Original KC with If Error Types

tr_2m_mirrorLog_convS.txt

```r
library(lme4)
```

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

```r
library(dplyr)
```

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

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

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

```r
library(ggplot2)
library(knitr)
library(reshape2)
library(reshape)
```

```
##
## Attaching package: 'reshape'

## The following objects are masked from 'package:reshape2':
##
##     colsplit, melt, recast

## The following object is masked from 'package:dplyr':
##
##     rename

## The following object is masked from 'package:Matrix':
##
##     expand
```

```r
library(RColorBrewer)
library(ggpubr)
```

```
## Loading required package: magrittr
```

```r
# remove warning
opts_chunk$set(message=FALSE, warning=FALSE)
```

Transaction to Rollup

- Load transaction

```r
# load a transaction file annotated with the split errors
read_transaction_file <- function(path){
  if(substr(path,nchar(path)-2, nchar(path)) == "txt"){
    df <- read.csv(path, sep = "\t")
  }else if (substr(path,nchar(path)-2, nchar(path)) == "csv"){

    df <- read.csv(path)
  }
  dataframe <- data.frame(df, na.string = "", stringsAsFactors = FALSE)
  return(dataframe)
}
```

- Add KC model according to model definition

```
#Use this function if using the new KC model

#df: transaction file,
#new_kc: dataframe which defines better model
#old_kc_model_name: column name of old kc

add_kc_model <- function(df, new_kc, old_kc_model_name){
  new_df <- left_join(df, new_kc, by = old_kc_model_name)
  return(new_df)
}
```

- Concatenate four binary columns to binary string

```
concatenate <- function(df){
  # rows where correct
  correct_row <- which((df[, "S_current"] == 1) & (df[, "I_current"] == 1))

  # might redefine correct ?? now treat every correct as "1100"
  df[correct_row, ]$S_downstream <- 0
  df[correct_row, ]$I_downstream <- 0

  # make new column of binary stirng
  df$binary_string <- paste(as.character(df$S_current), as.character(df$I_current),
                            as.character(df$S_downstream), as.character(df$I_downstream),sep = "")

 # print(unique(df$binary_string))
  return(df)
}
```

- Add error type from binary string according to error definition

```
add_error_types <- function(df, error_definition){

  df <- concatenate(df)
  df <- left_join(df, error_definition, by = "binary_string")

  return(df)
}
```

- Add hint and correct error to error types

```
#After adding the error types defined in error_definition, add in the hint error type and
# also correct ones

add_hint_correct <- function(df, error_definition){
  colname <- c(colnames(error_definition)[2:3])

#  df[, colname] <- as.character(df[, colname])
  hint_row <- which(df[, "Outcome"] == "HINT")
  df[hint_row, colname[1]] <- "Hint"
  df[hint_row, colname[2]] <- "Hint"

  correct_row <- which(df[, "Outcome"] == "CORRECT")
  df[correct_row, colname[1]] <- "correct"
  df[correct_row, colname[2]] <- "correct"
  return(df)
}
```

- Make step slices

```
# For every student, for each step,
# Make step slices, each step slice distinguished by sumCorrect
make_step_slices <- function(df){


  df <- df %>%
   # select (-drop_col) %>%
    group_by(Anon.Student.Id, Problem.Name, Step.Name) %>%
    mutate(correct = ifelse(Outcome == "CORRECT", 1, 0),
                     sumCorrect = cumsum(correct))

  df[df$correct == 0, ]$sumCorrect <- df[df$correct == 0, ]$sumCorrect + 1
  return(df)

}
```

- Rollup

```
transaction_to_rollup <- function(df, model_definition, model, error_definition, error, level){
  stopifnot(level %in% c("all", "Interleaved", "Blocked"))
  stopifnot(model %in% c("orig", "new"))
  stopifnot(error %in% c("orig", "new"))
  if(model == "orig"){
    m <- colnames(model_definition)[1]
  }else if(model == "new"){
    m <- colnames(model_definition)[2]
  }

  if(error == "orig"){
    e <- colnames(error_definition)[2]
  }else if(error == "new"){
    e <- colnames(error_definition)[3]
  }

  if(level != "all"){
    if(level == "Interleaved"){
      df <- df %>% filter(Level..ProblemSet. %in% c("Fraction Arithmetic Interleaved 1",
                                                    "Fraction Arithmetic Interleaved 2"))
      print(paste("Level of ProblemSet includes", unique(df$Level..ProblemSet.), sep = " "))

    }else if(level == "Blocked"){
      df <- df %>% filter(Level..ProblemSet. %in% c("Fraction Arithmetic Blocked 1",
                                                    "Fraction Arithmetic Blocked 2"))
      print(paste("Level of ProblemSet includes", unique(df$Level..ProblemSet.), sep = " "))

    }
  }





  df <- df %>% group_by(Anon.Student.Id, Problem.Name, Step.Name, sumCorrect) %>%
               mutate(KC_name =  tail(!!as.name(m), n = 1),  # KC_name is the KC student is working tow
               first_error = (!!as.name(e))[1],         # first error in the step slice
               hint_used = ifelse("HINT" %in% Outcome, "hint", ""), # Whether has used hint in that ste
               Total = ifelse( (Outcome[1] == "INCORRECT" | Outcome[1] == "HINT"), 1, 0) # general erro
               ) %>%

       distinct(df, .keep_all=TRUE) # Keep only first row for each step slice, so #row = #step slice

  return(df)
}
```

- Add indicator columns indicating which first error according to error definition

```
add_first_error <- function(roll){
  unique_type <- unique(roll$first_error)
  unique_type <- unique_type[!is.na(unique_type)]
  for (type in unique_type){
    roll[, type] <- ifelse(roll$first_error == type, 1, 0)
  }
  return(roll)
}
```

- Add opportunity count

```
# Count opp for each student:
# For each student, for each KC being worked toward, count number of step slice(rows)

add_opportunity <- function(roll){

  roll <- roll %>% group_by(Anon.Student.Id, KC_name) %>%
  mutate(opp = seq.int(n())) #  %>% arrange(by_group = KC_name)

  roll <- roll[roll$KC_name != "", ]
  roll$opp <- as.factor(roll$opp)
  return(roll)
}
```

- Aggregate errors

```
# Aggregated across KC
aggregate_all <- function(roll, exclude_KC = ""){
  unique_type <- unique(roll$first_error)
  unique_type <- unique_type[!is.na(unique_type)]
  # Filter out some KC
```

```r
  agg <- roll[!(roll$KC_name %in% exclude_KC), ]


  col <- c("opp", "Total")
  for (type in unique_type){
   col <- c(col, type)
  }
  agg <- agg[, (colnames(agg) %in% col)]


  # Make 3 aggregated tables
  agg1 <- agg %>% group_by(opp) %>% summarise(n = n())
  agg2 <- agg %>% group_by(opp) %>% summarise_all(.funs = mean, na.rm = T)
  agg3 <- agg %>% group_by(opp) %>% summarise_all(.fun = sum, na.rm = T)
  return(list(agg1, agg2, agg3))

}



# By individual KC
aggregate_kc <- function(roll, exclude_KC = ""){
  unique_type <- unique(roll$first_error)
  unique_type <- unique_type[!is.na(unique_type)]
  # Filter out some KC

  agg <- roll[!(roll$KC_name %in% exclude_KC), ]

  col <- c("opp", "KC_name", "Total")
  for (type in unique_type){
   col <- c(col, type)
  }
  agg <- agg[, (colnames(agg) %in% col)]




  # Make 3 aggregated tables
  agg1 <- agg %>% group_by(KC_name, opp) %>% summarise(n = n())
  agg2 <- agg %>% group_by(KC_name, opp) %>% summarise_all(.funs = mean, na.rm = T)
  agg3 <- agg %>% group_by(KC_name, opp) %>% summarise_all(.fun = sum, na.rm = T)

  kc_error <- agg2 %>% group_by(KC_name) %>% summarize_all(mean)


  # Another Table of Total Error by Opp given KC
  kc_by_opp <- agg2
  kc_by_opp <- dcast(kc_by_opp, opp ~ KC_name, value.var = "Total")

  return(list(agg1, agg2, agg3, kc_by_opp, kc_error))

}
```

Plotting

- Overall plot

```
# Pick a Palette
my_colors <- c("#E69F00", "#56B4E9", "#D55E00", "#0072B2","#CC79A7","#1B9E77")
```

```
plot_all <- function(res, exclude_error = "None", y_range, main, size,legendpos='right'){

  agg <- res[[2]]
  agg$opp <- as.numeric(agg$opp)
  agg <- data.frame(agg)
  err_type <- (colnames(agg))[-1]

  if (exclude_error != "None"){
    err_type <- err_type[!(err_type %in% exclude_error)]
  }
  #print(err_type)
  agg <- melt(agg, id = "opp", measure = err_type) %>% filter(variable != "correct")



  # reorder
  agg$variable <- as.factor(agg$variable)
  lev <- sort(levels(agg$variable))

  if('Total' %in% lev) {
    lev <- c(lev[-match('Total',lev)],'Total')
  }

  agg$variable <- factor(agg$variable, levels = lev, ordered = T)
  # plot after reordering

  width <- 4
  ggplot(agg, aes(opp, value, colour = variable)) + geom_line(size = size) + ylim(y_range) + xlim(0,25)
  scale_colour_manual(values = my_colors) +
  theme(text=element_text(family="Helvetica", face="bold", size=12),legend.position =legendpos)
  #+ scale_color_brewer(palette = "Set1")
}

plot_legend <- function(res, exclude_error = "None", y_range, main, size,legendpos='right'){

  agg <- res[[2]]
  agg$opp <- as.numeric(agg$opp)
  agg <- data.frame(agg)
  err_type <- (colnames(agg))[-1]

  if (exclude_error != "None"){
    err_type <- err_type[!(err_type %in% exclude_error)]
  }
  #print(err_type)
  agg <- melt(agg, id = "opp", measure = err_type) %>% filter(variable != "correct")
  # reorder
  agg$variable <- as.factor(agg$variable)
  lev <- sort(levels(agg$variable))

  if('Total' %in% lev) {
    lev <- c(lev[-match('Total',lev)],'Total')
  }

  agg$variable <- factor(agg$variable, levels = lev, ordered = T)
  # plot after reordering

  ggplot(agg, aes(value, colour = variable)) + geom_bar()
  #+ scale_color_brewer(palette = "Set1")
}




plot_kc <- function(res, exclude_error = "None", y_range, main, size){

  agg <- res[[2]]
  agg$opp <- as.numeric(agg$opp)
  agg <- data.frame(agg)
  err_type <- (colnames(agg))[-c(1,2)]
#  print("plot kc, err_type:")
#  print(err_type)



  if (exclude_error != "None"){
    err_type <- err_type[!(err_type %in% exclude_error)]
  }
  agg <- melt(agg, id = c("opp", "KC_name"), measure = err_type) %>% filter((variable != "correct") & (


  # reorder
  agg$variable <- as.factor(agg$variable)
  lev <- sort(levels(agg$variable))

  if('Total' %in% lev) {
    lev <- c(lev[-match('Total',lev)],'Total')
  }

  agg$variable <- factor(agg$variable, levels = lev, ordered = T)
  # plot after reordering



  ggplot(agg, aes(opp, value, colour = variable)) + geom_line(size = size) + ylim(y_range) + xlim(0,25)
  scale_colour_manual(values = my_colors)# + scale_color_brewer(palette = "Set1")
}
```

