RM - naming - 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	224	15	29	NA	NA	268
2	24	NA	195	19	30	268
3	108	NA	51	105	4	268
4	138	NA	71	17	13	239
5	64	1	71	22	12	170
6	62	NA	27	31	7	127
7	51	NA	16	5	3	75
8	17	NA	11	3	1	32
9	13	NA	NA	NA	1	14

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8358209	0.0559701	0.1082090	NA	NA	268
2	0.0895522	NA	0.7276119	0.0708955	0.1119403	268
3	0.4029851	NA	0.1902985	0.3917910	0.0149254	268
4	0.5774059	NA	0.2970711	0.0711297	0.0543933	239
5	0.3764706	0.0058824	0.4176471	0.1294118	0.0705882	170
6	0.4881890	NA	0.2125984	0.2440945	0.0551181	127

pos_factor	O	P	V	1	S	total
7	0.6800000	NA	0.2133333	0.0666667	0.0400000	75
8	0.5312500	NA	0.3437500	0.0937500	0.0312500	32
9	0.9285714	NA	NA	NA	0.0714286	14

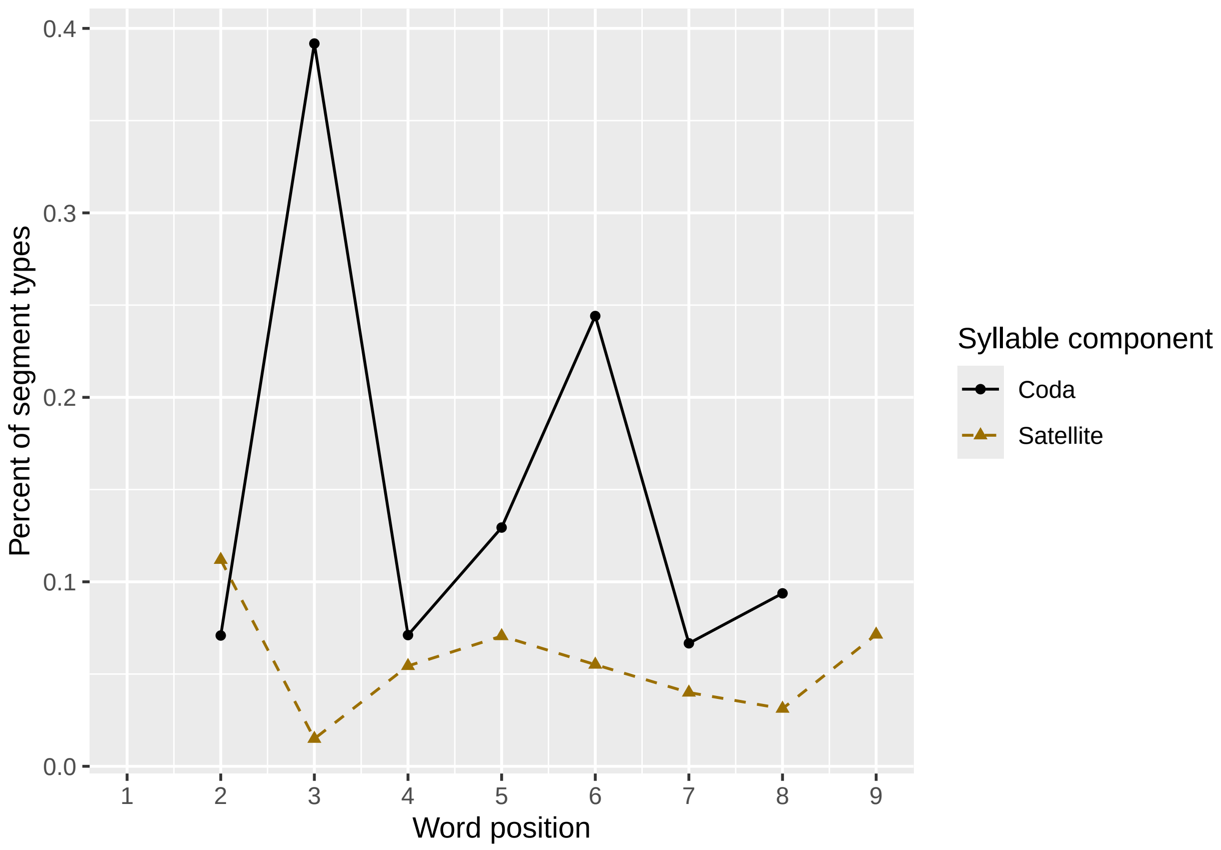
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 0.948 0.931 0.931 NA     NA     NA     NA     NA     NA
## 2       5 0.913 0.942 0.920  0.884 NA     NA     NA     NA     NA
## 3       6 0.884 0.884 0.860  0.814  0.791 NA     NA     NA     NA
## 4       7 0.913 0.885 0.856  0.837  0.837  0.808 NA     NA     NA
## 5       8 0.930 0.895 0.891  0.849  0.868  0.849  0.814 NA     NA
## 6       9 0.833 0.852 0.833  0.833  0.722  0.704  0.75   0.843 NA
## 7      10 0.745 0.796 0.735  0.806  0.888  0.857  0.745  0.857  0.714
```

