MI - repetition - Serial position analysis (v5)

```r
# do this in calling R script
# library(ggplot2)
# library(fmsb) # for TestModels
# library(lme4)
# library(kableExtra)
# library(MASS) # for dropterm
# library(tidyverse)
# library(dominanceanalysis)
# library(cowplot) # for multiple plots in one figure
# library(formatters) # to wrap strings for plot titles
# library(pals) # for continous color palettes
```

```r
# load needed functions
# source(paste0(RootDir,"/src/function_library/sp_functions.R"))
# do this in the calling R script
```

```r
# for testing - set by calling script normally
# ppt_parms <- read.csv(config$patient_param_file)
# CurPat <- ppt_parms$patient
# CurTask <- ppt_parms$task
# MinLength <- ppt_parms$min_length
# MaxLength <- ppt_parms$max_length
# RemoveFinalPosition<-ppt_parms$remove_final_position
```

```r
# create patient directory if it doesn't already exist
if(!dir.exists(paste0(RootDir,"/output/",CurPat))){
  dir.create(paste0(RootDir,"/output/",CurPat),showWarnings = TRUE)
}
TablesDir <- paste0(RootDir,"/output/",CurPat,"/tables/")
if(!dir.exists(TablesDir)){
  dir.create(TablesDir,showWarnings = TRUE)
}
FigDir <- paste0(RootDir,"/output/",CurPat,"/fig/")
if(!dir.exists(FigDir)){
  dir.create(FigDir,showWarnings = TRUE)
}
ReportsDir <- paste0(RootDir,"/output/",CurPat,"/reports/")
if(!dir.exists(ReportsDir)){
  dir.create(ReportsDir,showWarnings = TRUE)
}
results_report_DF <- data.frame(Description=character(), ParameterValue=double())
```

```r
ModelDatFilename<-paste0(RootDir,"/data/",CurPat,"/",CurPat,"_",CurTask,"_posdat.csv")
if(!file.exists(ModelDatFilename)){
  print("**ERROR** Input data file not found")
  print(sprintf("\tfile: ",ModelDatFilename))
  print(sprintf("\troot dir: ",RootDir))
}
PosDat<-read.csv(ModelDatFilename)
```

```
# may already be done in datafile
# if(RemoveFinalPosition){
#   PosDat <- PosDat[PosDat$pos < PosDat$stimlen,] # remove final position
# }
PosDat <- PosDat[PosDat$stimlen >= MinLength,] # include only lengths with sufficient N
PosDat <- PosDat[PosDat$stimlen <= MaxLength,]
# include only correct and nonword errors (not no response, word or w+nw errors)
PosDat <- PosDat[(PosDat$err_type == "correct") | (PosDat$err_type == "nonword"), ]
PosDat <- PosDat[(PosDat$pos) != (PosDat$stimlen), ]
# make sure final positions have been removed
PosDat$stim_number <- NumberStimuli(PosDat)
PosDat$raw_log_freq <- PosDat$log_freq
PosDat$log_freq <- PosDat$log_freq - mean(PosDat$log_freq) # centre frequency
PosDat$CumErr <- CalcCumErrFromPreserved(PosDat)
```

```
# plot distribution of complex onsets and codas
# across serial position in the stimuli -- do complexities concentrate
# on one part of the serial position curve?
# syll_component = 1 is coda
# syll_component = S is satellite

# get counts of syllable components in different serial positions
syll_comp_dist_N <- PosDat %>% mutate(pos_factor = as.factor(pos), syll_component_factor = as.factor(syl

syll_comp_dist_perc <- syll_comp_dist_N %>% ungroup() %>%
  mutate(across(O:S, ~ . / total))

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	550	33	127	NA	NA	710
2	62	NA	439	98	111	710
3	313	NA	169	213	15	710
4	307	NA	242	66	38	653
5	232	NA	215	73	38	558
6	210	1	136	72	22	441
7	178	NA	105	26	18	327
8	90	NA	55	26	4	175
9	76	NA	2	NA	7	85

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7746479	0.0464789	0.1788732	NA	NA	710
2	0.0873239	NA	0.6183099	0.1380282	0.1563380	710
3	0.4408451	NA	0.2380282	0.3000000	0.0211268	710
4	0.4701378	NA	0.3705972	0.1010720	0.0581930	653
5	0.4157706	NA	0.3853047	0.1308244	0.0681004	558
6	0.4761905	0.0022676	0.3083900	0.1632653	0.0498866	441

pos_factor	O	P	V	1	S	total
7	0.5443425	NA	0.3211009	0.0795107	0.0550459	327
8	0.5142857	NA	0.3142857	0.1485714	0.0228571	175
9	0.8941176	NA	0.0235294	NA	0.0823529	85

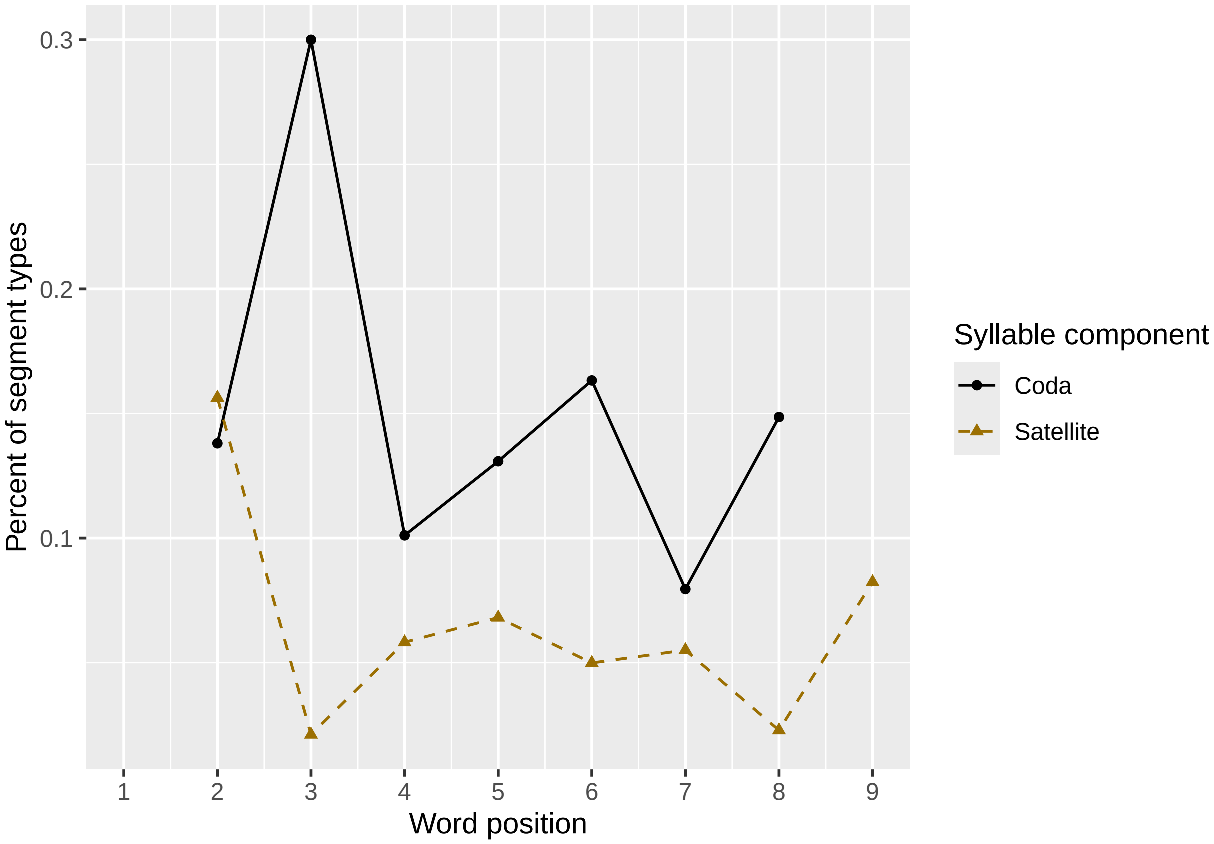
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# pivot to long format for plotting
syll_comp_plot_data <- syll_comp_perc %>% pivot_longer(cols=seq(1:(ncol(syll_comp_perc)-1)),
            names_to="syll_component", values_to="percent") %>% filter(syll_component %in% c("Coda","Sa
# plot satellite and coda occurances across position

syll_comp_plot <- ggplot(syll_comp_plot_data,
                         aes(x=pos,y=percent,group=syll_component,
                             color=syll_component,
                             linetype = syll_component,
                             shape = syll_component)) +
  geom_line() + geom_point() + scale_color_manual(name="Syllable component", values = palette_values) +
syll_comp_plot <- syll_comp_plot + scale_y_continuous(name="Percent of segment types") + scale_linetype_
  scale_shape_discrete(name="Syllable component")
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_pos_of_syll_complexities_in_stimuli.png"),plot=syll_comp_plot
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
syll_comp_plot
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range (`geom_line()`).
## Removed 3 rows containing missing values or values outside the scale range (`geom_point()`).
```



```
# len/pos table
pos_len_summary <- PosDat %>% group_by(stimlen,pos) %>% summarise(preserved = mean(preserved))
```

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

```
min_preserved_raw <- min(as.numeric(unlist(pos_len_summary$preserved)))
max_preserved_raw <- max(as.numeric(unlist(pos_len_summary$preserved)))
preserved_range <- (max_preserved_raw - min_preserved_raw)
min_preserved <- min_preserved_raw - (0.1 * preserved_range)
max_preserved <- max_preserved_raw + (0.1 * preserved_range)
pos_len_table <- pos_len_summary %>% pivot_wider(names_from = pos, values_from = preserved)
write.csv(pos_len_table,paste0(TablesDir,CurPat,"_",CurTask,"_preserved_percentages_by_len_position.csv
pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.947 0.982 0.930 NA     NA     NA     NA     NA     NA
## 2       5 0.905 0.937 0.926  0.884 NA     NA     NA     NA     NA
## 3       6 0.915 0.889 0.897  0.957  0.855 NA     NA     NA     NA
## 4       7 0.854 0.958 0.901  0.844  0.965  0.908 NA     NA     NA
## 5       8 0.842 0.917 0.929  0.910  0.910  0.936  0.910 NA     NA
## 6       9 0.878 0.911 0.867  0.933  0.889  0.889  0.933  0.9   NA
## 7      10 0.882 0.892 0.825  0.865  0.959  0.882  0.876  0.959  0.876
```

```
# len/pos table
pos_len_N <- PosDat %>% group_by(stimlen,pos) %>% summarise(preserved = mean(preserved),N=n()) %>% dply
```

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

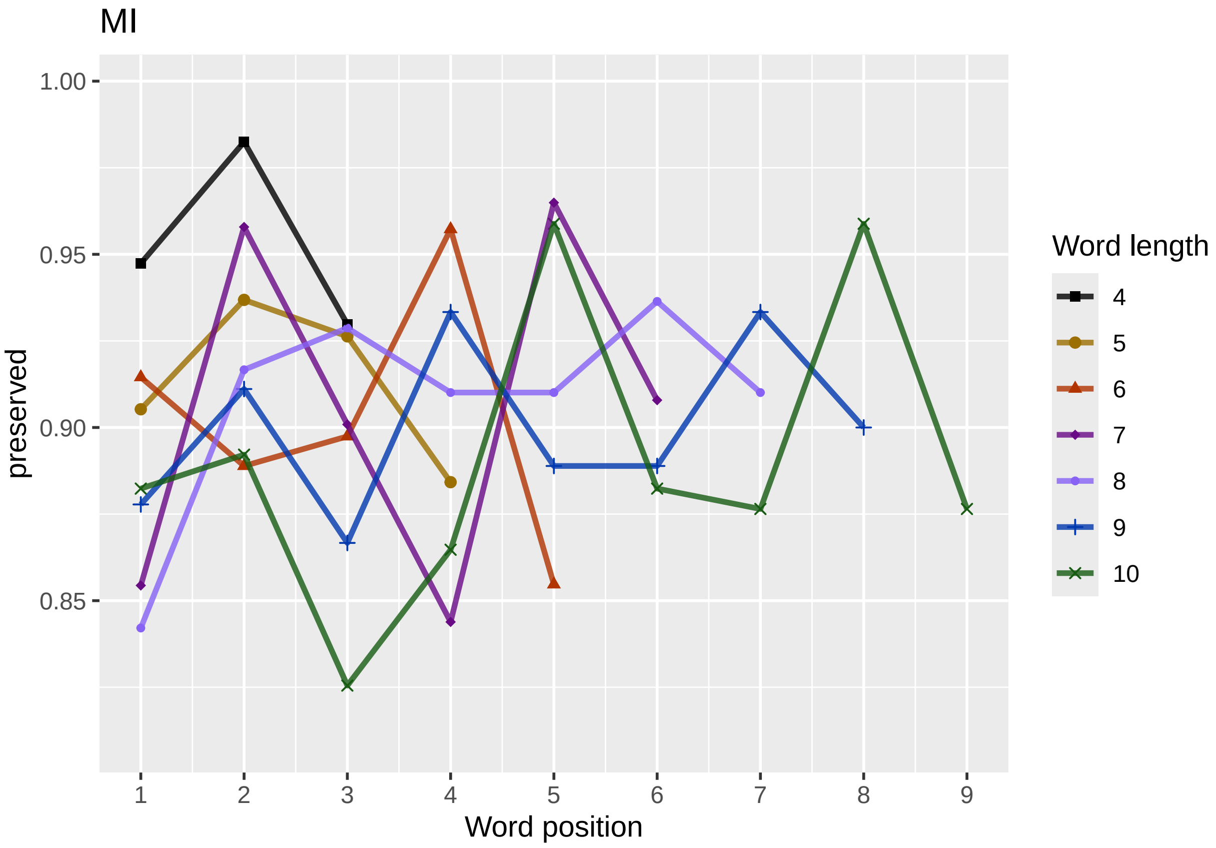
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       4    57    57    57    NA    NA    NA    NA    NA    NA
## 2       5    95    95    95    95    NA    NA    NA    NA    NA
## 3       6   117   117   117   117   117    NA    NA    NA    NA
## 4       7   114   114   114   114   114   114    NA    NA    NA
## 5       8   152   152   152   152   152   152   152    NA    NA
## 6       9    90    90    90    90    90    90    90    90    NA
## 7      10    85    85    85    85    85    85    85    85    85
```

```
obs_linetypes <- c("solid","solid","solid","solid",
                      "solid","solid","solid","solid","solid")
pred_linetypes <- "dashed"
pos_len_summary$stimlen<-factor(pos_len_summary$stimlen)
pos_len_summary$pos<-factor(pos_len_summary$pos)
# len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color=stimle
len_pos_plot <- plot_len_pos_obs_predicted(PosDat,
                                            paste0(CurPat),
                                            NULL,
                                            c(min_preserved,max_preserved),
                                            palette_values,
                                            shape_values,
                                            obs_linetypes,
                                            pred_linetypes)
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace the existing
## scale.
```

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_percent_preserved_by_length_pos.png"),
       plot=len_pos_plot,device="png",unit="cm",width=15,height=11)
len_pos_plot
```