- Residual plot

```r
residual_plot <- function(tables1, tables2, exclude_error = "None", h_line = FALSE, y_range = c(-0.4, 0
  t1 <- tables1[[2]]
  t2 <- tables2[[2]]
  # First col is opp,rest are errors
  # Reorder cols
  order <- colnames(t1)
  t2 <- t2[ ,order]

  # Truncate
  nrows <- min(nrow(t1), nrow(t2))
  longer <- which.max(c(nrow(t1), nrow(t2)))
  diff <- abs(nrow(t1) - nrow(t2))


  print(paste("Truncating", as.character(diff), "rows from table", as.character(longer), "...", sep = "
  t1 <- t1[c(1:nrows),]
  t2 <- t2[c(1:nrows),]

  residual <- t1 - t2

  residual$opp <- 1:nrow(residual)
  residual <- data.frame(residual)
  err_type <- (colnames(residual))[-1]

  if (exclude_error != "None"){
    err_type <- err_type[!(err_type %in% exclude_error)]
  }

  residual <- melt(residual, id = "opp", measure = err_type) %>% filter(variable != "correct")

  # reorder
  residual$variable <- as.factor(residual$variable)
  lev <- sort(levels(residual$variable))

  if('Total' %in% lev) {
    lev <- c(lev[-match('Total',lev)],'Total')
  }

  residual$variable <- factor(residual$variable, levels = lev, ordered = T)
  # plot after reordering



  p <- ggplot(residual, aes(opp, value, colour = variable)) +
  geom_line(size = size) + labs(title = main, x = "Opportunity", y = "Difference (AL - Human)", color =
  scale_colour_manual(values = my_colors) +
  theme(text=element_text(family="Helvetica", face="bold", size=12))+
  ylim(y_range) + xlim(0,25)

  if(h_line == TRUE){
    p <- p + geom_hline(aes(yintercept = 0), linetype = "dashed")
  }
  p
}
```

Wrapper functions

```
rollup_from_transaction <- function(transaction, error_definition, model_definition, old_model_name, mod
  path <- transaction

  # clean transaction
  trans <- read_transaction_file(path)
  trans <- add_kc_model(trans, model_definition, old_model_name)
  trans <- add_error_types(trans, error_definition)
  trans <- add_hint_correct(trans, error_definition)
  trans <- make_step_slices(trans)

  rollup <- transaction_to_rollup(trans, model_definition, model = model_to_use, error_definition, error
  rollup <- add_first_error(rollup)
  rollup <- add_opportunity(rollup)

  return(rollup)
}


# use this to make residual, returns a list of 3 tables
aggregate_from_rollup <- function(rollup, KC_to_remove = "", mode = "aggregated"){
  if(mode == "aggregated"){
    agg_res <- aggregate_all(rollup, exclude_KC = KC_to_remove)
  }else if(mode == "KC"){
    agg_res <- aggregate_kc(rollup, exclude_KC = KC_to_remove)
  }
  return(agg_res)
}


plot_from_aggregate <- function(table, mode = "aggregated", error_filter = "None", y_range = c(0, 0.5),
  if(mode == "aggregated"){
    plot <- plot_all(table, exclude_error = error_filter, y_range = y_range, main = main, size = size,
  }else if(mode == "KC"){
    plot <- plot_kc(table, exclude_error = error_filter, y_range = y_range, main = main, size = size)
  }else if(mode == "legend"){
    plot <- plot_legend(table, exclude_error = error_filter, y_range = y_range, main = main, size = size
  }
  return(plot)
}
```

Define better KC model

```
#orig_kc <- c("M den5", "M num5", "M done", "AS den5", "AS num5",
#              "AS check_convert", "AS done", "", "AD num3","AD check_convert",
#              "AD den3", "AD done", "AD den4", "AD num4", "AD num5", "AD den5",
#              "M check_convert", "AD operation2", "M blankProblem")
```

```
orig_kc <- c("M den5", "M num5", "M done", "AS den5", "AS num5",
              "AS done", "AD num3",
             "AD den3", "AD done", "AD den4", "AD num4", "AD num5", "AD den5")


better_kc <-  c("M den_ans", "M num_ans", "done", "A den_ans",
                "A num_ans", "check_convert", "done", "", "AD num_conv",
                "check_convert", "AD den_conv", "done", "AD den_conv",
                "AD num_conv", "A num_ans", "A den_ans", "check_convert",
                "AD operation2", "M blankProblem")

model_defined <- data.frame(matrix(NA, nrow = length(orig_kc), ncol = 2))
names(model_defined) <- c("KC..Field.", "KC_combined")
model_defined$KC..Field. <- orig_kc
#model_defined$KC_combined <- better_kc


kable(model_defined)
```

KC..Field.	KC_combined
M den5	NA
M num5	NA
M done	NA
AS den5	NA
AS num5	NA
AS done	NA
AD num3	NA
AD den3	NA
AD done	NA
AD den4	NA
AD num4	NA
AD num5	NA
AD den5	NA

```
#(model_defined
```

Define error types from binary string

```
#Binary strings that appear in the transaction file:
#"1100", "1000", "0000", "0011", "1001", "0110", "0010", "0111", "NA"
#where "NA" means that transaction is either tutor performed, or is a hint request

binary_str <- c("1100", "1000", "1001", "0000", "0010", "0110", "0011", "0111")
type_defined <- data.frame(matrix(NA, nrow = length(binary_str), ncol = 3))
orig_type <- c("correct", "incorrect", "misapplied", "out_of_graph", "wild", "where", "when", "where")

simplified_type <- c("correct", "Then_Error",  "Then_Error",  "If_Error", "Then_Error", "If_Error", "If_

names(type_defined) <- c("binary_string", "original_error_type", "simplified_error_type")
type_defined$binary_string <- binary_str
type_defined$original_error_type <- orig_type
```

```
type_defined$simplified_error_type <- simplified_type


kable(type_defined)
```

binary_string	original_error_type	simplified_error_type
1100	correct	correct
1000	incorrect	Then_Error
1001	misapplied	Then_Error
0000	out_of_graph	If_Error
0010	wild	Then_Error
0110	where	If_Error
0011	when	If_Error
0111	where	If_Error

Load human data

```
# For human, Level could be "all", "Interleaved", or "Blocked"
human <- rollup_from_transaction(transaction = "../human_convS.txt",
                        error_definition = type_defined,
                        model_definition = model_defined,
                        old_model_name = "KC..Field.",
                        model_to_use = "orig", error_to_use = "new", level = "all")
#print(human[ ,c('Selection','opp','')])
```

```
done_kcs <- c("M done", "AS done", "AD done")
convert_kcs <- c('AD check_convert', 'AS check_convert','M check_convert')
bad_kcs <- c("AS den3", "AS den4", "AS num3", "AS num4","M den3", "M den4", "M num3", "M num4", "AD ope

curve_human <- aggregate_from_rollup(rollup = human, KC_to_remove = c(convert_kcs, bad_kcs, done_kcs), 
#agg_all_with_done_human <- aggregate_from_rollup(rollup = human, KC_to_remove = c(convert_kcs, bad_kcs

#agg_kc_without_done_human <- aggregate_from_rollup(rollup = human, KC_to_remove = c(convert_kcs, bad_k
#agg_kc_with_done_human <- aggregate_from_rollup(rollup = human, KC_to_remove = c(convert_kcs, bad_kcs)
```

Load AL data

```
dt <- rollup_from_transaction(transaction = "../mirror/dt_2m_mirrorLog_convS.txt",
                        error_definition = type_defined,
                        model_definition = model_defined,
                        old_model_name = "KC..Field.",
                        model_to_use = "orig", error_to_use = "new")
tr <- rollup_from_transaction(transaction = "../mirror/tr_2m_mirrorLog_convS.txt",
                        error_definition = type_defined,
                        model_definition = model_defined,
                        old_model_name = "KC..Field.",
                        model_to_use = "orig", error_to_use = "new")
dt_in <- rollup_from_transaction(transaction = "../mirror/dt_in_2m_mirrorLog_convS.txt",
                        error_definition = type_defined,
                        model_definition = model_defined,
                        old_model_name = "KC..Field.",
                        model_to_use = "orig", error_to_use = "new")
```

```
done_kcs <- c("M done", "AS done", "AD done")
convert_kcs <- c('AD check_convert', 'AS check_convert','M check_convert')
bad_kcs <- c("AS den3", "AS den4", "AS num3", "AS num4","M den3", "M den4", "M num3", "M num4", "AD ope

curve_dt <- aggregate_from_rollup(rollup = dt, KC_to_remove = c(convert_kcs, bad_kcs, done_kcs), mode =
curve_tr <- aggregate_from_rollup(rollup = tr, KC_to_remove = c(convert_kcs, bad_kcs, done_kcs), mode =
curve_dt_in <- aggregate_from_rollup(rollup = dt_in, KC_to_remove = c(convert_kcs, bad_kcs, done_kcs),
```

