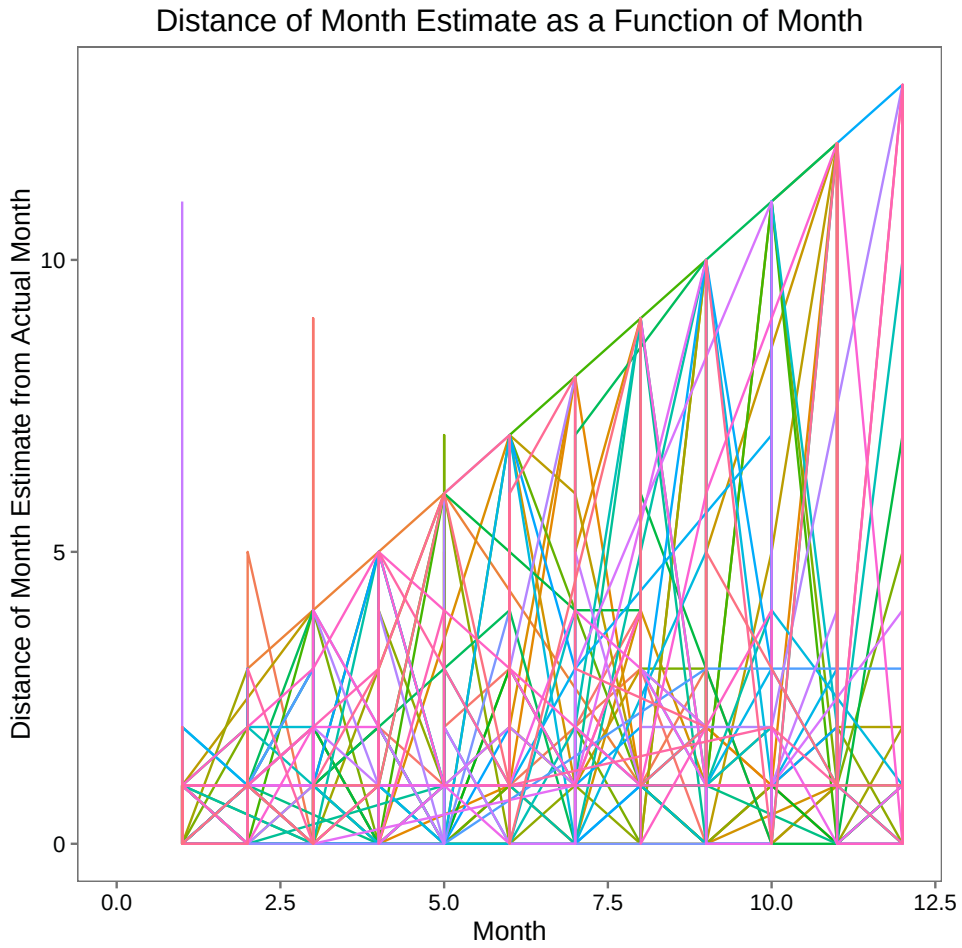
```
> library(ggplot2)
> library(ggthemes)
> g1 = ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance, group = ID)) +
+   geom_point(color = "purple") +
+   theme_few()+
+   xlim(0,12)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
> g1
```