Length and position

```
# length and position

LPModelEquations<-c("preserved ~ 1",
            "preserved ~ stimlen",
            "preserved ~ pos",
            "preserved ~ stimlen + pos",
            "preserved ~ stimlen*pos",
            "preserved ~ I(pos^2)+pos",
            "preserved ~ stimlen + I(pos^2) + pos",
            "preserved ~ stimlen * (I(pos^2) + pos)"
)

LPRes<-TestModels(LPModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  4
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.72172     -0.08530      0.04972
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2690  AIC: 2808
## log likelihood:  -1344.886
## Nagelkerke R2:  0.00382785
## % pres/err predicted correctly:  -747.6043
## % of predictable range [ (model-null)/(1-null) ]:  0.002089289
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.72578     -0.06147
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2693  AIC: 2809
## log likelihood:  -1346.749
## Nagelkerke R2:  0.001978647
## % pres/err predicted correctly:  -748.3192
## % of predictable range [ (model-null)/(1-null) ]:  0.001136294
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.11496     -0.13270     -0.08109      0.01533
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2689  AIC: 2809
## log likelihood:  -1344.479
## Nagelkerke R2:  0.004231169
## % pres/err predicted correctly:  -747.4736
## % of predictable range [ (model-null)/(1-null) ]:  0.00226352
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.621041    -0.082314    -0.006003     0.100267
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2689  AIC: 2810
## log likelihood:  -1344.731
## Nagelkerke R2:  0.003980993
## % pres/err predicted correctly:  -747.5731
## % of predictable range [ (model-null)/(1-null) ]:  0.002130889
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##       3.2773380       -0.1784781        -0.0109344        -0.1134543        -0.0007444
##      stimlen:pos
##       0.0362505
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4363 Residual
## Null Deviance:       2697
## Residual Deviance: 2687  AIC: 2812
## log likelihood:  -1343.604
## Nagelkerke R2:  0.005099155
## % pres/err predicted correctly:  -747.2577
## % of predictable range [ (model-null)/(1-null) ]:  0.002551338
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.25
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4368 Residual
## Null Deviance:       2697
## Residual Deviance: 2697  AIC: 2813
## log likelihood:  -1348.741
## Nagelkerke R2:  -4.820112e-16
## % pres/err predicted correctly:  -749.1717
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     2.15566      0.02483
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2696  AIC: 2814
## log likelihood:  -1348.216
## Nagelkerke R2:  0.0005216602
## % pres/err predicted correctly:  -749.0171
## % of predictable range [ (model-null)/(1-null) ]:  0.000206091
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.01653     -0.01037      0.11353
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2695  AIC: 2815
## log likelihood:  -1347.744
## Nagelkerke R2:  0.0009903871
## % pres/err predicted correctly:  -748.885
## % of predictable range [ (model-null)/(1-null) ]:  0.0003821438
## ***************************
```

```
BestLPModel<-LPRes$ModelResult[[1]]
BestLPModelFormula<-LPRes$Model[[1]]

LPAICSummary<-data.frame(Model=LPRes$Model,
                        AIC=LPRes$AIC,
                        row.names = LPRes$Model)
LPAICSummary$DeltaAIC<-LPAICSummary$AIC-LPAICSummary$AIC[1]
LPAICSummary$AICexp<-exp(-0.5*LPAICSummary$DeltaAIC)
LPAICSummary$AICwt<-LPAICSummary$AICexp/sum(LPAICSummary$AICexp)
LPAICSummary$NagR2<-LPRes$NagR2

LPAICSummary <- merge(LPAICSummary,LPRes$CoefficientValues, by='row.names',sort=FALSE)
LPAICSummary <- subset(LPAICSummary, select = -c(Row.names))

write.csv(LPAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_len_pos_model_summary.csv"),row.names = FA
kable(LPAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	2807.990	0.000000	1.000000	0.3537729	0.0038272	2.8721719	-0.0853037	0.0497225	NA	NA	NA
preserved ~ stimlen	2809.088	1.096827	0.5778659	0.2044332	0.0019786	2.6725775	-0.0614693	NA	NA	NA	NA
preserved ~ stimlen * pos	2809.220	1.229070	0.5408924	0.1913530	0.0042313	3.2114964	-0.1327015	-0.0810898	0.0153260	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	2809.831	1.840910	0.3983359	0.1409204	0.0039812	2.0621041	-0.0823139	0.1002675	NA	-0.0060029	NA
preserved ~ stimlen * (I(pos^2) + pos)	2811.933	3.945310	0.1390868	0.0492051	0.0050932	3.2277338	-0.1784781	-0.1134543	0.0362505	-0.0109340	-0.0007444
preserved ~ 1	2812.599	4.607505	0.0998833	0.0353360	0.0000000	2.0249775	NA	NA	NA	NA	NA
preserved ~ pos	2814.161	6.169330	0.0457454	0.0161835	0.0005212	1.7155663	NA	0.0248264	NA	NA	NA
preserved ~ I(pos^2) + pos	2815.387	7.388739	0.0248631	0.0087959	0.0009902	1.4016526	NA	0.1135261	NA	-0.0103730	NA

```r
print(BestLPModelFormula)
```

```
## [1] "preserved ~ stimlen + pos"
```

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.72172     -0.08530      0.04972
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2690  AIC: 2808
```

```r
PosDat$LPFitted<-fitted(BestLPModel)
fitted_len_pos_plot <- plot_len_pos_obs_predicted(PosDat, PosDat$patient[1],
                          NULL,palette_values=palette_values)
```

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

```r
# len/pos table
fitted_pos_len_summary <- PosDat %>% group_by(stimlen,pos) %>% summarise(fitted = mean(LPFitted))
```

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

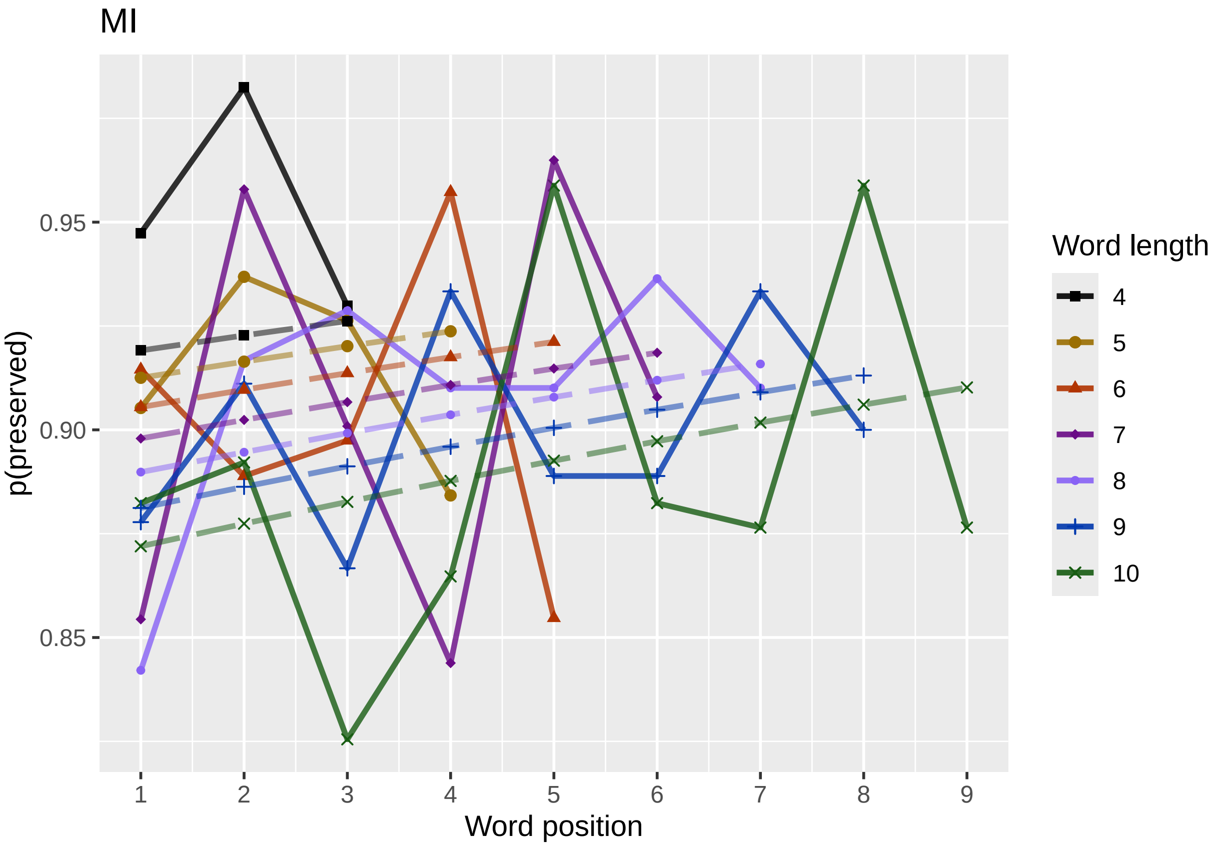
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.919 0.923 0.926 NA     NA     NA     NA     NA     NA
## 2       5 0.913 0.916 0.920  0.924 NA     NA     NA     NA     NA
## 3       6 0.905 0.910 0.914  0.917  0.921 NA     NA     NA     NA
## 4       7 0.898 0.902 0.907  0.911  0.915  0.919 NA     NA     NA
## 5       8 0.890 0.895 0.899  0.904  0.908  0.912  0.916 NA     NA
## 6       9 0.881 0.886 0.891  0.896  0.900  0.905  0.909  0.913 NA
## 7      10 0.872 0.877 0.883  0.888  0.893  0.897  0.902  0.906  0.910
```

```
fitted_pos_len_summary$stimlen<-factor(fitted_pos_len_summary$stimlen)
fitted_pos_len_summary$pos<-factor(fitted_pos_len_summary$pos)
# fitted_len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color
# fitted_len_pos_plot <- fitted_len_pos_plot + geom_line(data=fitted_pos_len_summary, aes(x=pos,y=fitte
# geom_point(data=fitted_pos_len_summary,aes(x=pos,y=fitted,group=stimlen,shape=stimlen)) + ggtitle(pas

fitted_len_pos_plot <- plot_len_pos_obs_predicted(PosDat,
                                         paste0(PosDat$patient[1]),
                                         "LPFitted",
                                         NULL,
                                         palette_values,
                                         shape_values,
                                         obs_linetypes,
                                         pred_linetypes = c("longdash")
                                         )
```

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_percent_preserved_by_length_pos_wfit.png"),plot=fitted_len_pos
fitted_len_pos_plot
```



length and position without fragments to see if this changes position^2 influence

```
# first number responses, then count resp with fragments - below we will eliminate fragments
# and re-run models
# number responses
resp_num<-0
prev_pos<-9999 # big number to initialize (so first position is smaller)
resp_num_array <- integer(length = nrow(PosDat))
for(i in seq(1,nrow(PosDat))){
  if(PosDat$pos[i] <= prev_pos){ # equal for resp length 1
    resp_num <- resp_num + 1
  }
  resp_num_array[i]<-resp_num
  prev_pos <- PosDat$pos[i]
}
PosDat$resp_num <- resp_num_array
```

```
# count responses with fragments
resp_with_frag <- PosDat %>% group_by(resp_num) %>% summarise(frag = as.logical(sum(fragment)))
num_frag <- resp_with_frag %>% summarise(frag_sum = sum(frag), N=n())
num_frag
```

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        4   710
```

```
num_frag$percent_with_frag[1] <- (num_frag$frag_sum[1] / num_frag$N[1])*100
```

```
## Warning: Unknown or uninitialised column: `percent_with_frag`.
```

```
print(sprintf("The number of responses with fragments was %d / %d = %.2f percent",
              num_frag$frag_sum[1],num_frag$N[1],num_frag$percent_with_frag[1]))
```

```
## [1] "The number of responses with fragments was 4 / 710 = 0.56 percent"
```

```
write.csv(num_frag,paste0(TablesDir,CurPat,"_",CurTask,"_percent_fragments.csv"),row.names = FALSE)
```

```
# length and position models for data without fragments
NoFragData <- PosDat %>% filter(fragment != 1)

NoFrag_LPRes<-TestModels(LPModelEquations,NoFragData)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.70181     -0.08657      0.06340
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4355 Residual
## Null Deviance:       2661
## Residual Deviance: 2652  AIC: 2760
## log likelihood:  -1325.782
## Nagelkerke R2:  0.004633376
## % pres/err predicted correctly:  -733.3
## % of predictable range [ (model-null)/(1-null) ]:  0.002474346
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.08930     -0.13332     -0.06689      0.01529
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4354 Residual
## Null Deviance:       2661
## Residual Deviance: 2651  AIC: 2761
## log likelihood:  -1325.39
## Nagelkerke R2:  0.005026839
## % pres/err predicted correctly:  -733.1737
## % of predictable range [ (model-null)/(1-null) ]:  0.002645848
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.642775    -0.084832    -0.003589     0.093398
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4354 Residual
## Null Deviance:       2661
## Residual Deviance: 2651  AIC: 2762
## log likelihood:  -1325.73
## Nagelkerke R2:  0.004686305
## % pres/err predicted correctly:  -733.2923
## % of predictable range [ (model-null)/(1-null) ]:  0.002484731
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.70907     -0.05667
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4356 Residual
## Null Deviance:       2661
## Residual Deviance: 2657  AIC: 2763
## log likelihood:  -1328.735
## Nagelkerke R2:  0.001672486
## % pres/err predicted correctly:  -734.418
## % of predictable range [ (model-null)/(1-null) ]:  0.0009555498
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         3.342791         -0.183548          0.005055         -0.187204         -0.002100
##      stimlen:pos
##         0.042376
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4352 Residual
## Null Deviance:       2661
## Residual Deviance: 2650  AIC: 2764
## log likelihood:  -1324.841
## Nagelkerke R2:  0.005577054
## % pres/err predicted correctly:  -733.0416
## % of predictable range [ (model-null)/(1-null) ]:  0.002825274
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.271
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4357 Residual
## Null Deviance:       2661
## Residual Deviance: 2661  AIC: 2766
## log likelihood:  -1330.401
## Nagelkerke R2:  2.429633e-16
## % pres/err predicted correctly:  -735.1214
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.12624      0.03841
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4356 Residual
## Null Deviance:       2661
## Residual Deviance: 2658  AIC: 2766
## log likelihood:  -1329.181
## Nagelkerke R2:  0.00122493
## % pres/err predicted correctly:  -734.7316
## % of predictable range [ (model-null)/(1-null) ]:  0.0005294643
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.019200    -0.008121     0.107288
##
## Degrees of Freedom: 4357 Total (i.e. Null);  4355 Residual
## Null Deviance:       2661
## Residual Deviance: 2658  AIC: 2768
## log likelihood:  -1328.904
## Nagelkerke R2:  0.00150241
## % pres/err predicted correctly:  -734.6551
## % of predictable range [ (model-null)/(1-null) ]:  0.0006334229
## ***************************
```