CORE PLOTS

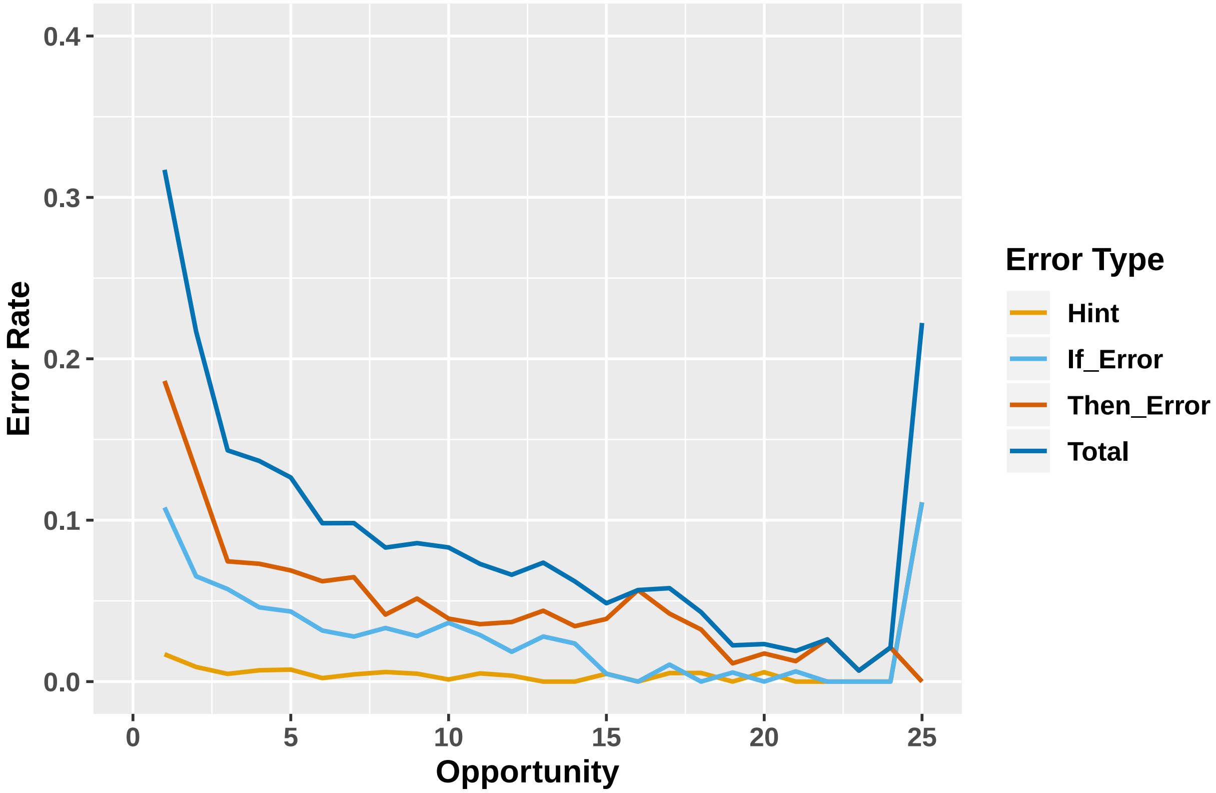
```
# Overall Total error without done KC
#library(tidyverse)
#library(egg)
plot_em <- function(al_curve, human_curve,name='dt'){
  cr <- residual_plot(al_curve, human_curve, h_line = TRUE, y_range = c(-0.16, 0.16), main = "", size =
  ca <- plot_from_aggregate(al_curve, mode = "aggregated", y_range = c(0, 0.4), main = "", size = 0.8,
  ch <- plot_from_aggregate(human_curve, mode = "aggregated", y_range = c(0, 0.4), main = "", size = 0.


  print(ch)
  #print()

  car <- ggarrange(ca,cr,nrow=1,widths=c(8,8),heights=c(1))
  #print(ca)
  print(car)
 # ggsave("~/Pictures/AIED2020/Human.eps",plot=ch,device='eps')
 # ggsave(paste("~/Pictures/AIED2020/", name, "_curve.eps",sep=''),plot=car,device='eps',height=3.5, wi
  #ggsave(paste("~/Pictures/AIED2020/", name, "_res.eps",sep=''),plot=cr,device='eps')
  return(list(ca,cr))
}
```

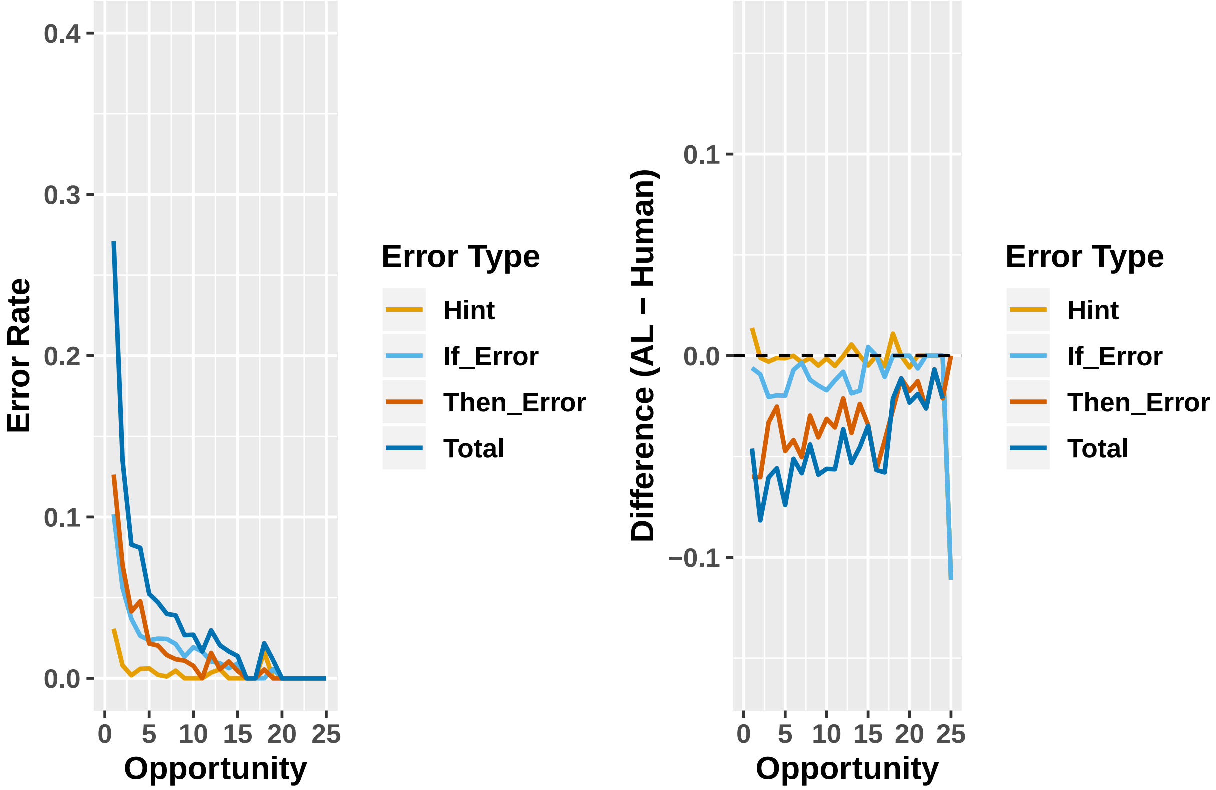
```
print("DECISION TREE")
```

```
## [1] "DECISION TREE"
```

```
dt_curves <- plot_em(curve_dt,curve_human,'dt')
```

```
## [1] "Truncating 0 rows from table 1 ..."
```



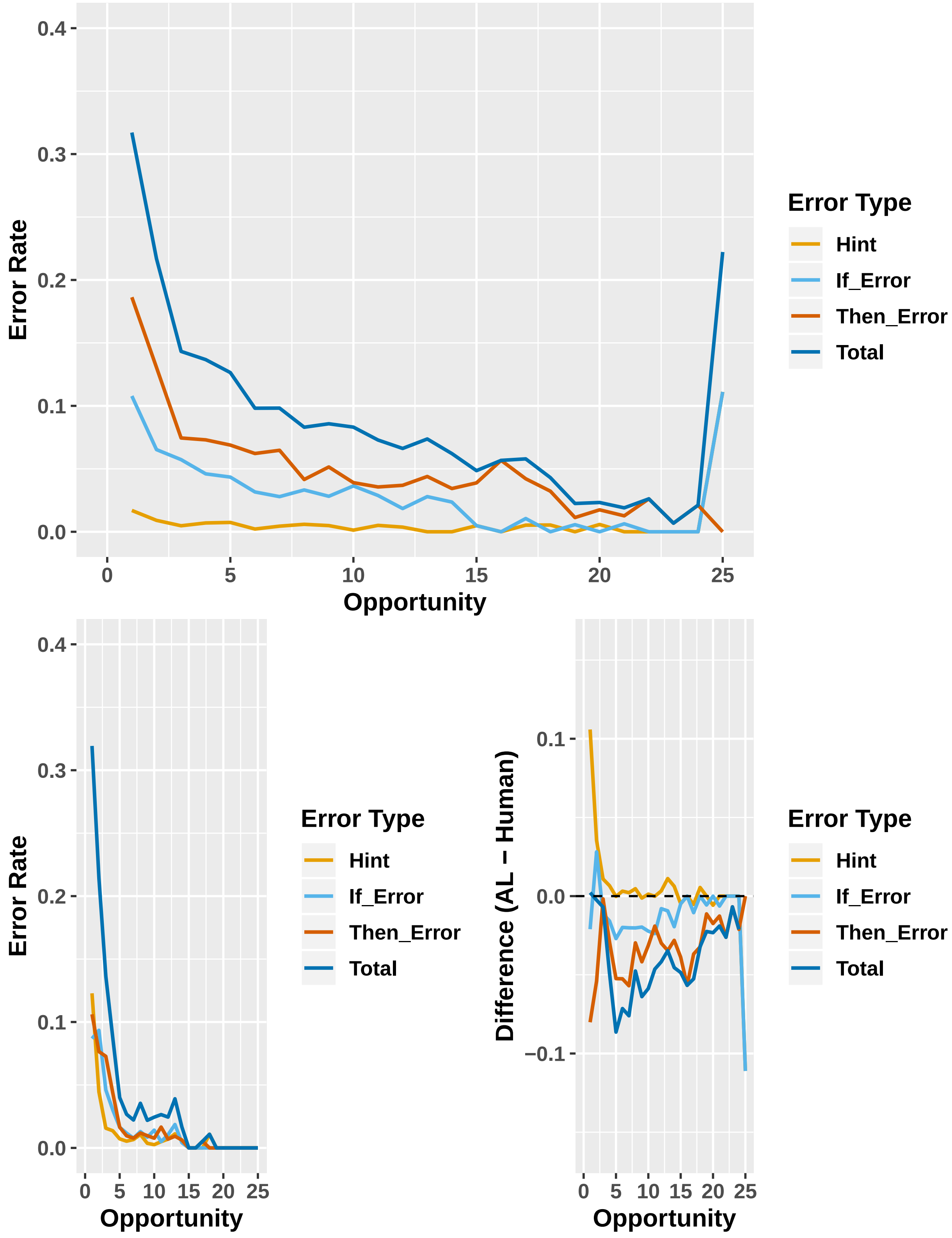


```
print("TRESTLE")
```

```
## [1] "TRESTLE"
```

```
tr_curves <- plot_em(curve_tr,curve_human,'tr')
```

```
## [1] "Truncating 0 rows from table 1 ..."
```