At the person level

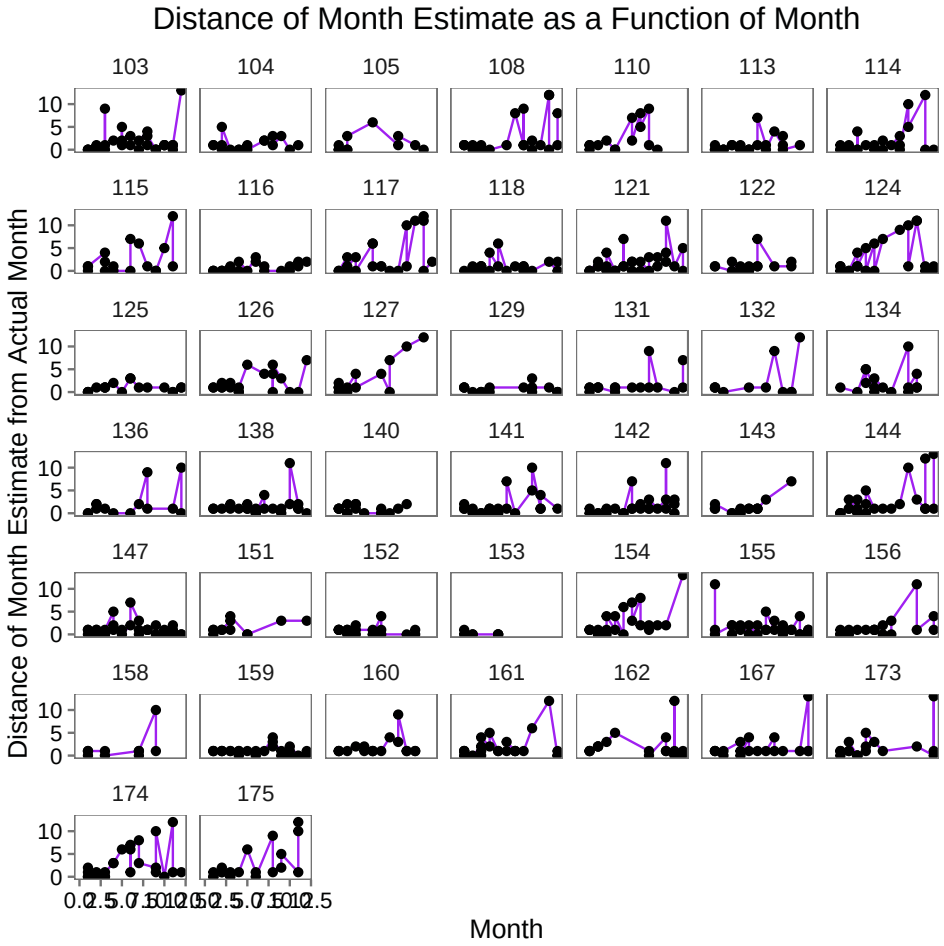
Without Faceting

```
> g2 <- ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance, group = ID)) +
+   geom_line() +
+   aes(color = factor(ID)) + guides(color = FALSE)+
+   theme_few()+
+   xlim(0,12)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
> g2
```