```
NoFragBestLPModel<-NoFrag_LPRes$ModelResult[[1]]
NoFragBestLPModelFormula<-NoFrag_LPRes$Model[[1]]

NoFragLPAICSummary<-data.frame(Model=NoFrag_LPRes$Model,
                        AIC=NoFrag_LPRes$AIC,
                        row.names = NoFrag_LPRes$Model)
NoFragLPAICSummary$DeltaAIC<-NoFragLPAICSummary$AIC-NoFragLPAICSummary$AIC[1]
NoFragLPAICSummary$AICexp<-exp(-0.5*NoFragLPAICSummary$DeltaAIC)
NoFragLPAICSummary$AICwt<-NoFragLPAICSummary$AICexp/sum(NoFragLPAICSummary$AICexp)
NoFragLPAICSummary$NagR2<-NoFrag_LPRes$NagR2

NoFragLPAICSummary <- merge(NoFragLPAICSummary,NoFrag_LPRes$CoefficientValues, by='row.names',sort=FALSE
NoFragLPAICSummary <- subset(NoFragLPAICSummary, select = -c(Row.names))

write.csv(NoFragLPAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_no_fragments_len_pos_model_summary.cs
kable(NoFragLPAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	2759.825	0.000000	1.0000000	0.4284089	0.0046333	2.4701814	-0.0865650	0.0633983	NA	NA	NA
preserved ~ stimlen * pos	2761.031	1.205707	0.5472470	0.2344458	0.0050268	3.8089300	-0.1333201	-0.0668947	0.0152863	NA	NA
preserved ~ stimlen + I(pos^2) + pos	2761.810	1.984649	0.3707139	0.1588171	0.0046862	2.3642775	-0.0848318	0.0933982	NA	-0.0035894	NA
preserved ~ stimlen	2763.143	3.317216	0.1904038	0.0815707	0.0016722	2.5709070	-0.0566674	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	2764.241	4.415296	0.1099590	0.0471070	0.0055773	3.1342791	-0.1835477	-0.1872036	0.0423764	0.0050551	-0.0021005

Model	AIC	DeltaAIC	expAIC	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ 1	2765.744	5.918889	0.0518477	0.0222120	0.0000000	2.270655	NA	NA	NA	NA	NA
preserved ~ pos	2766.069	6.243855	0.0440721	0.0188809	0.0012249	2.126244	NA	0.0384106	NA	NA	NA
preserved ~ I(pos^2) + pos	2767.652	7.826633	0.0199740	0.0085571	0.0015024	2.019200	NA	0.1072879	NA	-0.0081207	NA

```
# plot no fragment data

NoFragData$LPFitted<-fitted(NoFragBestLPModel)
nofrag_obs_len_pos_plot <- plot_len_pos_obs_predicted(NoFragData, NoFragData$patient[1],
                             NULL,palette_values=palette_values)
```

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

```
# len/pos table
nofrag_fitted_pos_len_summary <- NoFragData %>% group_by(stimlen,pos) %>% summarise(fitted = mean(LPFit
```

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

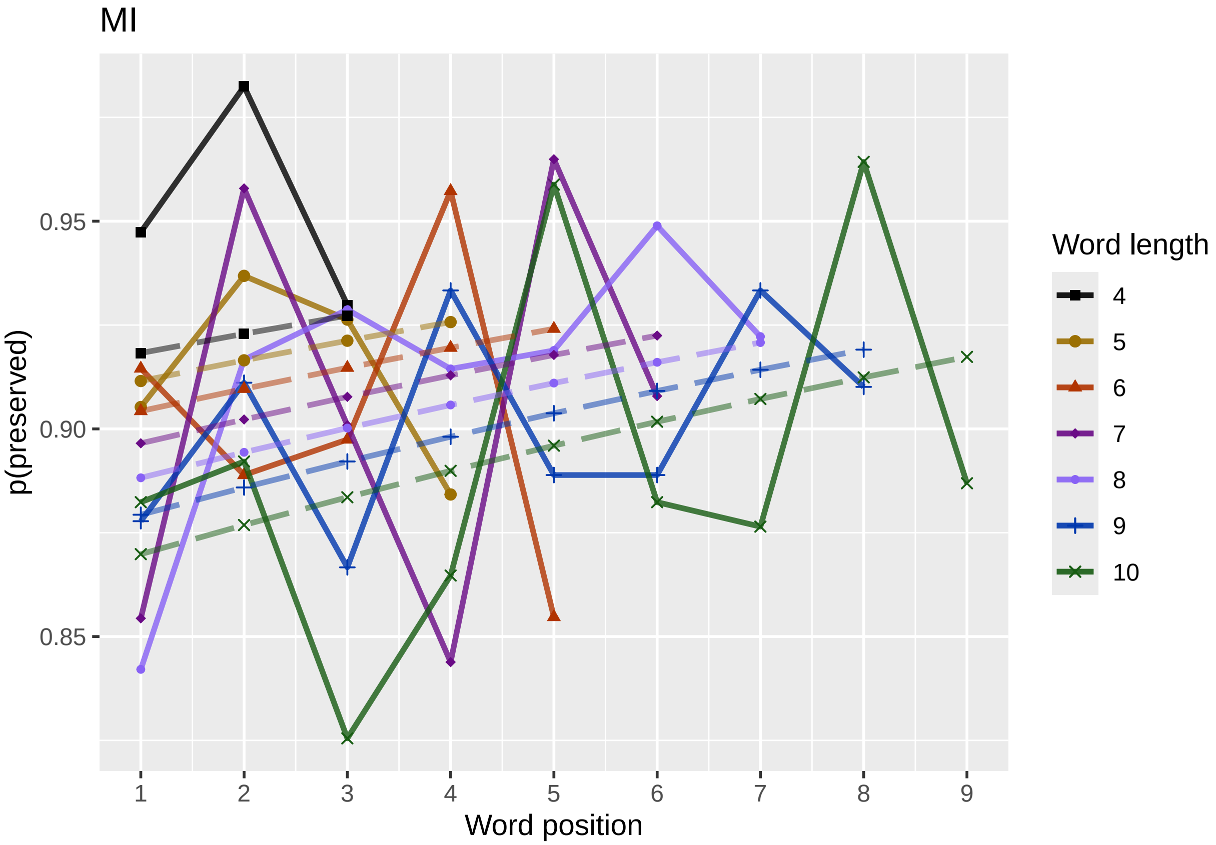
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.918 0.923 0.927 NA     NA     NA     NA     NA     NA
## 2       5 0.912 0.917 0.921  0.926 NA     NA     NA     NA     NA
## 3       6 0.904 0.910 0.915  0.920  0.924 NA     NA     NA     NA
## 4       7 0.897 0.902 0.908  0.913  0.918  0.922 NA     NA     NA
## 5       8 0.888 0.894 0.900  0.906  0.911  0.916  0.921 NA     NA
## 6       9 0.879 0.886 0.892  0.898  0.904  0.909  0.914  0.919 NA
## 7      10 0.870 0.877 0.884  0.890  0.896  0.902  0.907  0.912  0.917
```

```
fitted_pos_len_summary$stimlen<-factor(nofrag_fitted_pos_len_summary$stimlen)
fitted_pos_len_summary$pos<-factor(nofrag_fitted_pos_len_summary$pos)
# fitted_len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color
# fitted_len_pos_plot <- fitted_len_pos_plot + geom_line(data=fitted_pos_len_summary, aes(x=pos,y=fitte
# geom_point(data=fitted_pos_len_summary,aes(x=pos,y=fitted,group=stimlen,shape=stimlen)) + ggtitle(pas

nofrag_fitted_len_pos_plot <- plot_len_pos_obs_predicted(NoFragData,
                                        paste0(NoFragData$patient[1]),
                                        "LPFitted",
                                        NULL,
                                        palette_values,
                                        shape_values,
                                        obs_linetypes,
                                        pred_linetypes = c("longdash")
                                        )
```

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"no_fragments_percent_preserved_by_length_pos_wfit.png"),plot=no
nofrag_fitted_len_pos_plot
```