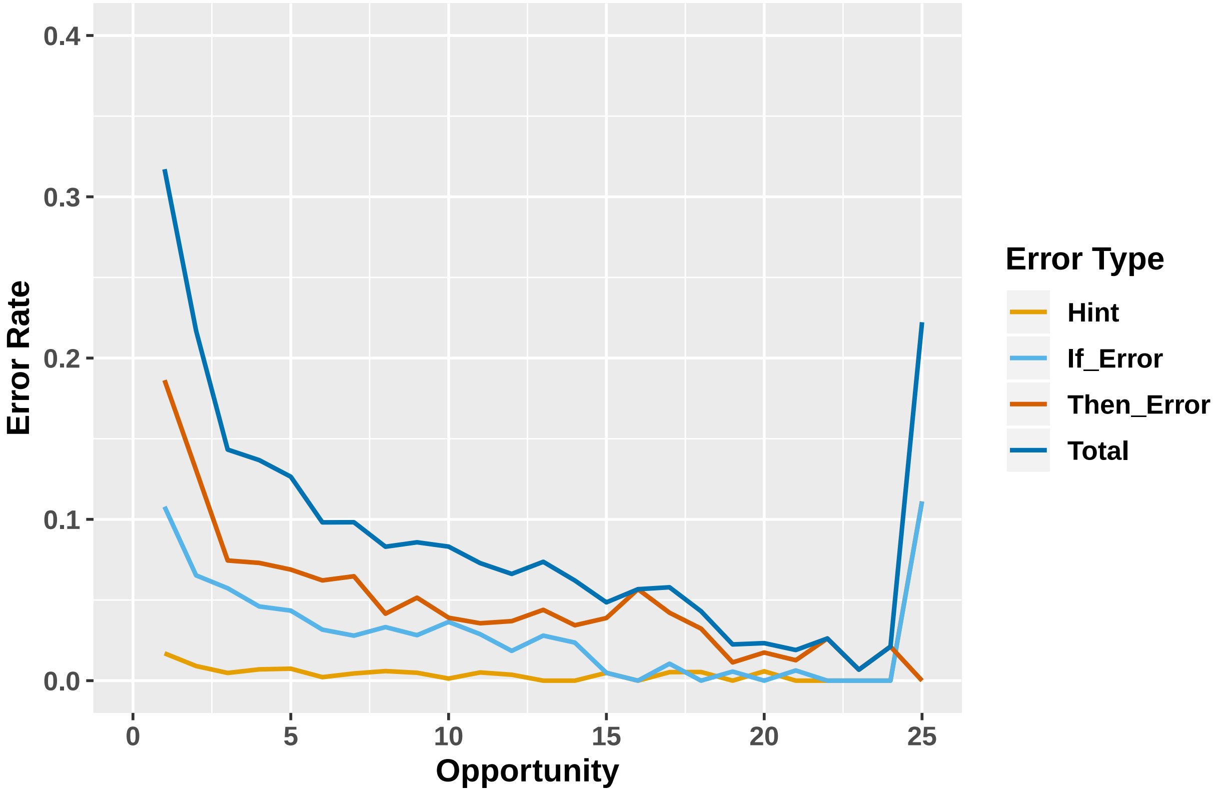


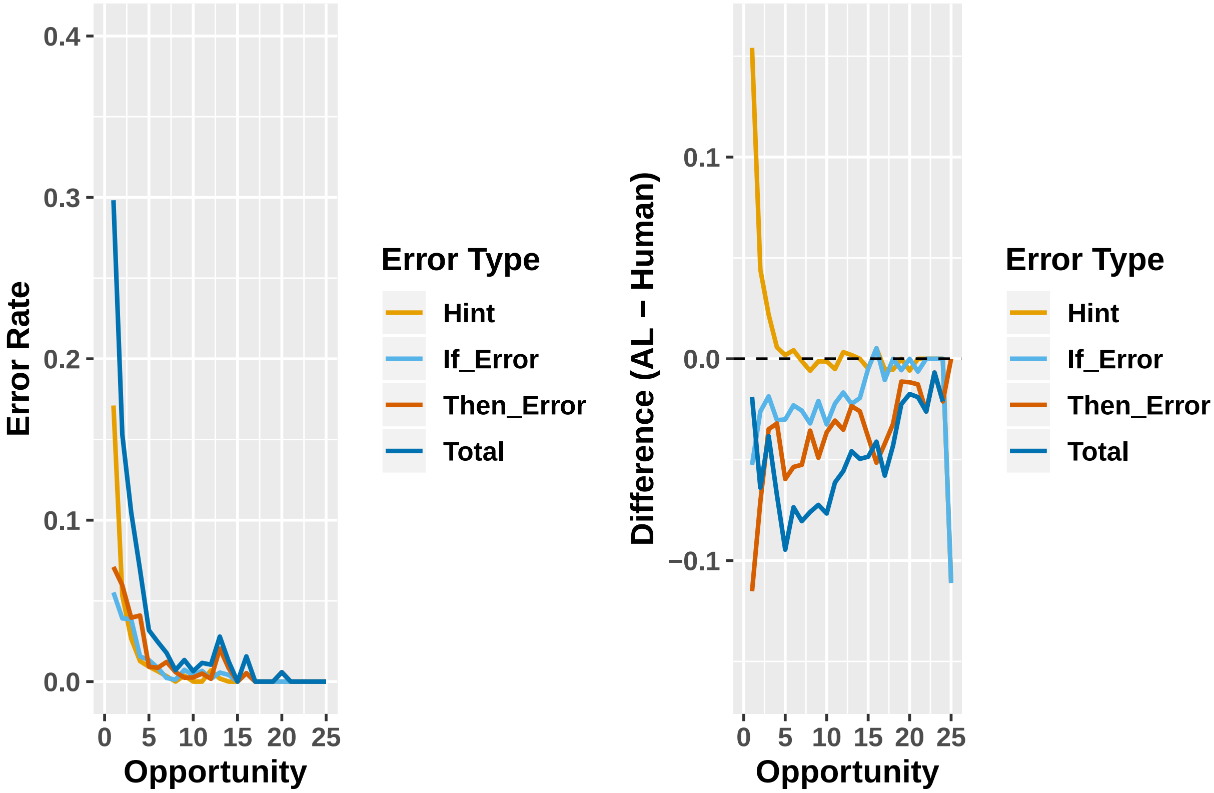
```r
print("IMPLICIT NEGATIVE")
```

```
## [1] "IMPLICIT NEGATIVE"
```

```r
dt_in_curves <- plot_em(curve_dt_in,curve_human,'dt_in')
```

```
## [1] "Truncating 0 rows from table 1 ..."
```