With Faceting

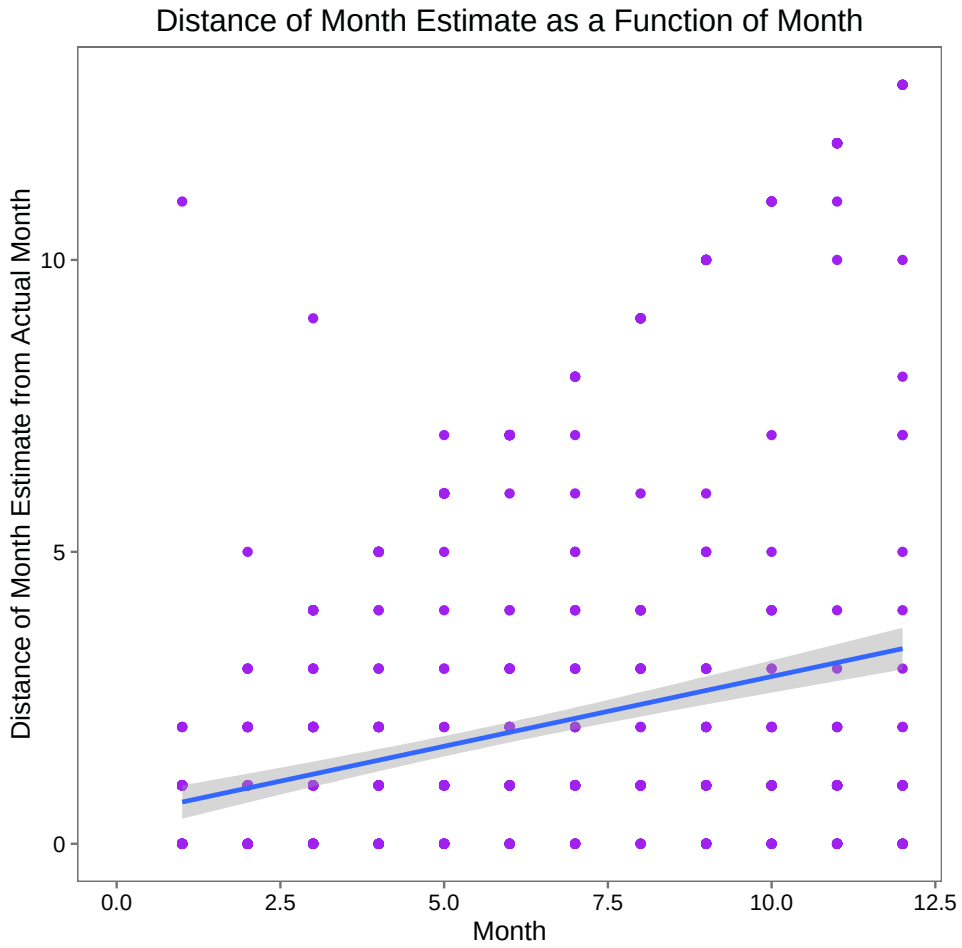
```
> g3 = ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance, group = ID)) +
+   geom_line(color = "purple") +
+   geom_point()+
+   theme_few()+
+   xlim(0,12)+
+   facet_wrap(~ID)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
> g3
```



6 Overall Trend Lines

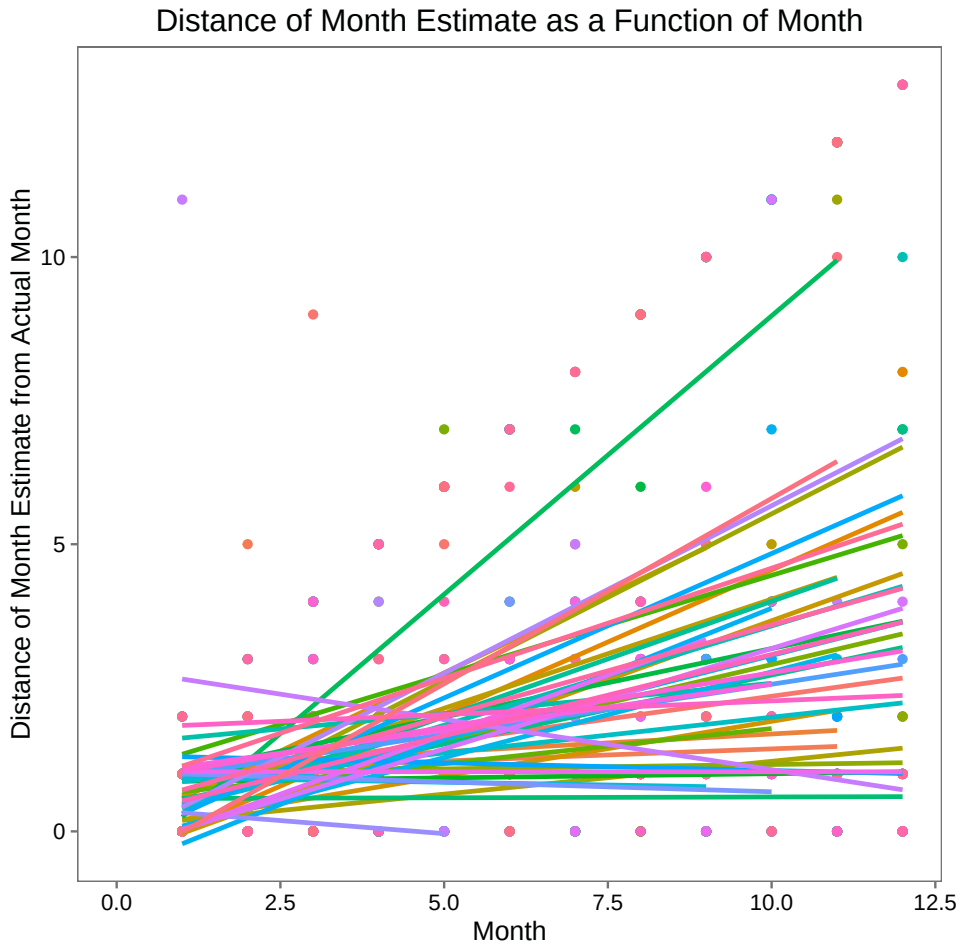
First, we look at the average trend line across subjects:

```
> g4 = ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance)) +
+   geom_point(color = "purple") +
+     stat_smooth(method = "lm")+
+   theme_few()+
+   xlim(0,12)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
> g4
```