```
# 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.
```

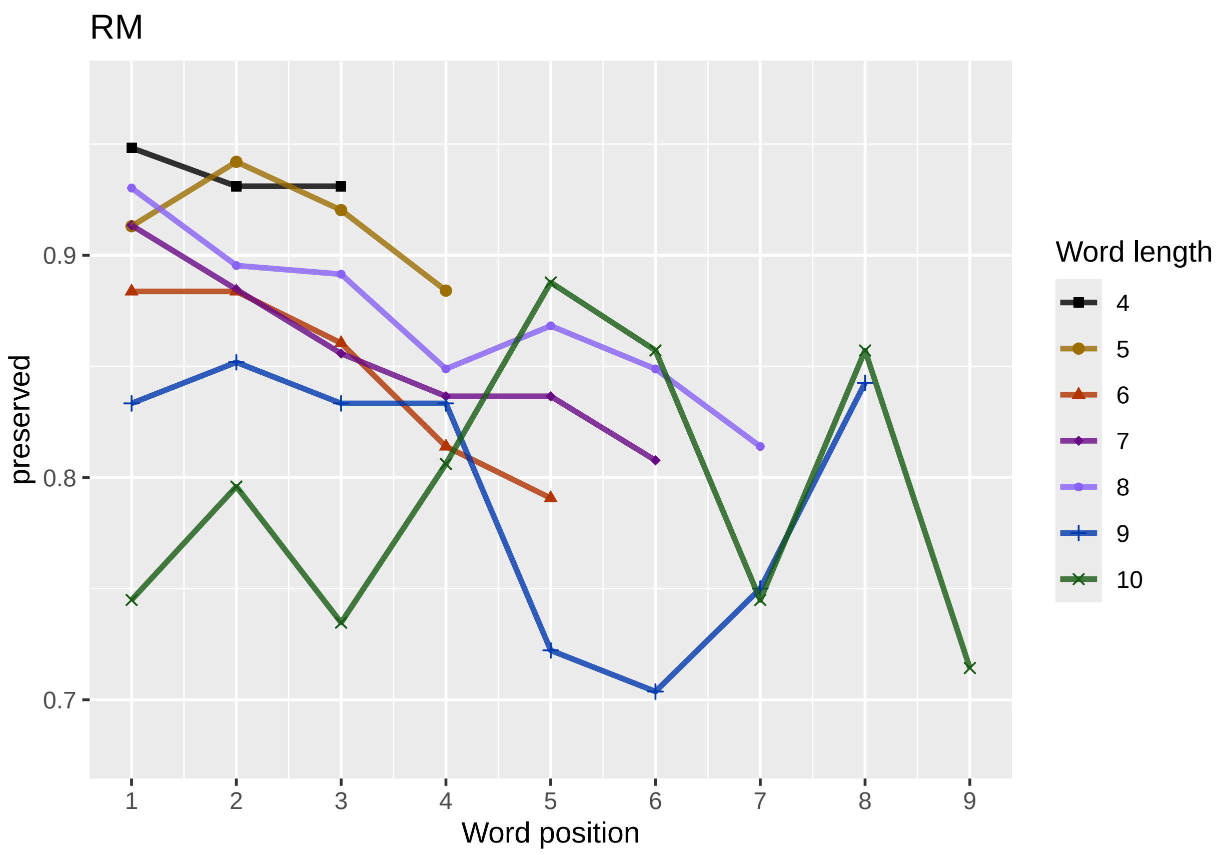
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    29    29    29    NA    NA    NA    NA    NA    NA
## 2       5    69    69    69    69    NA    NA    NA    NA    NA
## 3       6    43    43    43    43    43    NA    NA    NA    NA
## 4       7    52    52    52    52    52    52    NA    NA    NA
## 5       8    43    43    43    43    43    43    43    NA    NA
## 6       9    18    18    18    18    18    18    18    18    NA
## 7      10    14    14    14    14    14    14    14    14    14
```

```
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:  5
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##      4.2080      -0.2666      -0.4392       0.0416
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1109  AIC: 1194
## log likelihood:  -554.6168
## Nagelkerke R2:  0.0308274
## % pres/err predicted correctly:  -336.2694
## % of predictable range [ (model-null)/(1-null) ]:  0.01785716
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.1107      -0.1265      -0.1011
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1112  AIC: 1194
## log likelihood:  -556.0231
## Nagelkerke R2:  0.02731663
## % pres/err predicted correctly:  -336.6979
## % of predictable range [ (model-null)/(1-null) ]:  0.01660915
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.43010     -0.13787      0.01796     -0.25097
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1111  AIC: 1195
## log likelihood:  -555.4764
## Nagelkerke R2:  0.02868213
## % pres/err predicted correctly:  -336.5426
## % of predictable range [ (model-null)/(1-null) ]:  0.01706165
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##        3.12        -0.18
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1197
## log likelihood:  -559.0403
## Nagelkerke R2:  0.01976182
## % pres/err predicted correctly:  -338.1971
## % of predictable range [ (model-null)/(1-null) ]:  0.01224341
## ***************************
## 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)
##        4.358565         -0.287046          0.012081         -0.535452         -0.001544
##      stimlen:pos
##        0.054304
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1455 Residual