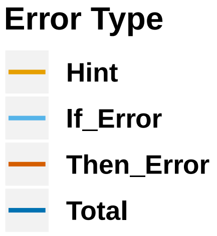




```r
g_legend <- function(a.gplot){
    tmp <- ggplot_gtable(ggplot_build(a.gplot))
    leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
    legend <- tmp$grobs[[leg]]
    legend
}
#print(g_legend(ch))
```

```r
#ggarrange(ch,
library("cowplot")
human_curve <- curve_human

ch <- plot_from_aggregate(human_curve, mode = "aggregated", y_range = c(0, 0.4), main = "", size = 0.8)
all_plots <- ggarrange(  ch + theme(axis.ticks.x = element_blank(),
                                    axis.title.x = element_blank(),
                                    axis.text.x = element_blank(),
                                    legend.position = 'none',
                                    plot.margin = margin(r = 2) ),
                    ggdraw(get_legend(ch + theme(
                            legend.title = element_text( size = 26),
                            legend.text = element_text( size = 20)
                            ))),

                    dt_curves[[1]] + theme(axis.ticks.x = element_blank(),
                                    axis.title.x = element_blank(),
                                    axis.text.x = element_blank(),
                                    legend.position = 'none',
                                    plot.margin = margin(r = 2) ),
                    dt_curves[[2]] + theme(axis.ticks.x = element_blank(),
                                           axis.title.x = element_blank(),
                                           axis.text.x = element_blank(),
                                           legend.position = 'none',
                                           plot.margin = margin(l = 2) )+
                               scale_y_continuous(position ='right',limits=c(-.16,.16)),
                    tr_curves[[1]] + theme(axis.ticks.x = element_blank(),
                                           axis.title.x = element_blank(),
                                           axis.text.x = element_blank(),
                                           legend.position = 'none',
                                plot.margin = margin(r = 2) ) ,
                    tr_curves[[2]] + theme(axis.ticks.x = element_blank(),
                                           axis.title.x = element_blank(),
                                           axis.text.x = element_blank(),
                                           legend.position = 'none',
                                plot.margin = margin(l = 2) ) +
                                scale_y_continuous(position ='right',limits=c(-.16,.16)),
                    dt_in_curves[[1]] + theme(axis.ticks.y = element_blank(),
                                        legend.position = 'none',
                                plot.margin = margin(r = 2) ),
                    dt_in_curves[[2]] + theme(axis.ticks.y = element_blank(),
                                        legend.position = 'none',
                                plot.margin = margin(l = 2) )+
                                scale_y_continuous(position ='right',limits=c(-.16,.16)),
nrow =4,ncol=2)#, nrow=2)


#ggsave("~/Pictures/AIED2020/all_curves.eps",plot=all_plots,device='eps',height=15, width=10)
```

```
library(gridExtra)
ch <- plot_from_aggregate(curve_human, mode = "aggregated", y_range = c(0, 0.4), main = "", size = 0.8)

ggdraw(get_legend(ch))
```



```
#my_hist <- ggplot(diamonds, aes(clarity, fill = cut)) +
#    geom_bar()

#ggarrange(ch,my_plots,ncol=1,nrow=2)
```

```
then_if <- function(curves){
  #print(curves)
  n_if <- sum(curves[3][[1]]$If_Error)
  n_then <- sum(curves[3][[1]]$Then_Error)
  n_hint <- sum(curves[3][[1]]$Hint)

  then_if_ratio <- n_then/n_if
  print("THEN_IF")
  print(then_if_ratio)
}

hint_other <- function(curves){
  #print(curves)
  n_if <- sum(curves[3][[1]]$If_Error[1:5])
  n_then <- sum(curves[3][[1]]$Then_Error[1:5])
  n_hint <- sum(curves[3][[1]]$Hint[1:5])

  print("HINT_OTHER")
  print(n_hint / (n_then + n_if+n_hint))
}

then_if(curve_dt)
```

```
## [1] "THEN_IF"
## [1] 1.075325
```

```
then_if(curve_tr)
```

```
## [1] "THEN_IF"
## [1] 1.111111
```

```
then_if(curve_dt_in)
```

```
## [1] "THEN_IF"
## [1] 1.392344
```

```
then_if(curve_human)
```

```
## [1] "THEN_IF"
## [1] 1.728814
```

```
hint_other(curve_dt)
```

```
## [1] "HINT_OTHER"
## [1] 0.08721805
```

```
hint_other(curve_tr)
```

```
## [1] "HINT_OTHER"
## [1] 0.260274
```

```
hint_other(curve_dt_in)
```

```
## [1] "HINT_OTHER"
## [1] 0.4214876
```

```
hint_other(curve_human)
```

```
## [1] "HINT_OTHER"
## [1] 0.05015674
```

```
#print(curve_dt[3][[1]]$If_Error)

#print(curve_dt)
```

Split curves

```
curve_human
```

```
## [[1]]
## # A tibble: 25 x 2
##    opp       n
##    <fct> <int>
##  1 1      1129
##  2 2      1111
##  3 3      1054
##  4 4      1002
##  5 5       949
##  6 6       917
##  7 7       896
##  8 8       843
##  9 9       816
## 10 10      770
## # ... with 15 more rows
##
## [[2]]
## # A tibble: 25 x 6
##    opp   correct  Total Then_Error If_Error    Hint
##    <fct>   <dbl>  <dbl>      <dbl>    <dbl>   <dbl>
##  1 1       0.689 0.317      0.186    0.108  0.0169
##  2 2       0.795 0.217      0.131    0.0653 0.00907
##  3 3       0.863 0.143      0.0745   0.0573 0.00478
##  4 4       0.874 0.137      0.073    0.046  0.007
##  5 5       0.880 0.126      0.0689   0.0434 0.00742
##  6 6       0.904 0.0981     0.0622   0.0316 0.00218
##  7 7       0.903 0.0982     0.0647   0.0279 0.00446
##  8 8       0.919 0.0830     0.0415   0.0332 0.00593
##  9 9       0.915 0.0858     0.0515   0.0282 0.00490
## 10 10      0.923 0.0831     0.0390   0.0364 0.00130
## # ... with 15 more rows
##
## [[3]]
## # A tibble: 25 x 6
##    opp   correct Total Then_Error If_Error  Hint
##    <fct>   <dbl> <dbl>      <dbl>    <dbl> <dbl>
##  1 1         773   358        209      121    19
##  2 2         877   241        144       72    10
##  3 3         904   151         78       60     5
##  4 4         874   137         73       46     7
##  5 5         831   120         65       41     7
##  6 6         829    90         57       29     2
##  7 7         809    88         58       25     4
##  8 8         775    70         35       28     5
##  9 9         747    70         42       23     4
## 10 10        710    64         30       28     1
## # ... with 15 more rows
```

```
curve_dt
```

```
## [[1]]
## # A tibble: 25 x 2
##    opp       n
##    <fct> <int>
##  1 1      1140
##  2 2      1124
##  3 3      1086
##  4 4      1026
##  5 5       974
##  6 6       936
##  7 7       902
##  8 8       846
##  9 9       822
## 10 10      778
## # ... with 15 more rows
##
## [[2]]
## # A tibble: 25 x 6
##    opp   correct  Total If_Error Then_Error    Hint
##    <fct>   <dbl>  <dbl>    <dbl>      <dbl>   <dbl>
##  1 1       0.741 0.271    0.102     0.126   0.0307
##  2 2       0.866 0.135    0.0560    0.0703  0.00801
##  3 3       0.920 0.0829   0.0368    0.0414  0.00184
##  4 4       0.920 0.0809   0.0263    0.0478  0.00585
##  5 5       0.949 0.0524   0.0236    0.0216  0.00616
##  6 6       0.953 0.0470   0.0246    0.0203  0.00214
##  7 7       0.960 0.0399   0.0244    0.0144  0.00111
##  8 8       0.962 0.0390   0.0213    0.0118  0.00473
##  9 9       0.976 0.0268   0.0134    0.0109  0
## 10 10      0.973 0.0270   0.0193    0.00771 0
## # ... with 15 more rows
##
## [[3]]
## # A tibble: 25 x 6
##    opp   correct Total If_Error Then_Error  Hint
##    <fct>   <dbl> <dbl>    <dbl>      <dbl> <dbl>
##  1 1         845   309      116        144    35
##  2 2         973   152       63         79     9
##  3 3         999    90       40         45     2
##  4 4         944    83       27         49     6
##  5 5         924    51       23         21     6
##  6 6         892    44       23         19     2
##  7 7         866    36       22         13     1
##  8 8         814    33       18         10     4
##  9 9         802    22       11          9     0
## 10 10        757    21       15          6     0
## # ... with 15 more rows
```

```
curve_tr
```

```
## [[1]]
## # A tibble: 25 x 2
##    opp       n
##    <fct> <int>
##  1 1      1140
##  2 2      1124
##  3 3       1086
##  4 4       1026
##  5 5        974
##  6 6        936
##  7 7        902
##  8 8        846
##  9 9        822
## 10 10       778
## # ... with 15 more rows
##
## [[2]]
## # A tibble: 25 x 6
##    opp   correct  Total If_Error    Hint Then_Error
##    <fct>   <dbl>  <dbl>    <dbl>   <dbl>      <dbl>
##  1 1       0.684 0.319   0.0868  0.123      0.106
##  2 2       0.786 0.214   0.0934  0.0445     0.0765
##  3 3       0.866 0.136   0.0460  0.0157     0.0727
##  4 4       0.912 0.0877  0.0302  0.0136     0.0439
##  5 5       0.960 0.0400  0.0164  0.00719    0.0164
##  6 6       0.973 0.0267  0.0118  0.00534    0.00962
##  7 7       0.978 0.0222  0.00776 0.00665    0.00776
##  8 8       0.965 0.0355  0.0130  0.0106     0.0118
##  9 9       0.978 0.0219  0.00852 0.00365    0.00973
## 10 10      0.976 0.0244  0.0141  0.00257    0.00771
## # ... with 15 more rows
##
## [[3]]
## # A tibble: 25 x 6
##    opp   correct Total If_Error  Hint Then_Error
##    <fct>   <dbl> <dbl>    <dbl> <dbl>      <dbl>
##  1 1         780   364       99   140        121
##  2 2         883   241      105    50         86
##  3 3         940   148       50    17         79
##  4 4         936    90       31    14         45
##  5 5         935    39       16     7         16
##  6 6         911    25       11     5          9
##  7 7         882    20        7     6          7
##  8 8         816    30       11     9         10
##  9 9         804    18        7     3          8
## 10 10        759    19       11     2          6
## # ... with 15 more rows
```