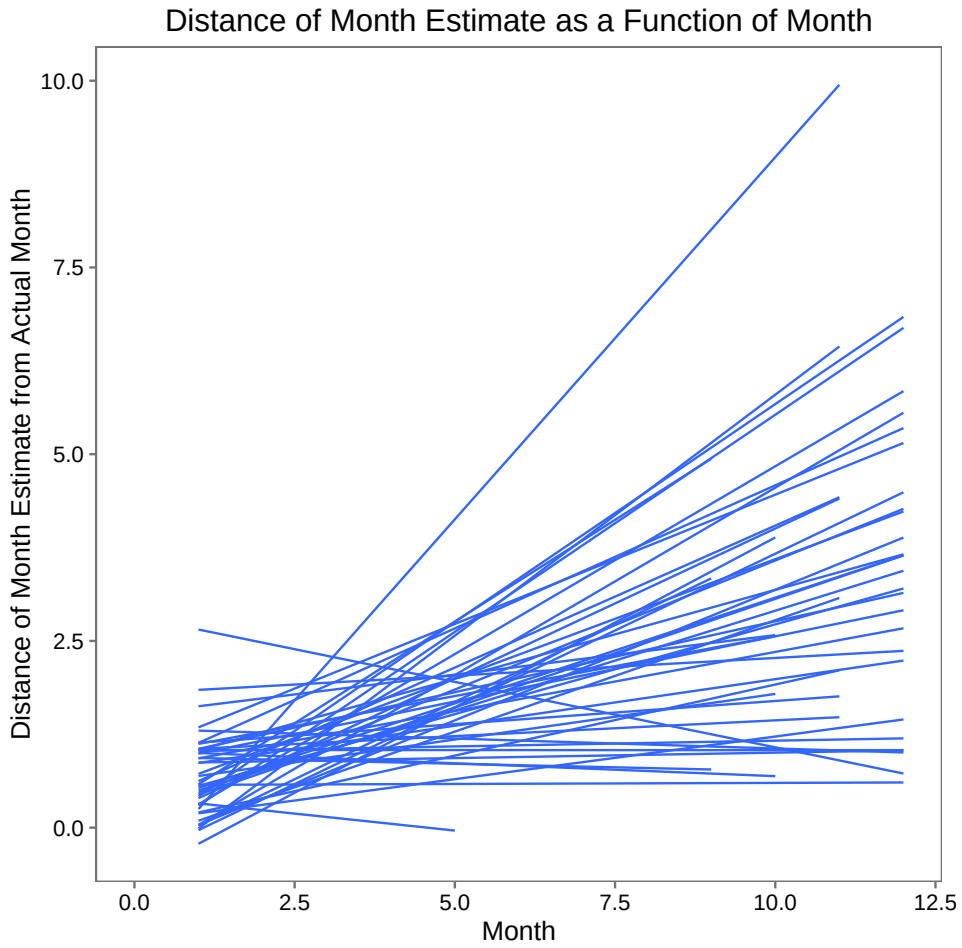


Then, we create an a trendline for each subject, in the same plot:

```
> g5 = ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance, group = ID)) +
+   geom_point() +
+   stat_smooth(method = "lm", se = FALSE)+
+   aes(color = factor(ID)) + guides(color = FALSE)+
+   theme_few()+
+   xlim(0,12)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
> g5
```