back to full data

```
results_report_DF <- AddReportLine(results_report_DF,"min preserved",min_preserved)
results_report_DF <- AddReportLine(results_report_DF,"max preserved",max_preserved)
print(sprintf("Min/max preserved range: %.2f - %.2f",min_preserved,max_preserved))
```

```
## [1] "Min/max preserved range: 0.81 - 1.00"
```

```
# find the average difference between lengths and the average difference
# between positions taking the other factor into account

# choose fitted or raw -- we use fitted values because they are smoothed averages
# that take into account the influence of the other variable
table_to_use <- fitted_pos_len_table
# take away column with length information
table_to_use <- table_to_use[,2:ncol(table_to_use)]

# take averages for positions
# don't want downward estimates influenced by return upward of U
# therefore, for downward influence, use only the values before the min
# take the difference between each value (differences between position proportion correct) **NOTE** pro
# average the difference in probabilities

# in case min is close to the end or we are not using a min (for non-U-shaped)
# functions, we need to get differences _first_ and then average those (e.g.
```

```r
# if there is an effect of length and we just average position data,
# later positions) will have a lower average (because of the length effect)
# even if all position functions go upward

table_pos_diffs <- t(diff(t(as.matrix(table_to_use))))
ncol_pos_diff_matrix <- ncol(table_pos_diffs)
first_col_mean <- mean(as.numeric(unlist(table_pos_diffs[,1])))
potential_u_shape <- 0
if(first_col_mean < 0){ # downward, so potential U-shape
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs >= 0] <- NA
  potential_u_shape <- 1
}else{
  # upward - use the positive values
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs <= 0] <- NA
}
average_pos_diffs <- apply(filtered_table_pos_diffs,2,mean,na.rm=TRUE)
OA_mean_pos_diff <- mean(average_pos_diffs,na.rm=TRUE)

table_len_diffs <- diff(as.matrix(table_to_use))
average_len_diffs <- apply(table_len_diffs,1,mean,na.rm=TRUE)
OA_mean_len_diff <- mean(average_len_diffs,na.rm=TRUE)
CurrentLabel<-"mean change in probability for each additional length: "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional length: "
```

```r
print(OA_mean_len_diff)
```

```
## [1] -0.007176548
```

```r
results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,OA_mean_len_diff)

CurrentLabel<-"mean change in probability for each additional position (excluding  U-shape): "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional position (excluding  U-shape): "
```

```r
print(OA_mean_pos_diff)
```

```
## [1] 0.004265134
```

```r
results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,OA_mean_pos_diff)
```

```r
if(potential_u_shape){ # downward, so potential U-shape
  # take only positive values from the table
  filtered_pos_upward_u<-table_pos_diffs
  filtered_pos_upward_u[table_pos_diffs <= 0] <- NA
  average_pos_u_diffs <- apply(filtered_pos_upward_u,
                               2,mean,na.rm=TRUE)
  OA_mean_pos_u_diff <- mean(average_pos_u_diffs,na.rm=TRUE)
  if(is.nan(OA_mean_pos_u_diff) | (OA_mean_pos_u_diff <= 0)){
    print("No U-shape in this participant")
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

    CurrentLabel<-"Average upward change after U minimum"
    print(CurrentLabel)
    print(OA_mean_pos_u_diff)
    results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, OA_mean_pos_u_diff)

    CurrentLabel<-"Proportion of average downward change"
    prop_ave_down <- abs(OA_mean_pos_u_diff/OA_mean_pos_diff)
    results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, prop_ave_down)
  }
}else{
  print("No U-shape in this participant")
  results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
}
```

```
## [1] "No U-shape in this participant"
```

```
if(potential_u_shape){
  n_rows<-nrow(table_to_use)
  downward_dist <- numeric(n_rows)
  upward_dist <- numeric(n_rows)
  for(i in seq(1,n_rows)){
    current_row <- as.numeric(unlist(table_to_use[i,1:9]))
    current_row <- current_row[!is.na(current_row)]
    current_row_len <- length(current_row)
    row_min <- min(current_row,na.rm=TRUE)
    min_pos <- which(current_row == row_min)
    left_max <- max(current_row[1:min_pos],na.rm=TRUE)
    right_max <- max(current_row[min_pos:current_row_len])
    left_diff <- left_max - row_min
    right_diff <- right_max - row_min
    downward_dist[i]<-left_diff
    upward_dist[i]<-right_diff
  }
  print("differences from left max to min for each row: ")
  print(downward_dist)
  print("differences from min to right max for each row: ")
  print(upward_dist)

  # Use the max return upward (min of upward_dist) and then the
  # downward distance from the left for the same row

  print(" ")
  biggest_return_upward_row <- which(upward_dist == max(upward_dist))
  CurrentLabel <- "row with biggest return upward"
  print(CurrentLabel)
  print(biggest_return_upward_row)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, biggest_return_upward_row)

  print(" ")
  CurrentLabel<-"downward distance for row with the largest upward value"
  print(CurrentLabel)
  print(downward_dist[biggest_return_upward_row])
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel, downward_dist[biggest_return_upwa

  CurrentLabel<-"return upward value"
  print(upward_dist[biggest_return_upward_row])
   results_report_DF <- AddReportLine(results_report_DF,
                                      CurrentLabel,
                                      upward_dist[biggest_return_upward_row])

  print(" ")
  # percentage return upward
  percentage_return_upward <- upward_dist[biggest_return_upward_row]/downward_dist[biggest_return_upward
  CurrentLabel <- "Return upward as a proportion of the downward distance:"
  print(CurrentLabel)
  print(percentage_return_upward)
  results_report_DF <- AddReportLine(results_report_DF, CurrentLabel,
                        percentage_return_upward)
}else{
  print("no U-shape in this participant")
}
```

```
## [1] "no U-shape in this participant"
```

```
FLPModelEquations<-c(
  # models with log frequency, but no three-way interactions (due to difficulty interpreting and small
            "preserved ~ stimlen*log_freq",
            "preserved ~ stimlen+log_freq",
            "preserved ~ pos*log_freq",
            "preserved ~ pos+log_freq",
            "preserved ~ stimlen*log_freq + pos*log_freq",
            "preserved ~ stimlen*log_freq + pos",
            "preserved ~ stimlen + pos*log_freq",
            "preserved ~ stimlen + pos + log_freq",
            "preserved ~ (I(pos^2)+pos)*log_freq",
            "preserved ~ stimlen*log_freq + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen*log_freq + I(pos^2) + pos",
            "preserved ~ stimlen + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen + I(pos^2) + pos + log_freq",

            # models without frequency
            "preserved ~ 1",
            "preserved ~ stimlen",
            "preserved ~ pos",
            "preserved ~ stimlen + pos",
            "preserved ~ stimlen*pos",
            "preserved ~ I(pos^2)+pos",
            "preserved ~ stimlen + I(pos^2) + pos",
            "preserved ~ stimlen * (I(pos^2) + pos)"
)

FLPRes<-TestModels(FLPModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.49348     -0.05382      0.04990      0.10217
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2677  AIC: 2794
## log likelihood:  -1338.357
## Nagelkerke R2:  0.01029481
## % pres/err predicted correctly:  -745.1663
## % of predictable range [ (model-null)/(1-null) ]:  0.005339272
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.13262      0.03566      0.11356
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2679  AIC: 2795
## log likelihood:  -1339.579
## Nagelkerke R2:  0.00908539
## % pres/err predicted correctly:  -745.7044
## % of predictable range [ (model-null)/(1-null) ]:  0.004621922
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.49776     -0.02992      0.10207
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2680  AIC: 2795
## log likelihood:  -1340.227
## Nagelkerke R2:  0.008444544
## % pres/err predicted correctly:  -745.8696
## % of predictable range [ (model-null)/(1-null) ]:  0.004401687
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.48429          -0.05386           0.19633           0.04989          -0.01203
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4364 Residual
## Null Deviance:       2697
## Residual Deviance: 2676  AIC: 2796
## log likelihood:  -1338.096
## Nagelkerke R2:  0.01055237
## % pres/err predicted correctly:  -745.1082
## % of predictable range [ (model-null)/(1-null) ]:  0.005416746
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    2.385731    -0.050576    -0.006412     0.103859     0.102336
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4364 Residual
## Null Deviance:       2697
## Residual Deviance: 2676  AIC: 2796
## log likelihood:  -1338.182
## Nagelkerke R2:  0.01046786
## % pres/err predicted correctly:  -745.1134
## % of predictable range [ (model-null)/(1-null) ]:  0.005409828
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     2.491932     -0.052692      0.047532      0.118851     -0.004442
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4364 Residual
## Null Deviance:       2697
## Residual Deviance: 2677  AIC: 2796
## log likelihood:  -1338.298
## Nagelkerke R2:  0.01035313
## % pres/err predicted correctly:  -745.1613
## % of predictable range [ (model-null)/(1-null) ]:  0.005345885
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)           stimlen          log_freq  stimlen:log_freq
##         2.48858          -0.02998           0.19625          -0.01203
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2680  AIC: 2797
## log likelihood:  -1339.966
## Nagelkerke R2:  0.00870256
## % pres/err predicted correctly:  -745.8183
## % of predictable range [ (model-null)/(1-null) ]:  0.00447012
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.14140       0.03264       0.13706      -0.00635
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2679  AIC: 2797
## log likelihood:  -1339.459
## Nagelkerke R2:  0.009204452
## % pres/err predicted correctly:  -745.6774
## % of predictable range [ (model-null)/(1-null) ]:  0.004657991
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          2.36995          -0.05042           0.19991          -0.00678           0.10693
## stimlen:log_freq
##         -0.01246
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4363 Residual
## Null Deviance:       2697
## Residual Deviance: 2676  AIC: 2798
## log likelihood:  -1337.901
## Nagelkerke R2:  0.01074518
## % pres/err predicted correctly:  -745.0509
## % of predictable range [ (model-null)/(1-null) ]:  0.005493145
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.484373         -0.053560          0.196538          0.049239         -0.011443
##     log_freq:pos
##        -0.001272
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4363 Residual
## Null Deviance:       2697
## Residual Deviance: 2676  AIC: 2798
## log likelihood:  -1338.092
## Nagelkerke R2:  0.01055653
## % pres/err predicted correctly:  -745.1094
## % of predictable range [ (model-null)/(1-null) ]:  0.005415133
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          2.407215          -0.049878          -0.004576           0.087544           0.186175
## I(pos^2):log_freq       pos:log_freq
##          0.004826          -0.046470
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4362 Residual
## Null Deviance:       2697
## Residual Deviance: 2675  AIC: 2799
## log likelihood:  -1337.734
## Nagelkerke R2:  0.01091081
## % pres/err predicted correctly:  -744.937
## % of predictable range [ (model-null)/(1-null) ]:  0.005644922
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##          2.039007          -0.007370           0.096978           0.200767           0.004565
##      pos:log_freq
##         -0.046341
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4363 Residual
## Null Deviance:       2697
## Residual Deviance: 2677  AIC: 2800
## log likelihood:  -1338.741
## Nagelkerke R2:  0.009914378
## % pres/err predicted correctly:  -745.3994
## % of predictable range [ (model-null)/(1-null) ]:  0.00502854
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          2.401943          -0.051031           0.294305          -0.004246           0.087138
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.014383           0.005608          -0.049195
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4361 Residual
## Null Deviance:       2697
## Residual Deviance: 2675  AIC: 2801
## log likelihood:  -1337.418
## Nagelkerke R2:  0.01122302
## % pres/err predicted correctly:  -744.8486
## % of predictable range [ (model-null)/(1-null) ]:  0.005762825
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.72172     -0.08530      0.04972
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2690  AIC: 2808
## log likelihood:  -1344.886
## Nagelkerke R2:  0.00382785
## % pres/err predicted correctly:  -747.6043
## % of predictable range [ (model-null)/(1-null) ]:  0.002089289
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.72578     -0.06147
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2693  AIC: 2809
## log likelihood:  -1346.749
## Nagelkerke R2:  0.001978647
## % pres/err predicted correctly:  -748.3192
## % of predictable range [ (model-null)/(1-null) ]:  0.001136294
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.11496     -0.13270     -0.08109      0.01533
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2689  AIC: 2809
## log likelihood:  -1344.479
## Nagelkerke R2:  0.004231169
## % pres/err predicted correctly:  -747.4736
## % of predictable range [ (model-null)/(1-null) ]:  0.00226352
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.621041    -0.082314    -0.006003     0.100267
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2689  AIC: 2810
## log likelihood:  -1344.731
## Nagelkerke R2:  0.003980993
## % pres/err predicted correctly:  -747.5731
## % of predictable range [ (model-null)/(1-null) ]:  0.002130889
## ***************************
## model index:  21
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)  
##        3.2773380        -0.1784781        -0.0109344        -0.1134543        -0.0007444  
##      stimlen:pos  
##        0.0362505  
## 
## Degrees of Freedom: 4368 Total (i.e. Null);  4363 Residual
## Null Deviance:       2697 
## Residual Deviance: 2687  AIC: 2812
## log likelihood:  -1343.604 
## Nagelkerke R2:  0.005099155 
## % pres/err predicted correctly:  -747.2577 
## % of predictable range [ (model-null)/(1-null) ]:  0.002551338 
## *************************** 
## model index:  14 
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)  
##        2.25  
## 
## Degrees of Freedom: 4368 Total (i.e. Null);  4368 Residual
## Null Deviance:       2697 
## Residual Deviance: 2697  AIC: 2813
## log likelihood:  -1348.741 
## Nagelkerke R2:  -4.820112e-16 
## % pres/err predicted correctly:  -749.1717 
## % of predictable range [ (model-null)/(1-null) ]:  0 
## *************************** 
## model index:  16 
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)          pos  
##     2.15566      0.02483  
## 
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697 
## Residual Deviance: 2696  AIC: 2814
## log likelihood:  -1348.216 
## Nagelkerke R2:  0.0005216602 
## % pres/err predicted correctly:  -749.0171 
## % of predictable range [ (model-null)/(1-null) ]:  0.000206091 
## *************************** 
## model index:  19 
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.01653     -0.01037      0.11353
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2695  AIC: 2815
## log likelihood:  -1347.744
## Nagelkerke R2:  0.0009903871
## % pres/err predicted correctly:  -748.885
## % of predictable range [ (model-null)/(1-null) ]:  0.0003821438
## ***************************
```

```
BestFLPModel<-FLPRes$ModelResult[[1]]
BestFLPModelFormula<-FLPRes$Model[[1]]

FLPAICSummary<-data.frame(Model=FLPRes$Model,
                          AIC=FLPRes$AIC,row.names=FLPRes$Model)
FLPAICSummary$DeltaAIC<-FLPAICSummary$AIC-FLPAICSummary$AIC[1]
FLPAICSummary$AICexp<-exp(-0.5*FLPAICSummary$DeltaAIC)
FLPAICSummary$AICwt<-FLPAICSummary$AICexp/sum(FLPAICSummary$AICexp)
FLPAICSummary$NagR2<-FLPRes$NagR2

FLPAICSummary <- merge(FLPAICSummary,FLPRes$CoefficientValues,
                             by='row.names',sort=FALSE)
FLPAICSummary <- subset(FLPAICSummary, select = -c(Row.names))

write.csv(FLPAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_freq_len_pos_model_summary.csv"),row.name
kable(FLPAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	[illegible]	I(pos^2)	I(pos^2):log_freq	stimlen:pos	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen + pos + log_freq	2794.237	0.000000	1.0000000	[illegible]	[illegible]	[illegible]	-0.0538153	0.1021703	NA	0.0499001	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1135579	NA	0.0356632	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0299247	0.1020714	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0538625	0.1963286	-0.0120264	0.0498938	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0505757	0.1023364	NA	0.1038594	NA	NA	-0.0064123	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0526925	0.1188514	NA	0.0475318	-0.0044416	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen * log_freq	2797.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0299770	0.1962502	-0.0120279	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	2797.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1370563	NA	0.0326395	-0.0063497	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	2797.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0504199	0.1999148	-0.0124617	0.1069255	NA	NA	-0.0067798	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	2797.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0535603	0.1965376	-0.0114429	0.0492388	NA	-0.0012722	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	2799.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0498781	0.1861751	NA	0.0875439	-0.0464696	NA	-0.0045760	0.0048255	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	2800.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.2007668	NA	0.0969777	-0.0463406	NA	-0.0073699	0.0045650	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	2800.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0510315	0.2943047	-0.0143830	0.0871376	NA	-0.0491950	-0.0042464	NA	0.0056083	NA	NA
preserved ~ stimlen + pos	2807.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0853037	NA	NA	0.0497225	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	2809.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0614693	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	2809.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1327015	NA	NA	-0.0810898	NA	NA	NA	NA	NA	0.0153260	NA
preserved ~ stimlen + I(pos^2) + pos	2809.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0823139	NA	NA	0.1002675	NA	NA	-0.0060029	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	2811.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1784781	NA	NA	-0.1134543	NA	NA	-0.0109344	NA	NA	0.0362505	-0.0007444
preserved ~ 1	2812.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	2814.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	0.0248264	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	2815.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	0.1135261	NA	NA	-0.0103730	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ stimlen + pos + log_freq"
```

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.49348     -0.05382      0.04990      0.10217
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2677  AIC: 2794
```

```r
# do a median split on frequency to plot hf/ls effects (analysis is continuous)
median_freq <- median(PosDat$log_freq)
PosDat$freq_bin[PosDat$log_freq >= median_freq]<-"hf"
PosDat$freq_bin[PosDat$log_freq <  median_freq]<-"lf"
```

```r
PosDat$FLPFitted<-fitted(BestFLPModel)

HFDat <- PosDat[PosDat$freq_bin == "hf",]
LFDat <- PosDat[PosDat$freq_bin == "lf",]

HF_Plot <- plot_len_pos_obs_predicted(HFDat,paste0(CurPat," - High frequency"),"FLPFitted",c(min_preser
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace the existing
## scale.
```

```r
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

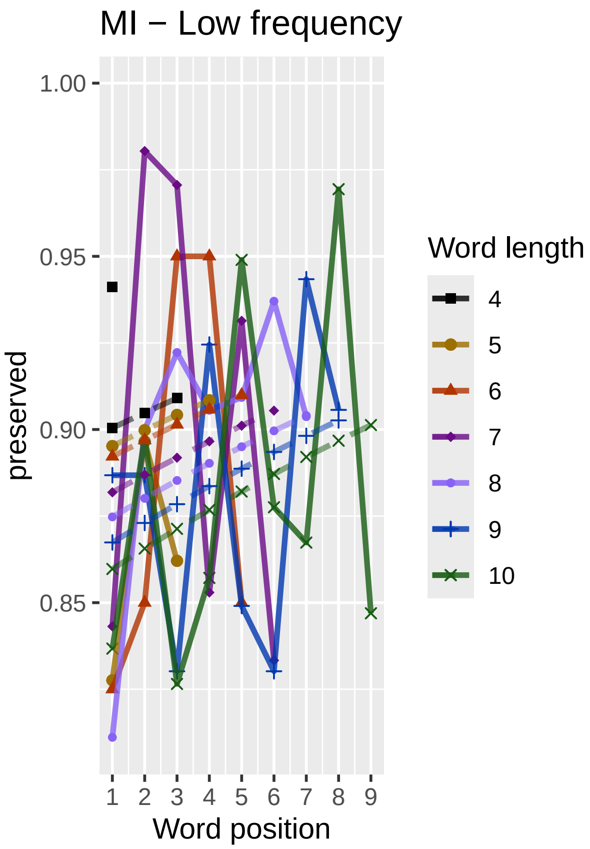
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale for y is already present. Adding another scale for y, which will replace the existing
## scale.
```

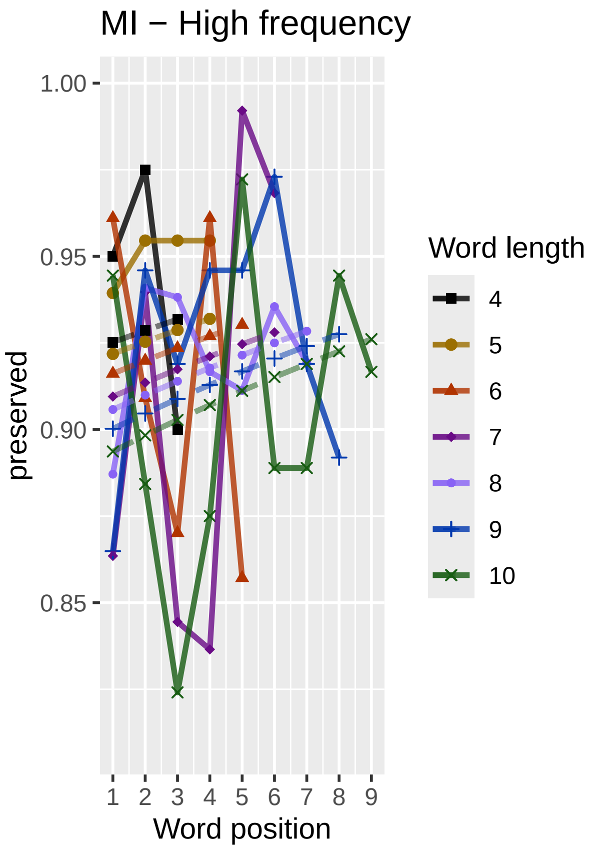
```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```r
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_frequency_effect_length_pos_wfit.png"),device="png",unit="cm"
print(Both_Plots)
```





```r
# only main effects
MEModelEquations<-c(
  "preserved ~ CumPres",
  "preserved ~ CumErr",
  "preserved ~ (I(pos^2)+pos)",
  "preserved ~ pos",
  "preserved ~ stimlen",
  "preserved ~ 1"
)
MERes<-TestModels(MEModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3960      -0.4006
```