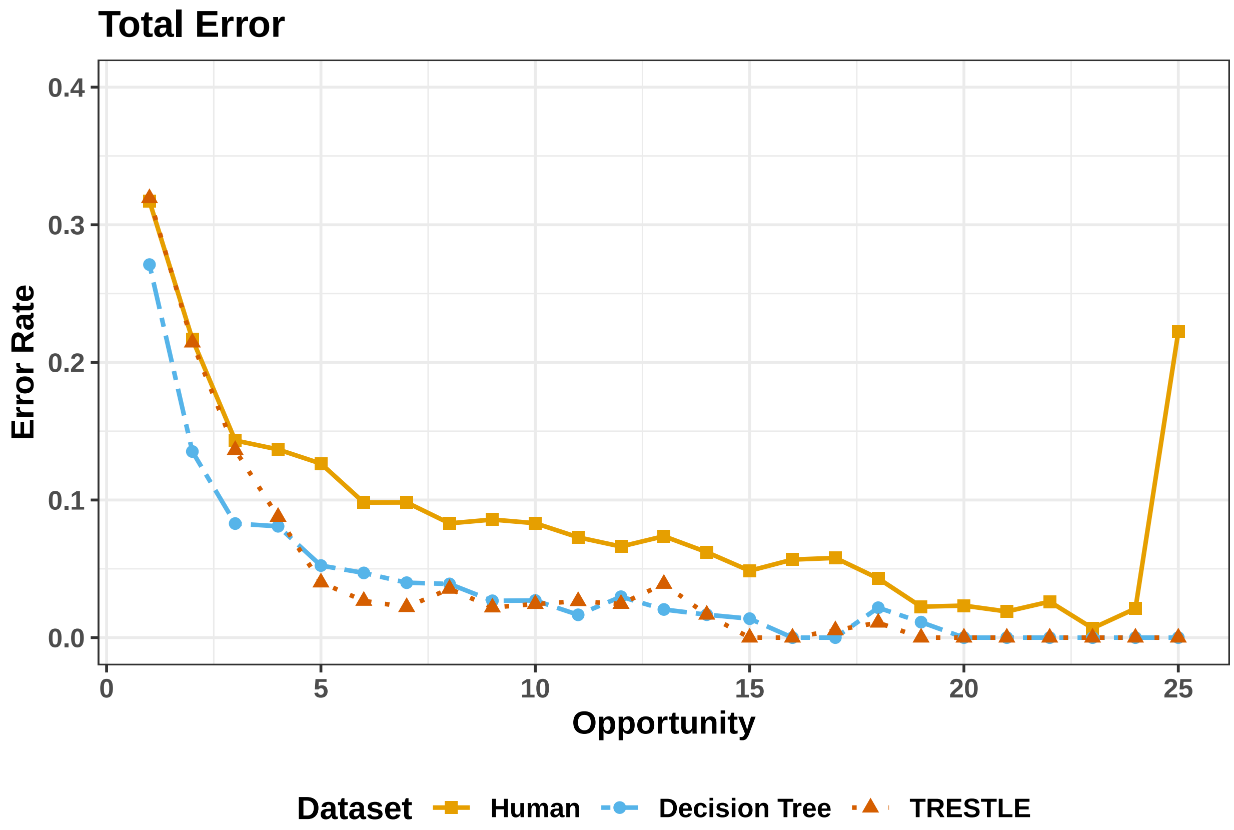
```
curve_dt_in
```

```
## [[1]]
## # A tibble: 25 x 2
##    opp       n
##    <fct> <int>
##  1 1      1140
##  2 2      1124
##  3 3      1086
##  4 4      1026
##  5 5       974
##  6 6       936
##  7 7       902
##  8 8       846
##  9 9       822
## 10 10      778
## # ... with 15 more rows
##
## [[2]]
## # A tibble: 25 x 6
##    opp   correct   Total    Hint If_Error Then_Error
##    <fct>   <dbl>   <dbl>   <dbl>    <dbl>      <dbl>
##  1 1       0.703 0.298   0.171    0.0553     0.0711
##  2 2       0.848 0.153   0.0534   0.0391     0.0596
##  3 3       0.895 0.105   0.0267   0.0387     0.0396
##  4 4       0.931 0.0692  0.0127   0.0156     0.0409
##  5 5       0.968 0.0318  0.00924  0.0133     0.00924
##  6 6       0.976 0.0246  0.00641  0.00855    0.00855
##  7 7       0.982 0.0177  0.00333  0.00222    0.0122
##  8 8       0.993 0.00709 0        0.00118    0.00591
##  9 9       0.987 0.0134  0.00365  0.00730    0.00243
## 10 10      0.994 0.00643 0        0.00386    0.00257
## # ... with 15 more rows
##
## [[3]]
## # A tibble: 25 x 6
##    opp   correct Total  Hint If_Error Then_Error
##    <fct>   <dbl> <dbl> <dbl>    <dbl>      <dbl>
##  1 1         801   340   195       63         81
##  2 2         953   172    60       44         67
##  3 3         972   114    29       42         43
##  4 4         955    71    13       16         42
##  5 5         943    31     9       13          9
##  6 6         914    23     6        8          8
##  7 7         886    16     3        2         11
##  8 8         840     6     0        1          5
##  9 9         811    11     3        6          2
## 10 10        773     5     0        3          2
## # ... with 15 more rows
```

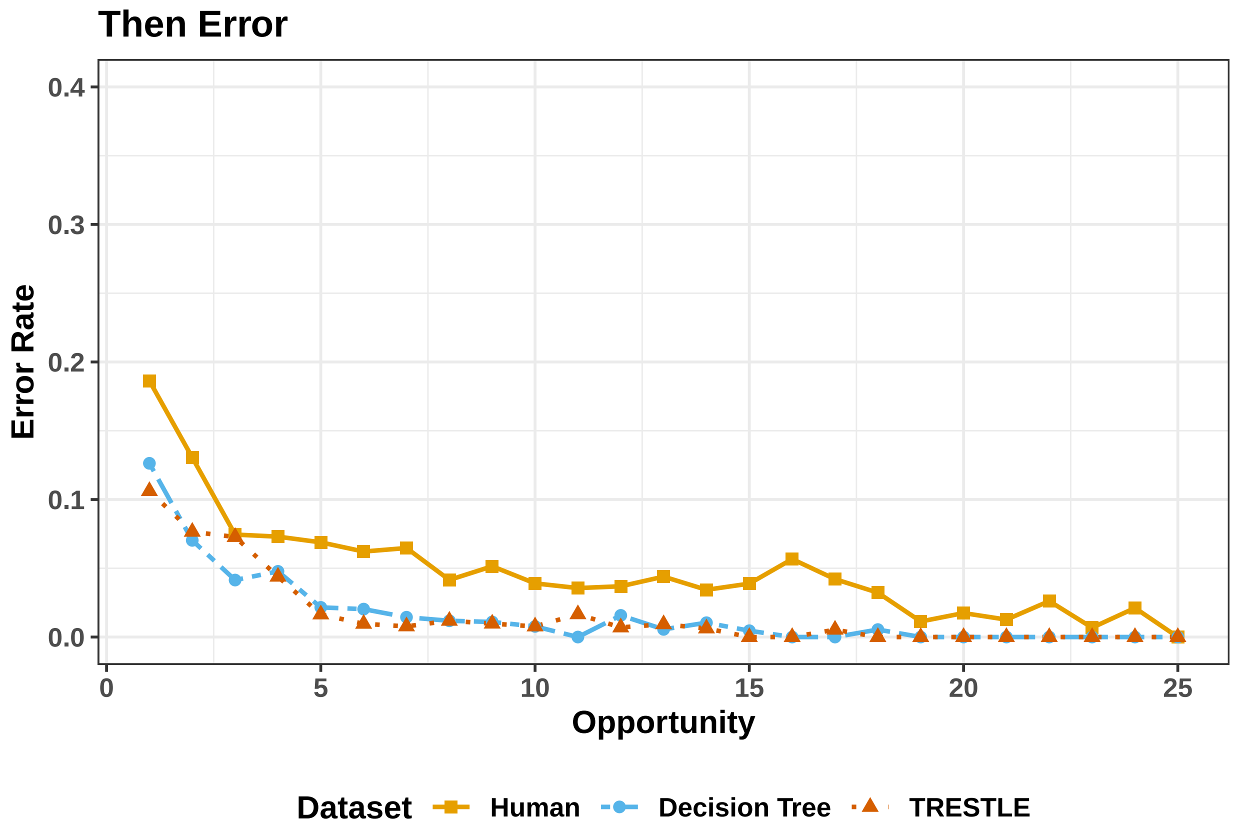
```r
library(plyr)
my_lines <- c("solid", "twodash", "dotted", "dotdash")
my_colors <- c("#E69F00", "#56B4E9", "#D55E00", "#0072B2","#CC79A7","#1B9E77")
my_shapes <- c(15,16,17,18,19)
#reorder vars, remove space in vars, specify aesthetics, blank background, line types, font size, save,



split <- function(agg_results, order, rename, title = "", legend_title = "Dataset", line_size = 0.8, po
  name <- names(agg_results) # get names of datasets, "human", "dt", "tr", "dt_in"
  res <- data.frame(matrix(NA, nrow = 0, ncol = 4)) # res combines agg results from all datasets
  for(i in 1:length(agg_results)){
    agg <- agg_results[[i]]
    curve.agg <- data.frame(agg[[2]])
    curve.agg$dataset <- name[i] # add column indicating which dataset

    # get error names
    colname <- colnames(curve.agg)
    errtype <- colname[!colname %in% c("dataset", "opp")]

    curve.agg <- reshape2::melt(curve.agg, id = c("opp", "dataset"), measure = errtype) %>% filter(vari
    res <- rbind(res, curve.agg)
  }

  plots <- list()

  # Remove spaces
  res$variable <- revalue(res$variable, c("Total" = "Total Error", "If_Error" = "If Error",
                          "Then_Error" = "Then Error", "Hint" = "Hint"))
  unique.err <- (unique(res$variable))

  # Reorder and Rename
  res$dataset <- factor(res$dataset, levels = order)
  res$dataset <- mapvalues(res$dataset, from = order, to = rename)


  # store plots for each kind of error in a list
  for(e in unique.err){
    sel.err <- res[res$variable == e, ]

    # Title Name
    plot.title <- paste(as.character(e), title, sep = " ")

    err.plot <- ggplot(sel.err,
                       aes(x = as.numeric(opp), y = as.numeric(value),
                       colour = dataset, linetype = dataset, shape = dataset)) +
                geom_line(size = line_size) +
                geom_point(size = point_size) +
                ylim(y_range) +
                labs(title = plot.title, x = "Opportunity", y = "Error Rate") +
                scale_colour_manual(name = legend_title, values = my_colors) +
                scale_linetype_manual(name = legend_title, values = my_lines) +
                scale_shape_manual(name = legend_title, values = my_shapes) +
                theme_bw() +
                theme(text = element_text(family="Helvetica", face="bold", size=12),
                      legend.position ="bottom",
                      #panel.background = element_rect(fill = 'white', color = "black"),
                      #panel.grid.major = element_blank(),panel.grid.minor = element_blank())
                )
              #  guides(color=guide_legend(title='NEW TITLE'))

    plots[[e]] <- err.plot
  }
   return(plots)
}
```

```
# Put aggregated results of different datasets in a list in "agg_result"
# Specify order of variables in legend in "order"
# Rename in the order of variables in "rename"

plot.by.error <- split(agg_result = list("human" = curve_human, "dt" = curve_dt,
                             "tr" = curve_tr),#"dt_in" = curve_dt_in),
           order = c("human", "dt", "tr"),#  "dt_in"),
           rename = c("Human", "Decision Tree", "TRESTLE"))#, "Decision Tree IN"))
plot.by.error
```