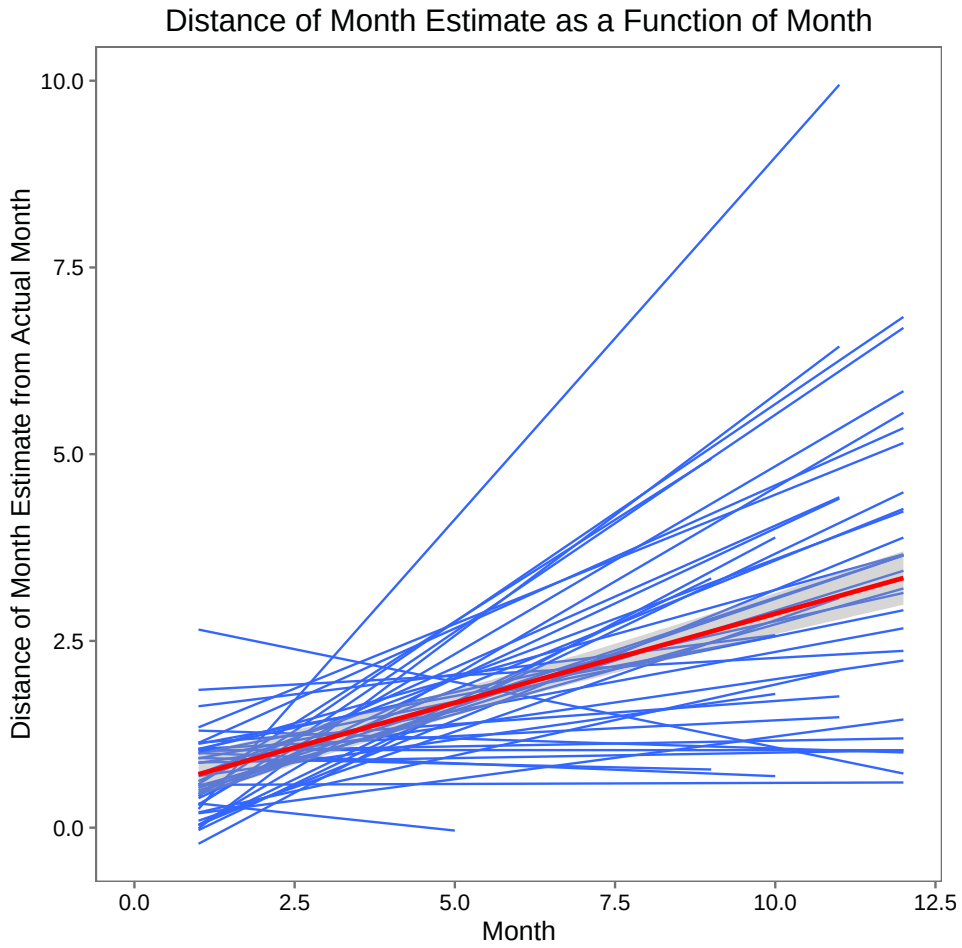



```
> #without colors and individual points:
>
> g6 = ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance, group = ID)) +
+   stat_smooth(method = "lm", se = FALSE, size =0.5)+
+   theme_few()+
+   xlim(0,12)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
> g6
```