```
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2666  AIC: 2777
## log likelihood:  -1333.03
## Nagelkerke R2:  0.01555655
## % pres/err predicted correctly:  -742.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008886832
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.05762      0.07958
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2688  AIC: 2806
## log likelihood:  -1344.188
## Nagelkerke R2:  0.00452011
## % pres/err predicted correctly:  -747.5429
## % of predictable range [ (model-null)/(1-null) ]:  0.002171177
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.72578     -0.06147
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2693  AIC: 2809
## log likelihood:  -1346.749
## Nagelkerke R2:  0.001978647
## % pres/err predicted correctly:  -748.3192
## % of predictable range [ (model-null)/(1-null) ]:  0.001136294
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        2.25
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4368 Residual
## Null Deviance:       2697
## Residual Deviance: 2697  AIC: 2813
## log likelihood:  -1348.741
## Nagelkerke R2:  -4.820112e-16
## % pres/err predicted correctly:  -749.1717
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.15566      0.02483
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2696  AIC: 2814
## log likelihood:  -1348.216
## Nagelkerke R2:  0.0005216602
## % pres/err predicted correctly:  -749.0171
## % of predictable range [ (model-null)/(1-null) ]:  0.000206091
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.01653     -0.01037      0.11353
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2695  AIC: 2815
## log likelihood:  -1347.744
## Nagelkerke R2:  0.0009903871
## % pres/err predicted correctly:  -748.885
## % of predictable range [ (model-null)/(1-null) ]:  0.0003821438
## ***************************
```

```r
BestMEModel<-MERes$ModelResult[[ BestModelIndexL1 ]]
BestMEModelFormula<-MERes$Model[[BestModelIndexL1]]

MEAICSummary<-data.frame(Model=MERes$Model,
                       AIC=MERes$AIC,row.names=MERes$Model)
MEAICSummary$DeltaAIC<-MEAICSummary$AIC-MEAICSummary$AIC[1]
MEAICSummary$AICexp<-exp(-0.5*MEAICSummary$DeltaAIC)
MEAICSummary$AICwt<-MEAICSummary$AICexp/sum(MEAICSummary$AICexp)
MEAICSummary$NagR2<-MERes$NagR2

MEAICSummary <- merge(MEAICSummary,MERes$CoefficientValues,
                          by='row.names',sort=FALSE)
MEAICSummary <- subset(MEAICSummary, select = -c(Row.names))
```

```r
write.csv(MEAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_model_summary.csv"),row.names
kable(MEAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2777.454	0.00000	1e+00	0.9999993	0.0155566	2.395972	NA	-0.4006481	NA	NA	NA
preserved ~ CumPres	2806.341	28.88669	5e-07	0.0000005	0.0045202	1.057623	0.0795814	NA	NA	NA	NA
preserved ~ stimlen	2809.088	31.63403	1e-07	0.0000001	0.0019788	2.725775	NA	NA	NA	NA	-0.0614693
preserved ~ 1	2812.599	35.14471	0e+00	0.0000000	0.0000000	2.249775	NA	NA	NA	NA	NA
preserved ~ pos	2814.161	36.70653	0e+00	0.0000000	0.0005217	2.155663	NA	NA	NA	0.0248264	NA
preserved ~ (I(pos^2) + pos)	2815.380	37.92594	0e+00	0.0000000	0.0009904	2.016526	NA	NA	-0.010373	0.1135261	NA

```r
if(DoSimulations){
      BestMEModelFormulaRnd <- BestMEModelFormula
    if(grepl("CumPres",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumPres","RndCumPres",BestMEModelFormulaRnd)
    }else if(grepl("CumErr",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumErr","RndCumErr",BestMEModelFormulaRnd)
    }

    RndModelAIC<-numeric(length=RandomSamples)
    for(rindex in seq(1,RandomSamples)){
    # Shuffle cumulative values
      PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
      PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
      BestModelRnd <- glm(as.formula(BestMEModelFormulaRnd),
                          family="binomial",data=PosDat)
      RndModelAIC[rindex] <- BestModelRnd$aic
    }
    ModelNames<-c(paste0("***",BestMEModelFormula),
                  rep(BestMEModelFormulaRnd,RandomSamples))
    AICValues <- c(BestMEModel$aic,RndModelAIC)
    BestMEModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
    BestMEModelRndDF <- BestMEModelRndDF %>% arrange(AIC)
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random average"),
                                         AIC=c(mean(RndModelAIC))))
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random SD"),
                                         AIC=c(sd(RndModelAIC))))

    write.csv(BestMEModelRndDF,
              paste0(TablesDir,CurPat,"_",CurTask,
                     "_best_main_effects_model_with_random_cum_term.csv"),
              row.names = FALSE)
}
```

```r
syll_component_summary <- PosDat %>%
  group_by(syll_component) %>% summarise(MeanPres = mean(preserved),
                                         N = n())
write.csv(syll_component_summary,paste0(TablesDir,CurPat,"_",CurTask,"_syllable_component_summary.csv")
kable(syll_component_summary)
```

syll_component	MeanPres	N
1	0.9027875	574
O	0.8729435	2018
P	0.7647059	34
S	0.7872200	253
V	0.9713870	1490

```
#  main effects models for data without satellite positions

keep_components = c("O","V","1")
OV1Data <- PosDat[PosDat$syll_component %in% keep_components,]
OV1Data <- OV1Data %>% select(stim_number,
                            stimlen,stim,pos,
                            preserved,syll_component)
OV1Data$CumPres <- CalcCumPres(OV1Data)
OV1Data$CumErr <- CalcCumErrFromPreserved(OV1Data)

SimpSyllMEAICSummary <- EvaluateSubsetData(OV1Data,MEModelEquations)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.4994      -0.4607
##
## Degrees of Freedom: 4081 Total (i.e. Null);  4080 Residual
## Null Deviance:       2361
## Residual Deviance: 2330  AIC: 2442
## log likelihood:  -1165.118
## Nagelkerke R2:  0.01705304
## % pres/err predicted correctly:  -637.6995
## % of predictable range [ (model-null)/(1-null) ]:  0.009491247
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##     2.81528     -0.05983
##
## Degrees of Freedom: 4081 Total (i.e. Null);  4080 Residual
## Null Deviance:       2361
## Residual Deviance: 2358  AIC: 2473
## log likelihood:  -1178.833
## Nagelkerke R2:  0.00181562
## % pres/err predicted correctly:  -643.1627
## % of predictable range [ (model-null)/(1-null) ]:  0.00101881
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.20971      0.06159
##
## Degrees of Freedom: 4081 Total (i.e. Null);  4080 Residual
## Null Deviance:       2361
## Residual Deviance: 2357  AIC: 2474
## log likelihood:  -1178.286
## Nagelkerke R2:  0.002425483
## % pres/err predicted correctly:  -643.095
## % of predictable range [ (model-null)/(1-null) ]:  0.001123749
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.352
##
## Degrees of Freedom: 4081 Total (i.e. Null);  4081 Residual
## Null Deviance:       2361
## Residual Deviance: 2361  AIC: 2475
## log likelihood:  -1180.462
## Nagelkerke R2:  -5.055769e-16
## % pres/err predicted correctly:  -643.8196
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.29978      0.01355
##
## Degrees of Freedom: 4081 Total (i.e. Null);  4080 Residual
## Null Deviance:       2361
## Residual Deviance: 2361  AIC: 2477
## log likelihood:  -1180.326
## Nagelkerke R2:  0.0001515188
## % pres/err predicted correctly:  -643.7882
## % of predictable range [ (model-null)/(1-null) ]:  4.878408e-05
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.09965     -0.01473      0.14032
##
## Degrees of Freedom: 4081 Total (i.e. Null);  4079 Residual
## Null Deviance:       2361
## Residual Deviance: 2359  AIC: 2478
## log likelihood:  -1179.477
## Nagelkerke R2:  0.001097659
## % pres/err predicted correctly:  -643.5551
## % of predictable range [ (model-null)/(1-null) ]:  0.0004102303
## ***************************
```

```
write.csv(SimpSyllMEAICSummary,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV1_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2442.243	0.00000	1e+00	0.9999995	0.0170532	2.499396	NA	-0.4607082	NA	NA	NA
preserved ~ stimlen	2472.703	30.46049	2e-07	0.0000002	0.0018152	2.815280	NA	NA	NA	NA	-0.0598329
preserved ~ CumPres	2473.543	31.29954	2e-07	0.0000002	0.0024255	2.209705	0.0615913	NA	NA	NA	NA
preserved ~ 1	2475.316	33.07276	1e-07	0.0000001	0.0000000	2.351755	NA	NA	NA	NA	NA
preserved ~ pos	2477.303	35.05951	0e+00	0.0000000	0.0001515	2.299776	NA	NA	NA	0.0135525	NA
preserved ~ (I(pos^2) + pos)	2477.905	35.66223	0e+00	0.0000000	0.0010977	2.099648	NA	NA	-0.0147256	0.1403201	NA

```
#  main effects models for data without satellite or coda positions
# (tradeoff -- takes away more complex segments, in addition to satellites, but
#  also reduces data)

keep_components = c("O","V")
OVData <- PosDat[PosDat$syll_component %in% keep_components,]
OVData <- OVData %>% select(stim_number,
                            stimlen,stim,pos,
                            preserved,syll_component)
OVData$CumPres <- CalcCumPres(OVData)
OVData$CumErr <- CalcCumErrFromPreserved(OVData)

SimpSyllMEAICSummary2<-EvaluateSubsetData(OVData,MEModelEquations)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.4588      -0.3585
##
## Degrees of Freedom: 3507 Total (i.e. Null);  3506 Residual
## Null Deviance:       2006
## Residual Deviance: 1996  AIC: 2079
## log likelihood:  -997.921
## Nagelkerke R2:  0.006661849
## % pres/err predicted correctly:  -541.5475
## % of predictable range [ (model-null)/(1-null) ]:  0.003959738
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.82793     -0.05884
##
## Degrees of Freedom: 3507 Total (i.e. Null);  3506 Residual
## Null Deviance:       2006
## Residual Deviance: 2003  AIC: 2087
## log likelihood:  -1001.671
## Nagelkerke R2:  0.001761209
## % pres/err predicted correctly:  -543.1803
## % of predictable range [ (model-null)/(1-null) ]:  0.0009621607
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)
##       2.373
##
## Degrees of Freedom: 3507 Total (i.e. Null);  3507 Residual
## Null Deviance:       2006
## Residual Deviance: 2006  AIC: 2089
## log likelihood:  -1003.017
## Nagelkerke R2:  -5.098409e-16
## % pres/err predicted correctly:  -543.7044
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.27281      0.05112
##
## Degrees of Freedom: 3507 Total (i.e. Null);  3506 Residual
## Null Deviance:       2006
## Residual Deviance: 2004  AIC: 2090
## log likelihood:  -1002.028
## Nagelkerke R2:  0.00129502
## % pres/err predicted correctly:  -543.3901
## % of predictable range [ (model-null)/(1-null) ]:  0.0005770597
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##    2.364282     0.002311
##
## Degrees of Freedom: 3507 Total (i.e. Null);  3506 Residual
## Null Deviance:       2006
## Residual Deviance: 2006  AIC: 2091
## log likelihood:  -1003.014
## Nagelkerke R2:  4.667497e-06
## % pres/err predicted correctly:  -543.7057
## % of predictable range [ (model-null)/(1-null) ]:  -2.382471e-06
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.13732     -0.01711      0.14969
##
## Degrees of Freedom: 3507 Total (i.e. Null);  3505 Residual
## Null Deviance:       2006
## Residual Deviance: 2004  AIC: 2091
## log likelihood:  -1001.963
## Nagelkerke R2:  0.001380006
## % pres/err predicted correctly:  -543.41
## % of predictable range [ (model-null)/(1-null) ]:  0.0005405116
## ****************************
```

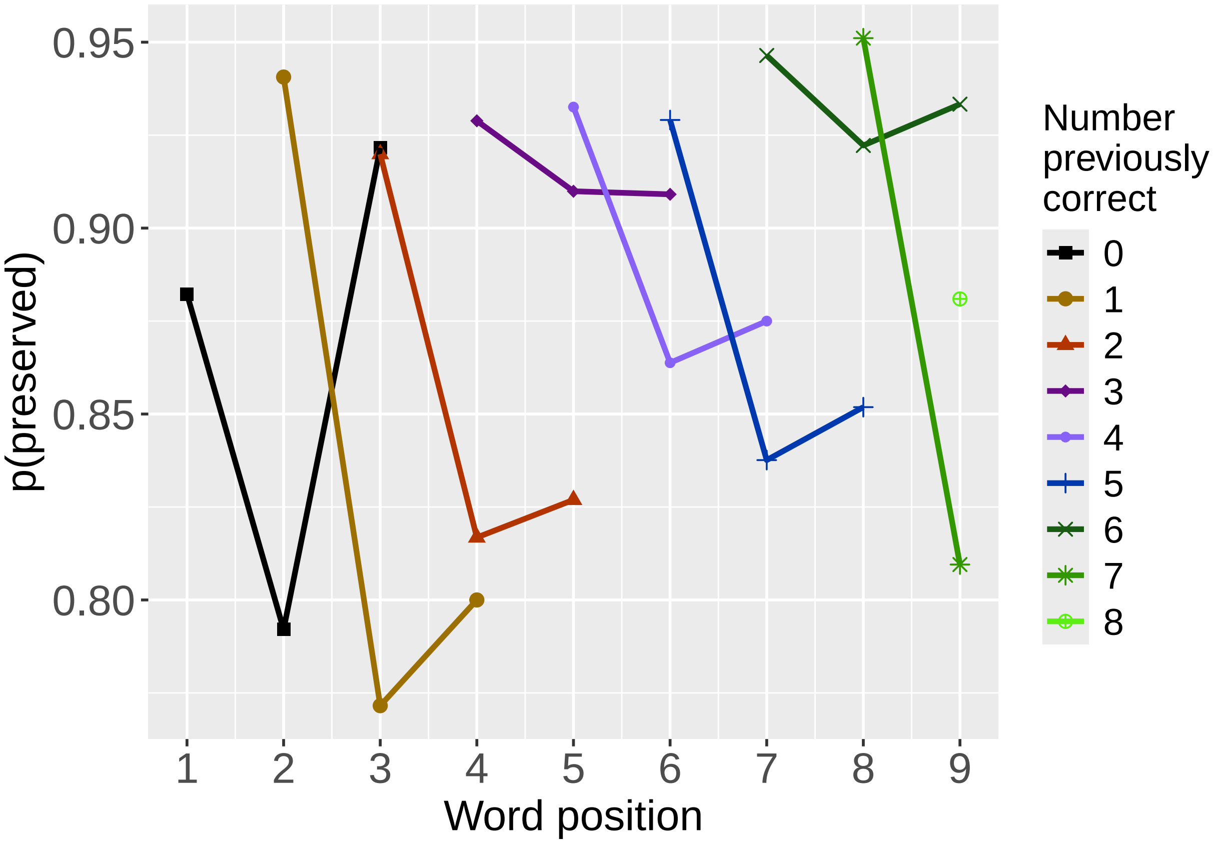
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2078.874	0.000000	1.0000000	0.9701105	0.0066618	2.458756	NA	-0.35854	NA	NA	NA
preserved ~ stimlen	2087.257	8.383564	0.0151193	0.0146674	0.0017612	2.827931	NA	NA	NA	NA	-0.0588446
preserved ~ 1	2088.965	10.091527	0.0064365	0.0062442	0.0000002	2.373149	NA	NA	NA	NA	NA
preserved ~ CumPres	2089.553	10.679174	0.0047979	0.0046544	0.0012952	2.272806	0.0511183	NA	NA	NA	NA
preserved ~ pos	2091.004	12.130771	0.0023219	0.0022525	0.0000047	2.364282	NA	NA	NA	0.0023113	NA
preserved ~ (I(pos^2) + pos)	2091.172	12.298775	0.0021348	0.0020710	0.0013802	2.137323	NA	NA	-0.0171107	0.1496921	NA