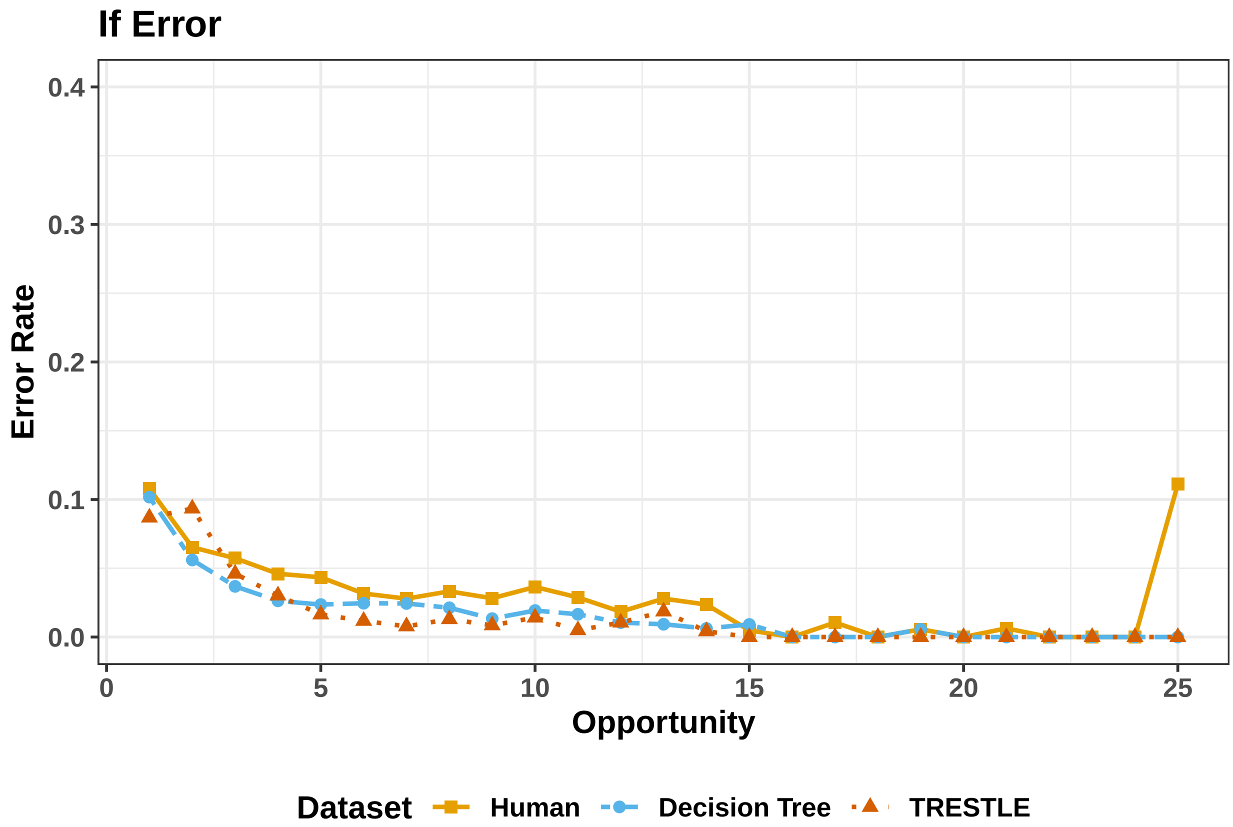
```
## $`Total Error`
```



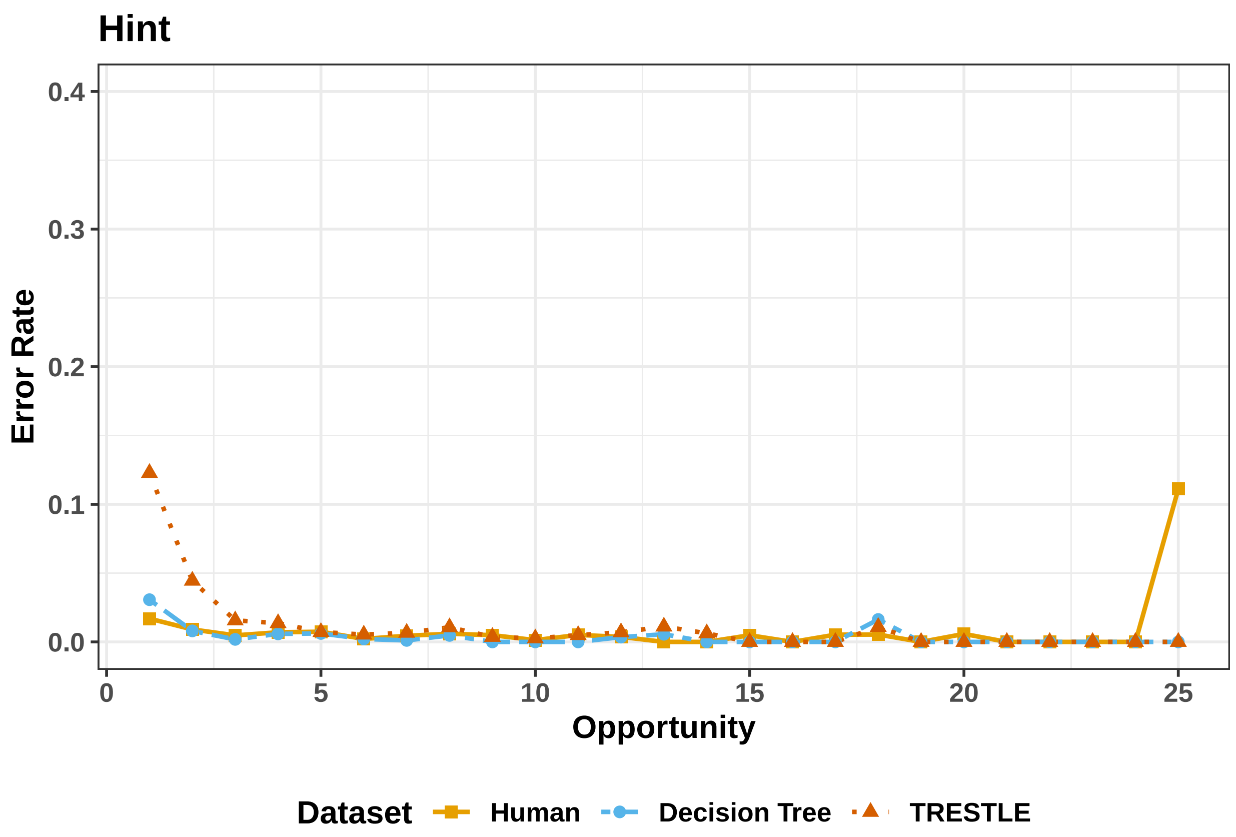
```
##
## $`Then Error`
```


Then Error
Error Rate
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
Opportunity
Dataset
Human
Decision Tree
TRESTLE

```
##
## $`If Error`
```


If Error
Error Rate
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
Opportunity
Dataset
Human
Decision Tree
TRESTLE

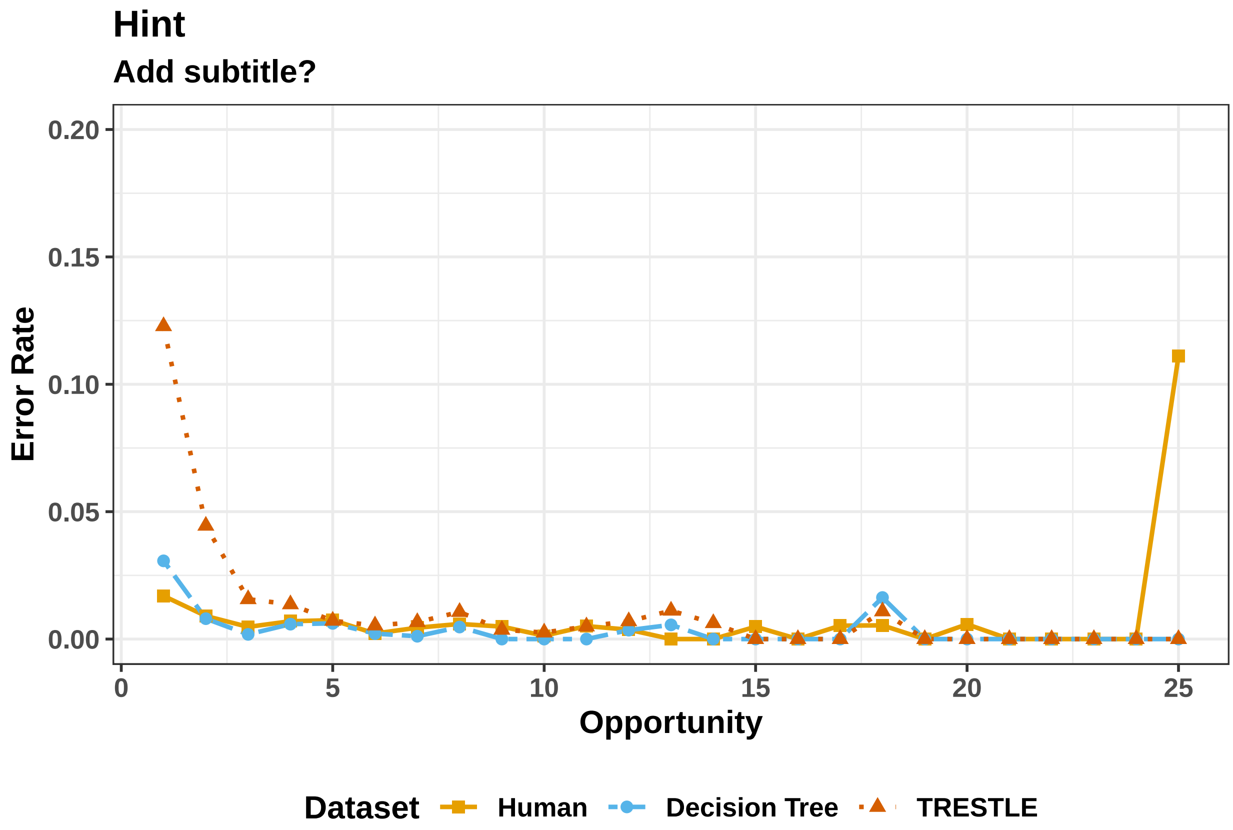
```
##
## $Hint
```



```
# Get individual error plot from list of plots, add subtitle, adjust ylim
names(plot.by.error)
```

```
## [1] "Total Error" "Then Error"  "If Error"    "Hint"
```