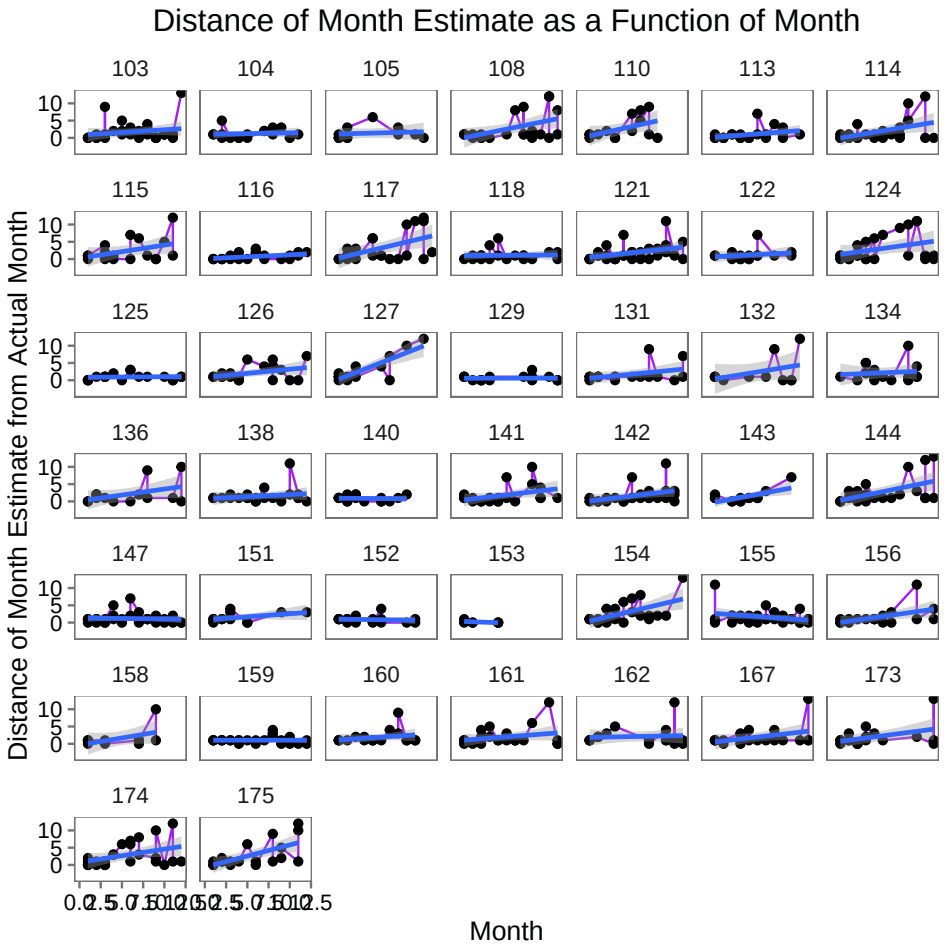
What if we also want the overall average trendline in the same plot?

```
> g7 <- g6 +
+   stat_summary(aes(group=1), fun.y = mean)+
+   stat_smooth(aes(group =1), method = "lm", color = "red")
> g7
```



And finally, if we want to look at the average trend line for each subject, along with its datapoints, in separate plots, we can use facetwrap():

```
> ggplot(cell_withitems, aes(x = Month, y = TimeJudgmentDistance, group = ID)) +
+   geom_line(color = "purple") +
+   geom_point()+
+   stat_smooth(method = "lm")+
+   theme_few()+
+   xlim(0,12)+
+   facet_wrap(~ID)+
+   xlab("Month") + ylab("Distance of Month Estimate from Actual Month") +
+   ggtitle("Distance of Month Estimate as a Function of Month")
```



Alternatively, We may also want to highlight only some of the participants:

```
> library(dplyr)
> library(MASS)
> set.seed(11)
> #cell_withitems$ID = as.factor(as.integer(cell_withitems$ID))
> #random_sample <- cell_withitems %>%
> #  select(ID) %>%
>  # distinct %>%
>   #sample_n(10)
>
> #sample_10 = left_join(random_sample, cell_withitems)
```