```
# plot prev err and prev cor plots
PrevCorPlot<-PlotObsPreviousCorrect(PosDat,palette_values,shape_values)
```

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

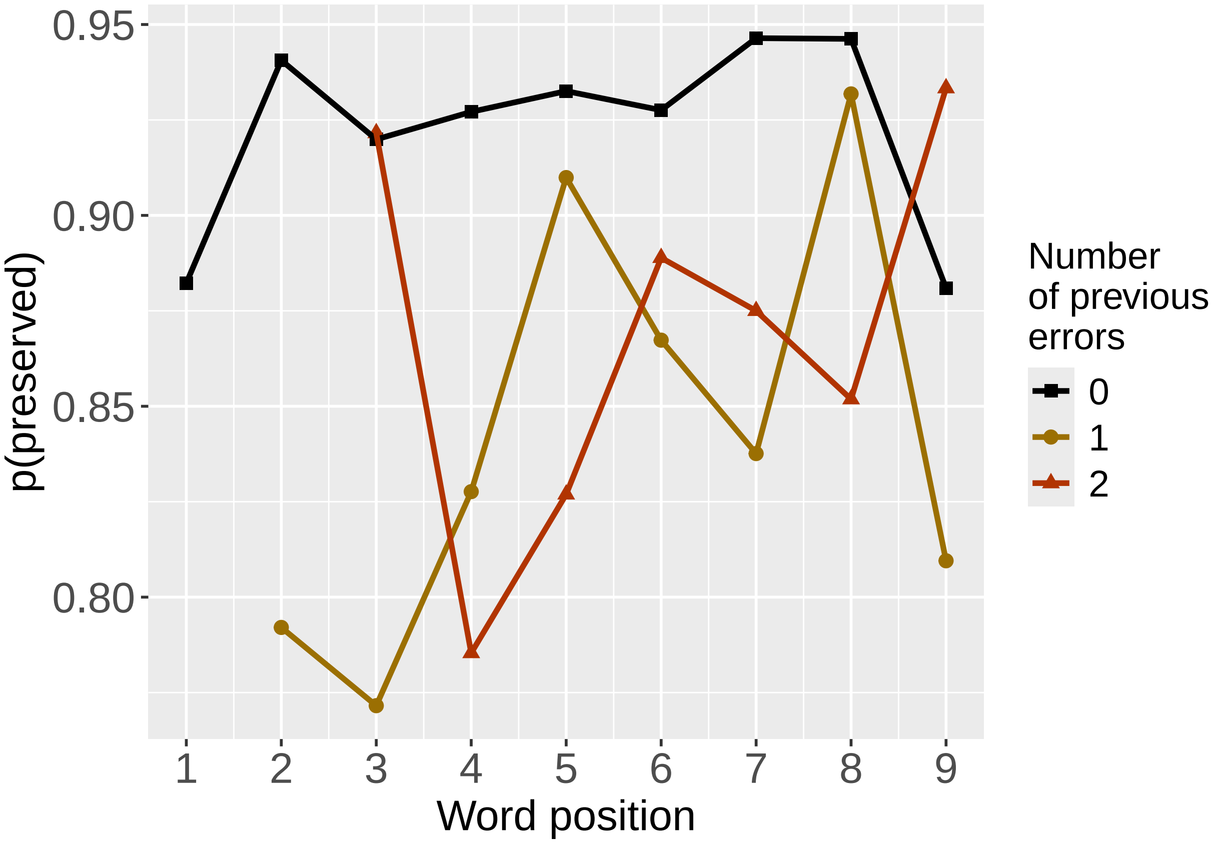
```
print(PrevCorPlot)
```



```
PrevErrPlot<-PlotObsPreviousError(PosDat,palette_values,shape_values)
```

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

```
print(PrevErrPlot)
```



```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevCorPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevCorPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

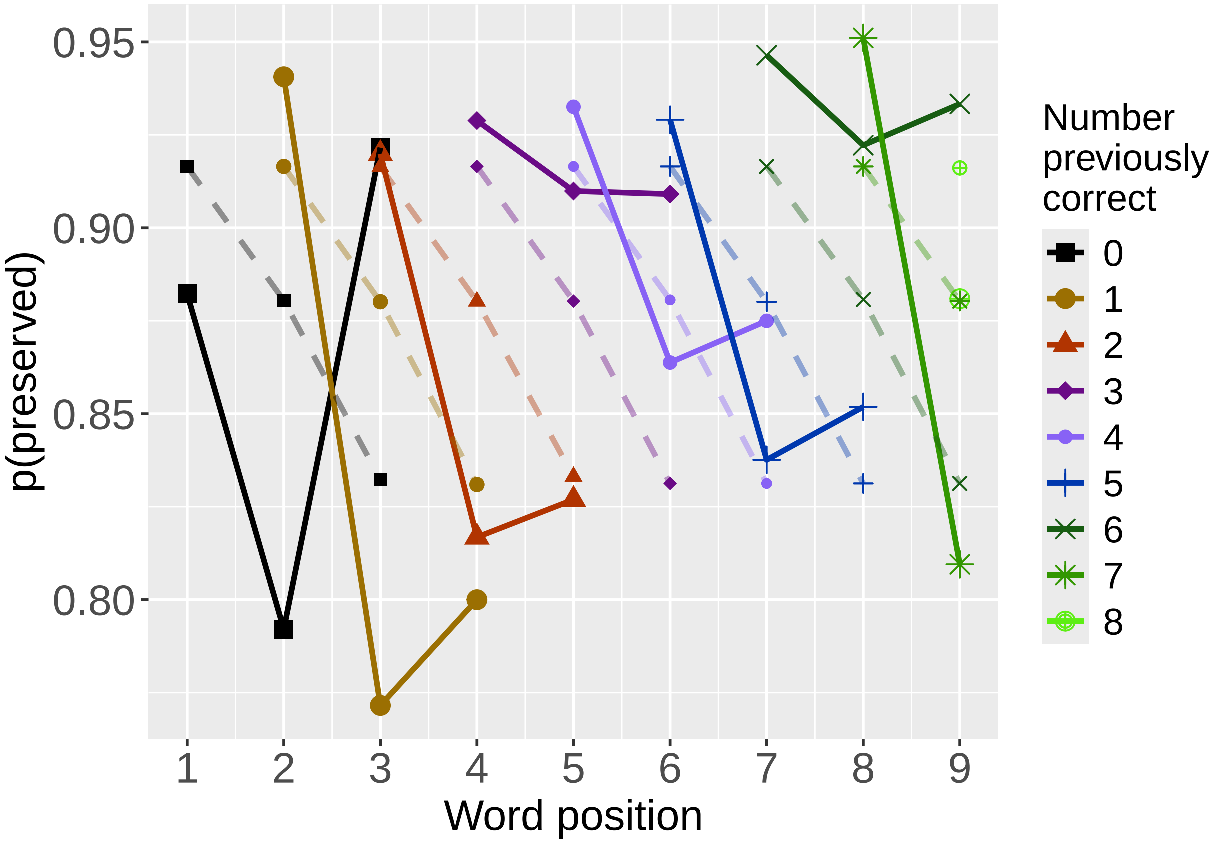
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

```
# plot prev err and prev cor with predicted values
MEModel<-MERes$ModelResult[[ BestModelIndexL1 ]]
PosDat$MEPred<-fitted(MEModel)

PrevCorPlot<-PlotObsPredPreviousCorrect(PosDat,"preserved","MEPred",palette_values,shape_values)
```

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

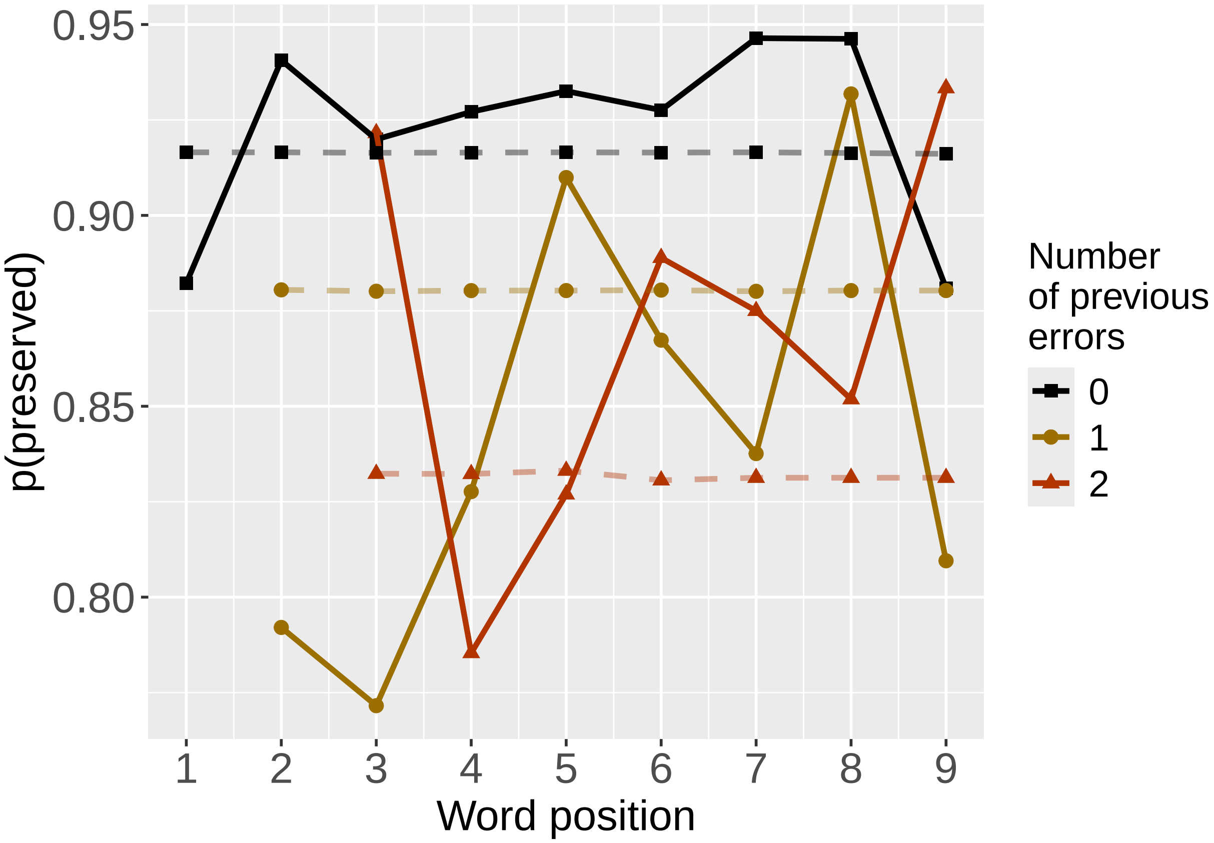
```
print(PrevCorPlot)
```



```
PrevErrPlot<-PlotObsPredPreviousError(PosDat,"preserved","MEPred",palette_values,shape_values)
```

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

```
print(PrevErrPlot)
```



```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"PrevCorPlot_ObsPredME")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevCorPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"PrevErrPlot_ObsPredME")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## Saving 6.5 x 4.5 in image
```

```
CumAICSummary <- NULL
```

```
########
# level 2 -- Add position squared (quadratic with position)
########

# After establishing the primary variable, see about additions

BestMEFactor<-BestMEModelFormula
OtherFactor<-"I(pos^2) + pos"
OtherFactorName<-"quadratic_by_pos"

if(!grepl("pos",BestMEFactor,fixed = TRUE)){ # if best model does not include pos
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor,OtherFactorName)
  CumAICSummary <- AICSummary
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##    1.954496    -0.515138    -0.009635     0.176146
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2653  AIC: 2769
## log likelihood:  -1326.567
## Nagelkerke R2:  0.02192398
## % pres/err predicted correctly:  -740.1612
## % of predictable range [ (model-null)/(1-null) ]:  0.01201125
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3960      -0.4006
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2666  AIC: 2777
## log likelihood:  -1333.03
## Nagelkerke R2:  0.01555655
## % pres/err predicted correctly:  -742.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008886832
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.01653     -0.01037      0.11353
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2695  AIC: 2815
## log likelihood:  -1347.744
## Nagelkerke R2:  0.0009903871
## % pres/err predicted correctly:  -748.885
## % of predictable range [ (model-null)/(1-null) ]:  0.0003821438
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	2769.391	0.000000	1.0000000	0.9825611	0.0219240	1.954496	-0.5151382	-0.0096355	0.1761465

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	2777.454	8.062918	0.0177484	0.0174389	0.0155566	2.395972	-0.4006481	NA	NA
preserved ~ I(posˆ2) + pos	2815.380	45.988859	0.0000000	0.0000000	0.0009904	2.016526	NA	-0.0103730	0.1135261

```
########
# level 2 -- Add length
########

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

if(!grepl(OtherFactor,BestMEFactor,fixed = TRUE)){
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor)
  CumAICSummary <- ConcatAndAddMissingColumns(CumAICSummary,AICSummary)
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3960      -0.4006
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2666  AIC: 2777
## log likelihood:  -1333.03
## Nagelkerke R2:  0.01555655
## % pres/err predicted correctly:  -742.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008886832
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.61751     -0.38669     -0.02928
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2665  AIC: 2778
## log likelihood:  -1332.604
## Nagelkerke R2:  0.01597653
## % pres/err predicted correctly:  -742.3575
## % of predictable range [ (model-null)/(1-null) ]:  0.009083406
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.72578     -0.06147
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2693  AIC: 2809
## log likelihood:  -1346.749
## Nagelkerke R2:  0.001978647
## % pres/err predicted correctly:  -748.3192
## % of predictable range [ (model-null)/(1-null) ]:  0.001136294
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	2777.454	0.0000000	1.0000000	0.5766734	0.0155566	2.395972	-0.4006481	NA
preserved ~ CumErr + stimlen	2778.072	0.6182643	0.7340837	0.4233266	0.0159765	2.617508	-0.3866916	-0.0292783
preserved ~ stimlen	2809.088	31.6340299	0.0000001	0.0000001	0.0019786	2.725775	NA	-0.0614693

```
########
# level 2 -- add cumulative preserved
########

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

if(!grepl(OtherFactor,BestMEFactor,fixed = TRUE)){
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor)
  CumAICSummary <- ConcatAndAddMissingColumns(CumAICSummary,AICSummary)
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     2.17693     -0.42195      0.09407
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2654  AIC: 2768
## log likelihood:  -1326.967
## Nagelkerke R2:  0.02152973
## % pres/err predicted correctly:  -740.1368
## % of predictable range [ (model-null)/(1-null) ]:  0.01204371
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3960      -0.4006
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2666  AIC: 2777
## log likelihood:  -1333.03
## Nagelkerke R2:  0.01555655
## % pres/err predicted correctly:  -742.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008886832
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.05762      0.07958
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2688  AIC: 2806
## log likelihood:  -1344.188
## Nagelkerke R2:  0.00452011
## % pres/err predicted correctly:  -747.5429
## % of predictable range [ (model-null)/(1-null) ]:  0.002171177
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	2768.037	0.000000	1.0000000	0.9910648	0.0215297	2.176935	-0.4219492	0.0940710
preserved ~ CumErr	2777.454	9.417574	0.0090157	0.0089352	0.0155566	2.395972	-0.4006481	NA
preserved ~ CumPres	2806.341	38.304266	0.0000000	0.0000000	0.0045201	2.057623	NA	0.0795814