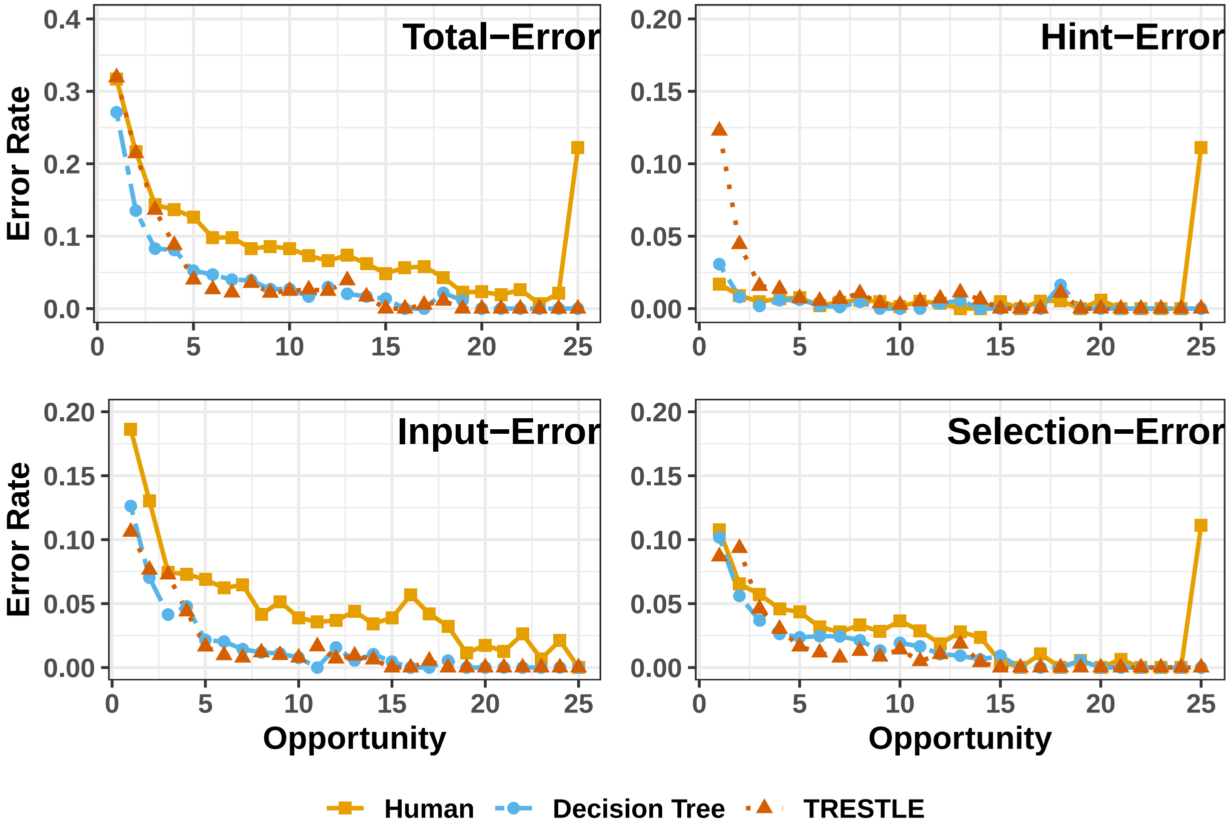
```
hint.plot <- plot.by.error$"Hint" + labs(subtitle = "Add subtitle?") + ylim(0, 0.2)
hint.plot
```



```
# Save plot
#ggsave("~/...", plot = hint.plot, device='eps')
```

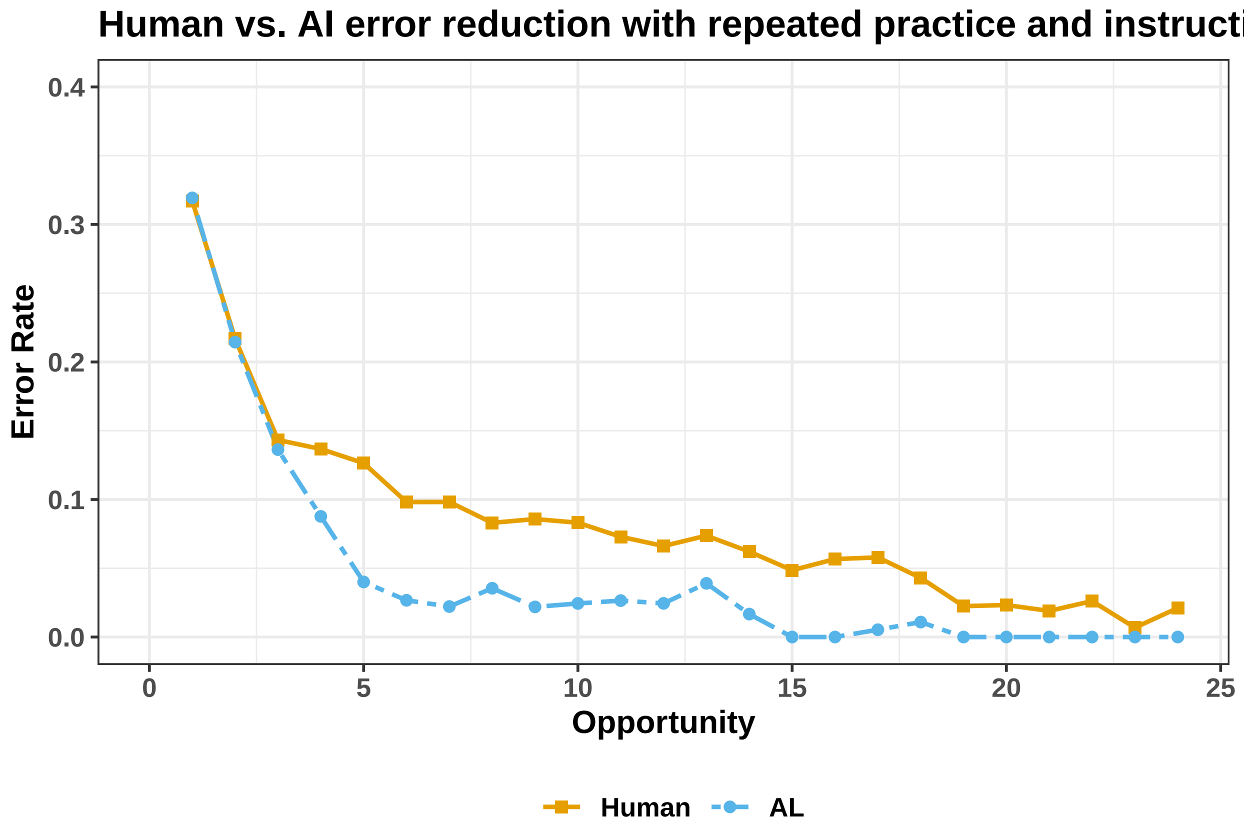
```
#legend = get_legend(plot.by.error$"Total Error" + theme(legend.title =element_blank()))
total.plot <- plot.by.error$"Total Error" + theme(legend.title = element_blank(),
    #axis.ticks.x = element_blank(),
    axis.title.x = element_blank(),
    #axis.text.x = element_blank(),
    plot.title = element_text(margin = margin(t = 10, b = -20),hjust=1)) +
    ggtitle("Total-Error")
hint.plot <- plot.by.error$"Hint" + ylim(0, 0.2) + theme(legend.title = element_blank(),
    #axis.ticks.x = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    #axis.text.x = element_blank(),
    plot.title = element_text(margin = margin(t = 10, b = -20),hjust=1)) +
    ggtitle("Hint-Error")
then.plot <- plot.by.error$"Then Error" + ylim(0, 0.2) + theme(legend.title = element_blank(), plot.titl
    ggtitle("Input-Error")
if.plot <- plot.by.error$"If Error" + ylim(0, 0.2) + theme(legend.title = element_blank(), plot.title =
    ggtitle("Selection-Error")

#if.plot
#then.plot
#hint.plot

#m <- matrix(c(1,2,3,4,7,7),nrow = 3,ncol = 2,byrow = TRUE)
```

```
#layout(mat = m,heights = c(0.4,0.4,0.2))

library(ggpubr)


all_plots <- ggarrange(total.plot, hint.plot, then.plot, if.plot,  nrow=2, ncol=2, common.legend = TRUE
#grid.arrange(all_plots, legend, nrow=2)
all_plots
```



```
ggsave("all_plots.eps", plot = all_plots, device='eps')
```

```
justHtr <- split(agg_result = list("human" = curve_human, "AL" = curve_tr),
           order = c("human", "AL"),
           rename = c("Human", "AL"),
           )
out <- justHtr$"Total Error"  + theme(legend.title = element_blank()) + scale_x_continuous(limits = c(0


#, plot.title = element_text(margin = margin(t = 10, b = -20),hjust=1), axis.title.y = element_blank()
ggsave("al_human_total.eps", plot = out, device='eps')

out
```



```
#total.plot + scale_fill_manual(values=c("#999999", "#E69F00", "#56B4E9"),
#                       name="Experimental\nCondition",
#                       breaks=c("ctrl", "trt1", "trt2"),
#                       labels=c("Control", "Treatment 1", "Treatment 2"))#+ scale_colour_manual(values
```