```
########
# level 2 -- Add linear position (NOT quadratic)
########
```

```
BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

if(!grepl(OtherFactor,BestMEFactor,fixed = TRUE)){
  AICSummary<-TestLevel2Models(PosDat,BestMEFactor,OtherFactor)
  CumAICSummary <- ConcatAndAddMissingColumns(CumAICSummary,AICSummary)
  kable(AICSummary)
}
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     2.08286     -0.51602      0.09407
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2654   AIC: 2768
## log likelihood:  -1326.967
## Nagelkerke R2:  0.02152973
## % pres/err predicted correctly:  -740.1368
## % of predictable range [ (model-null)/(1-null) ]:  0.01204371
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3960      -0.4006
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2666   AIC: 2777
## log likelihood:  -1333.03
## Nagelkerke R2:  0.01555655
## % pres/err predicted correctly:  -742.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008886832
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     2.15566      0.02483
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2696  AIC: 2814
## log likelihood:  -1348.216
## Nagelkerke R2:  0.0005216602
## % pres/err predicted correctly:  -749.0171
## % of predictable range [ (model-null)/(1-null) ]:  0.000206091
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	2768.037	0.000000	1.0000000	0.9910648	0.0215297	2.082864	-0.5160202	0.0940710
preserved ~ CumErr	2777.454	9.417574	0.0090157	0.0089352	0.0155566	2.395972	-0.4006481	NA
preserved ~ pos	2814.161	46.124107	0.0000000	0.0000000	0.0005217	2.155663	NA	0.0248264

```r
CumAICSummary <- CumAICSummary %>% arrange(AIC)
BestModelFormulaL2 <- CumAICSummary$Model[BestModelIndexL2]
write.csv(CumAICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_plus_one_model_summary.csv")
kable(CumAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + CumPres	2768.037	0.0000000	1.0000000	0.9910648	0.0215297	2.176935	-0.4219492	NA	NA	NA	0.0940710
preserved ~ CumErr + pos	2768.037	0.0000000	1.0000000	0.9910648	0.0215297	2.082864	-0.5160202	NA	0.0940710	NA	NA
preserved ~ CumErr + I(pos^2) + pos	2769.390	1.0000000	1.0000000	0.9825611	0.0219241	1.0954496	-0.5151382	-0.0096355	0.1761465	NA	NA
preserved ~ CumErr	2777.458	4.0629176	0.0177484	0.0174389	0.0155566	2.395972	-0.4006481	NA	NA	NA	NA
preserved ~ CumErr	2777.454	1.0000000	1.0000000	0.5766734	0.0155566	2.395972	-0.4006481	NA	NA	NA	NA
preserved ~ CumErr	2777.454	9.4175736	0.0090157	0.0089352	0.0155566	2.395972	-0.4006481	NA	NA	NA	NA
preserved ~ CumErr	2777.454	9.4175736	0.0090157	0.0089352	0.0155566	2.395972	-0.4006481	NA	NA	NA	NA
preserved ~ CumErr + stimlen	2778.072	2.6182643	0.7340837	0.7423326	0.0159762	2.5617508	-0.3866916	NA	NA	-0.0292783	NA
preserved ~ CumPres	2806.343	38.3042658	0.0000000	0.0000000	0.0045202	2.1057623	NA	NA	NA	NA	0.0795814
preserved ~ stimlen	2809.088	41.6340299	0.0000001	0.0000001	0.0019782	2.6725775	NA	NA	NA	-0.0614693	NA
preserved ~ pos	2814.161	46.1241066	0.0000000	0.0000000	0.0005217	2.155663	NA	NA	0.0248264	NA	NA
preserved ~ I(pos^2) + pos	2815.380	45.9888593	0.0000000	0.0000000	0.0009902	2.4016526	NA	-0.0103730	0.1135261	NA	NA

```
# explore influence of frequency and length

if(grepl("stimlen",BestModelFormulaL2)){ # if length is already in the formula
  Level3ModelEquations<-c(
      BestModelFormulaL2, # best model from level 2
    "preserved ~ 1", # null model
      paste0(BestModelFormulaL2," + log_freq")
  )
}else if(grepl("log_freq",BestModelFormulaL2)){ # if log_freq is already in the formula
    Level3ModelEquations<-c(
      BestModelFormulaL2, # best model from level 2
    "preserved ~ 1", # null model
      paste0(BestModelFormulaL2," + stimlen")
  )
}else{
  Level3ModelEquations<-c(
      BestModelFormulaL2, # best model from level 2
    "preserved ~ 1", # null model
      paste0(BestModelFormulaL2," + log_freq"),
      paste0(BestModelFormulaL2," + stimlen"),
      paste0(BestModelFormulaL2," + stimlen + log_freq")
  )
}

Level3Res<-TestModels(Level3ModelEquations,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres      stimlen     log_freq
##     2.50310     -0.37329      0.11244     -0.04839      0.08872
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4364 Residual
## Null Deviance:       2697
## Residual Deviance: 2639  AIC: 2754
## log likelihood:  -1319.605
## Nagelkerke R2:  0.02876076
## % pres/err predicted correctly:  -737.2711
## % of predictable range [ (model-null)/(1-null) ]:  0.01586385
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq
##      2.1658      -0.3888       0.1001       0.0988
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2641  AIC: 2754
## log likelihood:  -1320.584
## Nagelkerke R2:  0.02780054
## % pres/err predicted correctly:  -737.6885
## % of predictable range [ (model-null)/(1-null) ]:  0.01530737
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres      stimlen
##     2.69729     -0.39202      0.11464     -0.07507
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2649  AIC: 2764
## log likelihood:  -1324.426
## Nagelkerke R2:  0.0240286
## % pres/err predicted correctly:  -739.0836
## % of predictable range [ (model-null)/(1-null) ]:  0.01344773
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     2.17693     -0.42195      0.09407
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2654  AIC: 2768
## log likelihood:  -1326.967
## Nagelkerke R2:  0.02152973
## % pres/err predicted correctly:  -740.1368
## % of predictable range [ (model-null)/(1-null) ]:  0.01204371
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##        2.25
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4368 Residual
## Null Deviance:       2697
## Residual Deviance: 2697  AIC: 2813
## log likelihood:  -1348.741
## Nagelkerke R2:  -4.820112e-16
## % pres/err predicted correctly:  -749.1717
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
BestModelL3<-Level3Res$ModelResult[[ BestModelIndexL3 ]]
BestModelFormulaL3<-Level3Res$Model[[BestModelIndexL3]]

AICSummary<-data.frame(Model=Level3Res$Model,
                       AIC=Level3Res$AIC,
                       row.names = Level3Res$Model)
AICSummary$DeltaAIC<-AICSummary$AIC-AICSummary$AIC[1]
AICSummary$AICexp<-exp(-0.5*AICSummary$DeltaAIC)
AICSummary$AICwt<-AICSummary$AICexp/sum(AICSummary$AICexp)
AICSummary$NagR2<-Level3Res$NagR2

AICSummary <- merge(AICSummary,Level3Res$CoefficientValues,
                          by='row.names',sort=FALSE)
AICSummary <- subset(AICSummary, select = -c(Row.names))

write.csv(AICSummary,paste0(TablesDir,CurPat,"_",CurTask,"_main_effects_plus_one_len_freq_summary.csv")

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres	log_freq	stimlen
preserved ~ CumErr + CumPres + stimlen + log_freq	2753.963	0.0000000	1.0000000	0.5555680	0.0287608	2.503098	-0.3732940	0.1124409	0.0887211	-0.0483859
preserved ~ CumErr + CumPres + log_freq	2754.431	0.4676647	0.7914945	0.4397290	0.0278002	2.165761	-0.3887930	0.1000548	0.0987958	NA
preserved ~ CumErr + CumPres + stimlen	2763.726	9.7628259	0.0075863	0.0042147	0.0240282	2.697286	-0.3920242	0.1146443	NA	-0.0750732
preserved ~ CumErr + CumPres	2768.037	14.0736704	0.0008789	0.0004883	0.0215292	2.176935	-0.4219492	0.0940710	NA	NA
preserved ~ 1	2812.599	58.6359547	0.0000000	0.0000000	0.0000000	2.249775	NA	NA	NA	NA

```
# BestModelIndex is set by the parameter file and is used to choose
# a model that is essentially equivalent to the lowest AIC model, but
# simpler (and with a slightly higher AIC value, so not in first position)
BestModelL3<-Level3Res$ModelResult[[ BestModelIndexL3 ]]
BestModelFormulaL3<-Level3Res$Model[BestModelIndexL3]

BestModel<-BestModelL3
BestModelDeletions<-arrange(dropterm(BestModel),desc(AIC))
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
# order by terms that increase AIC the most when they are the one dropped
BestModelDeletions
```

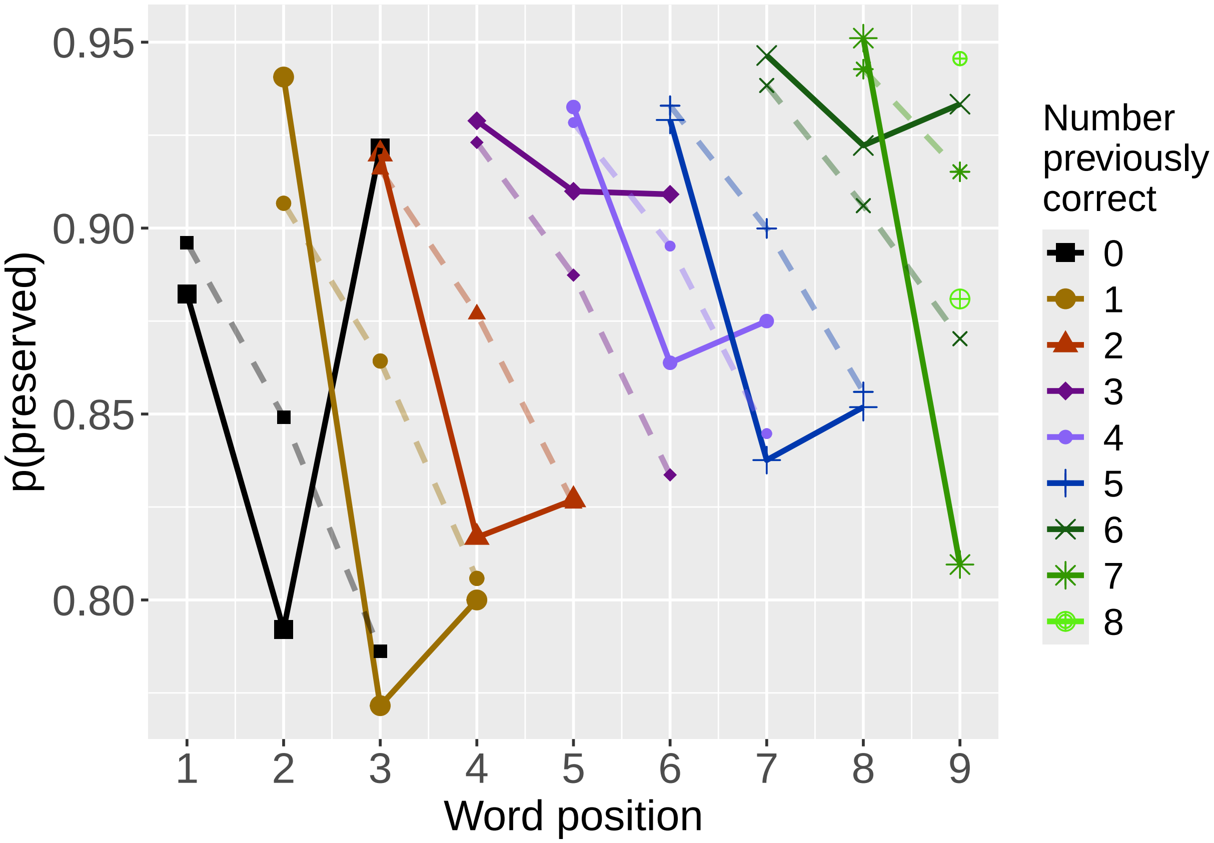
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + CumPres + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   2665.6 2778.3
## CumPres   1   2654.9 2767.7
## log_freq  1   2648.8 2761.6
## <none>        2639.2 2754.0
## stimlen   1   2641.2 2753.9
```

```
################################
# Single deletions from best model
################################

write.csv(BestModelDeletions,paste0(TablesDir,CurPat,"_",CurTask,"_best_model_single_term_deletions.csv
```

```
# plot prev err and prev cor with predicted values
PosDat$OAPred<-fitted(BestModel)

PrevCorPlot<-PlotObsPredPreviousCorrect(PosDat,"preserved","OAPred",palette_values,shape_values)
```

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

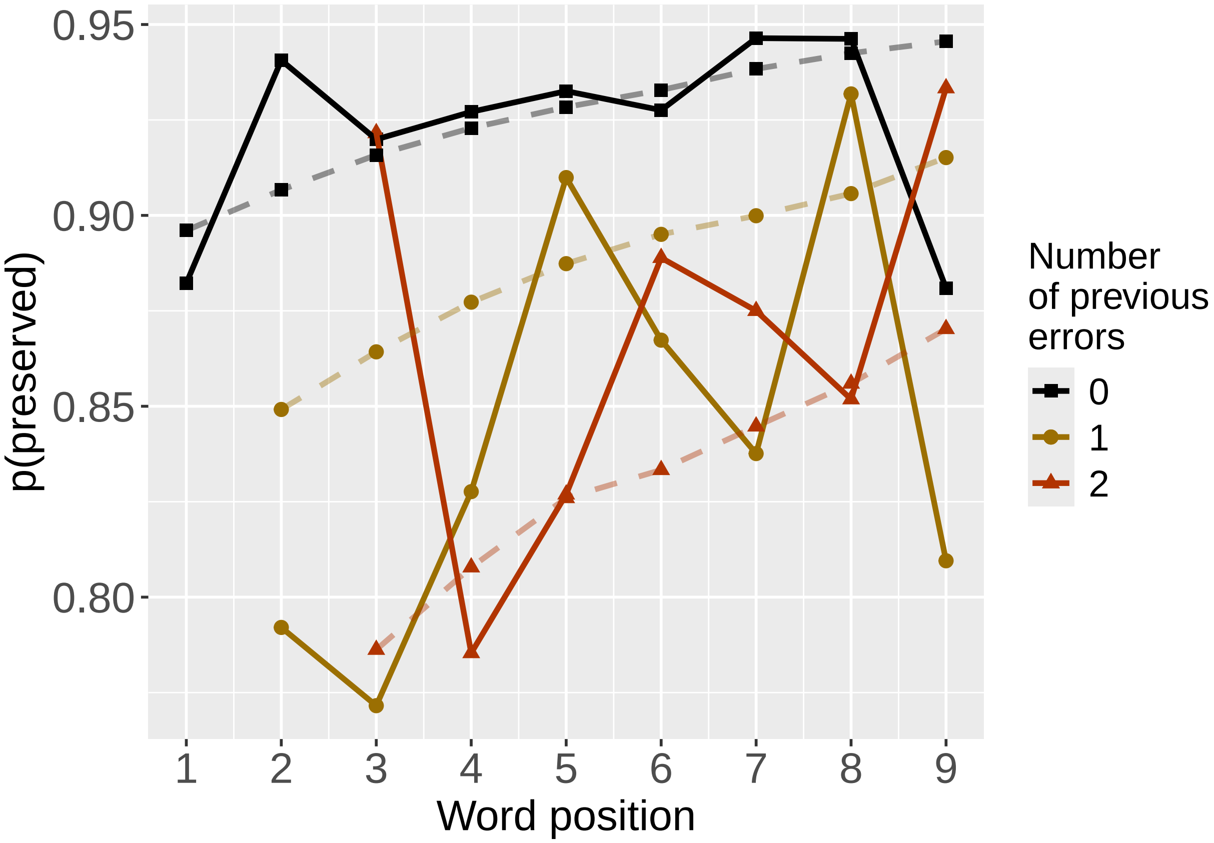
```
print(PrevCorPlot)
```



```
PrevErrPlot<-PlotObsPredPreviousError(PosDat,"preserved","OAPred",palette_values,shape_values)
```

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

```
print(PrevErrPlot)
```



```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_ObsPred_BestFullModel")
ggsave(filename=paste(PlotName,"_prev_correct.tif",sep=""),plot=PrevCorPlot,device="tiff",compression =
```

```
## Saving 6.5 x 4.5 in image
```

```
ggsave(filename=paste(PlotName,"_prev_error.tif",sep=""),plot=PrevErrPlot,device="tiff",compression = "
```

```
## Saving 6.5 x 4.5 in image
```

```
if(DoSimulations){
  BestModelFormulaL3Rnd <- BestModelFormulaL3
  if(grepl("CumPres",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumPres","RndCumPres",BestModelFormulaL3Rnd)
  }
  if(grepl("CumErr",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumErr","RndCumErr",BestModelFormulaL3Rnd)
  }

  RndModelAIC<-numeric(length=RandomSamples)
  for(rindex in seq(1,RandomSamples)){
  # Shuffle cumulative values
    PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
    PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
    BestModelRnd <- glm(as.formula(BestModelFormulaL3Rnd),
                        family="binomial",data=PosDat)
    RndModelAIC[rindex] <- BestModelRnd$aic
  }
  ModelNames<-c(paste0("***",BestModelFormulaL3),
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random average"),
                                     AIC=c(mean(RndModelAIC))))
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random SD"),
                                     AIC=c(sd(RndModelAIC))))
  write.csv(BestModelRndDF,paste0(TablesDir,CurPat,"_",CurTask,
                                  "_best_model_with_random_cum_term.csv"),
            row.names = FALSE)
}
```

```
# plot influence factors
num_models <- nrow(BestModelDeletions) - 1
# minus 1 because last
# model is always <none> and we don't want to include that

if(num_models > 4){
  last_model_num <- 4
}else{
  last_model_num <- num_models
}

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_FactorPlots")

FactorPlot<-PlotModelSetWithFreq(PosDat,FinalModelSet,
                                 palette_values,FinalModelSet,PptID=CurPat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3960      -0.4006
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4367 Residual
## Null Deviance:       2697
## Residual Deviance: 2666  AIC: 2777
## log likelihood:  -1333.03
## Nagelkerke R2:  0.01555655
## % pres/err predicted correctly:  -742.505
## % of predictable range [ (model-null)/(1-null) ]:  0.008886832
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     2.17693     -0.42195      0.09407
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4366 Residual
## Null Deviance:       2697
## Residual Deviance: 2654  AIC: 2768
## log likelihood:  -1326.967
## Nagelkerke R2:  0.02152973
## % pres/err predicted correctly:  -740.1368
## % of predictable range [ (model-null)/(1-null) ]:  0.01204371
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq
##      2.1658      -0.3888       0.1001       0.0988
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4365 Residual
## Null Deviance:       2697
## Residual Deviance: 2641  AIC: 2754
## log likelihood:  -1320.584
## Nagelkerke R2:  0.02780054
## % pres/err predicted correctly:  -737.6885
## % of predictable range [ (model-null)/(1-null) ]:  0.01530737
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq      stimlen
##     2.50310     -0.37329      0.11244      0.08872     -0.04839
##
## Degrees of Freedom: 4368 Total (i.e. Null);  4364 Residual
## Null Deviance:       2697
## Residual Deviance: 2639  AIC: 2754
## log likelihood:  -1319.605
## Nagelkerke R2:  0.02876076
## % pres/err predicted correctly:  -737.2711
## % of predictable range [ (model-null)/(1-null) ]:  0.01586385
## ***************************

## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'freq_bin'. You can override using the `.groups` argument.

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 9 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).
```

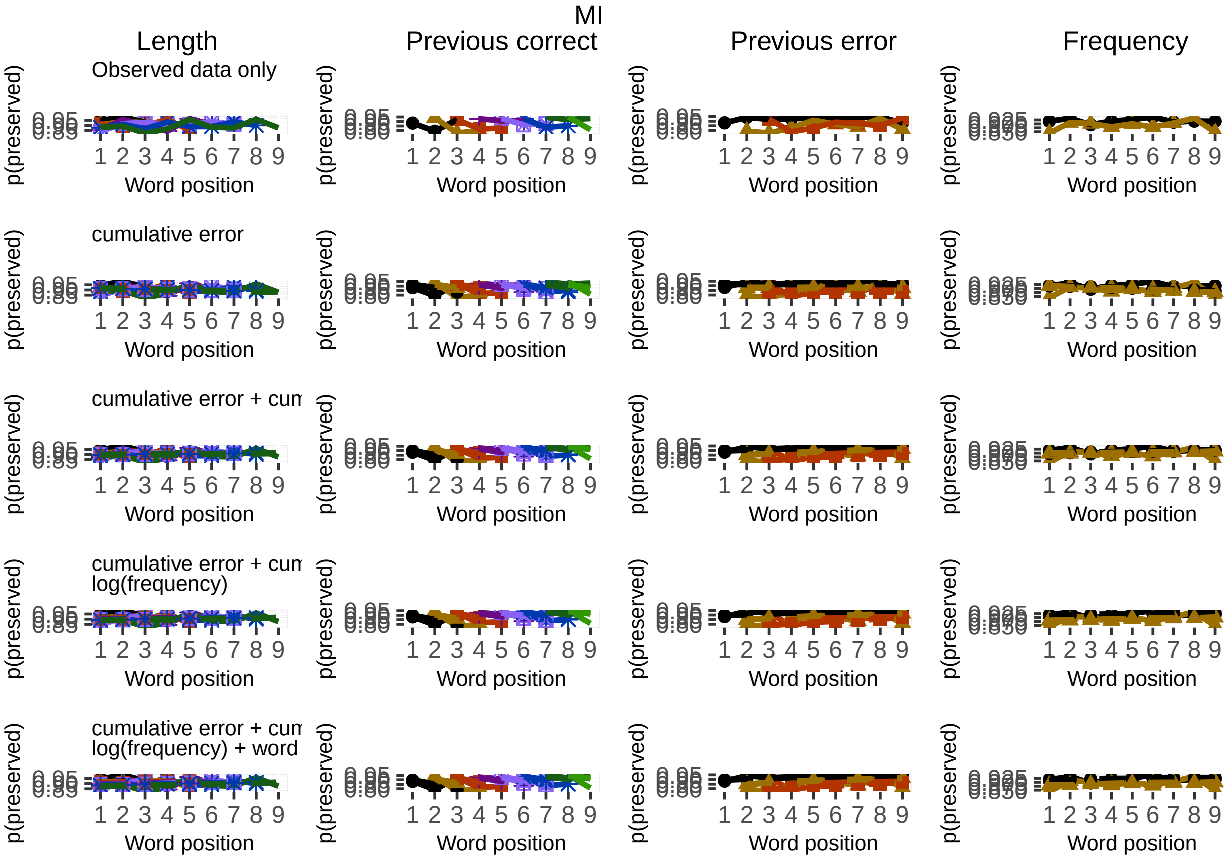
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## Warning: Removed 9 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 9 rows containing missing values or values outside the scale range (`geom_point()`).
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 6 rows containing missing values or values outside the scale range (`geom_point()`).
```

```r
ggsave(paste0(PlotName,".tif"),plot=FactorPlot,width = 360,height=400,units="mm",device="tiff",compress
# use \blandscape and \elandscape to make markdown plots landscape if needed
FactorPlot
```



```
DA.Result<-dominanceAnalysis(BestModel)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
DAContributionAverage<-ConvertDominanceResultToTable(DA.Result)
write.csv(DAContributionAverage,paste0(TablesDir,CurPat,"_",CurTask,"_dominance_analysis_table.csv"),ro
kable(DAContributionAverage)
```

	CumErr	CumPres	stimlen	log_freq
McFadden	0.0120183	0.0045418	0.0015219	0.0056268
SquaredCorrelation	0.0076718	0.0028967	0.0009733	0.0035944
Nagelkerke	0.0076718	0.0028967	0.0009733	0.0035944
Estrella	0.0077650	0.0029359	0.0009822	0.0036341

```
deviance_differences_df<-GetDevianceSet(FinalModelSet,PosDat)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
write.csv(deviance_differences_df,paste0(TablesDir,CurPat,"_",CurTask,"_deviance_differences_table.csv"),row.names = FALSE)
deviance_differences_df
```

```
##                                                                       model deviance
## CumErr + CumPres + log_freq + stimlen CumErr + CumPres + log_freq + stimlen 2639.211
## CumErr + CumPres + log_freq                     CumErr + CumPres + log_freq 2641.169
## CumErr + CumPres                                           CumErr + CumPres 2653.935
## CumErr                                                               CumErr 2666.060
## null                                                                   null 2697.483
##                                       deviance_explained percent_explained
## CumErr + CumPres + log_freq + stimlen           58.27180          2.160229
## CumErr + CumPres + log_freq                     56.31373          2.087640
## CumErr + CumPres                                43.54781          1.614387
## CumErr                                          31.42240          1.164879
## null                                             0.00000          0.000000
##                                       percent_of_explained_deviance increment_in_explained
## CumErr + CumPres + log_freq + stimlen                     100.00000               3.360234
## CumErr + CumPres + log_freq                                96.63977              21.907534
## CumErr + CumPres                                           74.73223              20.808380
## CumErr                                                     53.92385              53.923851
## null                                                             NA               0.000000
```

```
kable(deviance_differences_df[,2:3], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

```
kable(deviance_differences_df[,c(4:6)], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

```
NagContributions <- t(DAContributionAverage[3,])
NagTotal<-sum(NagContributions)
NagPercents <- NagContributions / NagTotal
NagResultTable <- data.frame(NagContributions = NagContributions, NagPercents = NagPercents)
```

	deviance	deviance_explained
CumErr + CumPres + log_freq + stimlen	2639.211	58.27180
CumErr + CumPres + log_freq	2641.169	56.31373
CumErr + CumPres	2653.935	43.54781
CumErr	2666.060	31.42240
null	2697.483	0.00000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + CumPres + log_freq + stimlen	2.160229	100.00000	3.360234
CumErr + CumPres + log_freq	2.087640	96.63977	21.907535
CumErr + CumPres	1.614387	74.73223	20.808380
CumErr	1.164879	53.92385	53.923851
null	0.000000	NA	0.000000

```
names(NagResultTable)<-c("NagContributions","NagPercents")
write.csv(NagResultTable,paste0(TablesDir,CurPat,"_",CurTask,"_nagelkerke_contributions_table.csv"),row.names = TRUE)
NagPercents
```

```
##          Nagelkerke
## CumErr    0.5068504
## CumPres   0.1913781
## stimlen   0.0643046
## log_freq  0.2374669
```

```
sse_results_list<-compare_SS_accounted_for(FinalModelSet,"preserved ~ 1",PosDat,N_cutoff=10)
```

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

## Warning in (null_cumcor_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

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

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

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

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.3570464	2666.060
preserved ~ CumErr+CumPres	0.5329802	2653.935
preserved ~ CumErr+CumPres+log_freq	0.5465865	2641.169
preserved ~ CumErr+CumPres+log_freq+stimlen	0.5574713	2639.211

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

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

```
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
## Warning in (null_cumerr_resid^2) * (N_values$len_pos_N): longer object length is not a
## multiple of shorter object length
```

```
sse_table<-sse_results_table(sse_results_list)
write.csv(sse_table,paste0(TablesDir,CurPat,"_",CurTask,"_sse_results_table.csv"),row.names = TRUE)
sse_table
```

```
##                                           model p_accounted_for model_deviance diff_CumErr
## 1                            preserved ~ CumErr       0.3570464       2666.060   0.0000000
## 2                    preserved ~ CumErr+CumPres       0.5329802       2653.935   0.1759338
## 3           preserved ~ CumErr+CumPres+log_freq       0.5465865       2641.169   0.1895401
## 4   preserved ~ CumErr+CumPres+log_freq+stimlen       0.5574713       2639.211   0.2004249
##   diff_CumErr+CumPres diff_CumErr+CumPres+log_freq diff_CumErr+CumPres+log_freq+stimlen
## 1         -0.17593376                  -0.18954006                          -0.20042489
## 2          0.00000000                  -0.01360630                          -0.02449113
## 3          0.01360630                   0.00000000                          -0.01088484
## 4          0.02449113                   0.01088484                           0.00000000
```

```
kable(sse_table[,1:3], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

```
write.csv(results_report_DF,paste0(TablesDir,CurPat,"_",CurTask,"_results_report_df.csv"),row.names = FALSE)
```

```
kable(sse_table[,c(1,4:6)], format="latex", booktabs=TRUE) %>%
  kable_styling(latex_options="scale_down")
```

model	diff_CumErr	diff_CumErr+CumPres	diff_CumErr+CumPres+log_freq
preserved ~ CumErr	0.0000000	-0.1759338	-0.1895401
preserved ~ CumErr+CumPres	0.1759338	0.0000000	-0.0136063
preserved ~ CumErr+CumPres+log_freq	0.1895401	0.0136063	0.0000000
preserved ~ CumErr+CumPres+log_freq+stimlen	0.2004249	0.0244911	0.0108848