## Null Deviance:       1134
## Residual Deviance: 1109  AIC: 1198
## log likelihood:  -554.6063
## Nagelkerke R2:  0.03085349
## % pres/err predicted correctly:  -336.2651
## % of predictable range [ (model-null)/(1-null) ]:  0.01786947
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3864      -0.1496
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1200
## log likelihood:  -559.0287
## Nagelkerke R2:  0.01979099
## % pres/err predicted correctly:  -338.2881
## % of predictable range [ (model-null)/(1-null) ]:  0.01197868
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.507068     0.008543    -0.222840
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1201
## log likelihood:  -558.8998
## Nagelkerke R2:  0.02011432
## % pres/err predicted correctly:  -338.2814
## % of predictable range [ (model-null)/(1-null) ]:  0.01199808
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.835
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1460 Residual
## Null Deviance:       1134
## Residual Deviance: 1134  AIC: 1213
## log likelihood:  -566.8741
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -342.4016
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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	1193.519	0.0000000	1.0000000	0.3825170	0.0308274	4.4207971	-0.2665926	-0.4391814	0.0416016	NA	NA
preserved ~ stimlen + pos	1194.134	0.6156460	0.7350452	0.2811670	0.0273163	3.6110729	-0.1265410	-0.1011372	NA	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	1194.841	1.324447	0.5157032	0.1972652	0.0286832	1.430105	-0.1378737	-0.2509742	NA	0.0179573	NA
preserved ~ stimlen	1197.103	3.583722	0.1666497	0.0637463	0.0197633	3.120125	-0.1800221	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1197.531	4.011943	0.1345295	0.0514598	0.0308545	4.358565	-0.2870460	-0.5354516	0.0543042	0.0120810	-0.0015439
preserved ~ pos	1199.778	6.259237	0.0437345	0.0167292	0.0197912	2.386369	NA	-0.1495964	NA	NA	NA
preserved ~ I(pos^2) + pos	1201.497	7.976318	0.0185338	0.0070895	0.0201142	2.507068	NA	-0.2228404	NA	0.0085426	NA
preserved ~ 1	1212.741	19.220634	0.0000670	0.0000256	0.0000000	1.834601	NA	NA	NA	NA	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  stimlen:pos
##      4.2080      -0.2666      -0.4392       0.0416
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1109  AIC: 1194
```

```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.
```

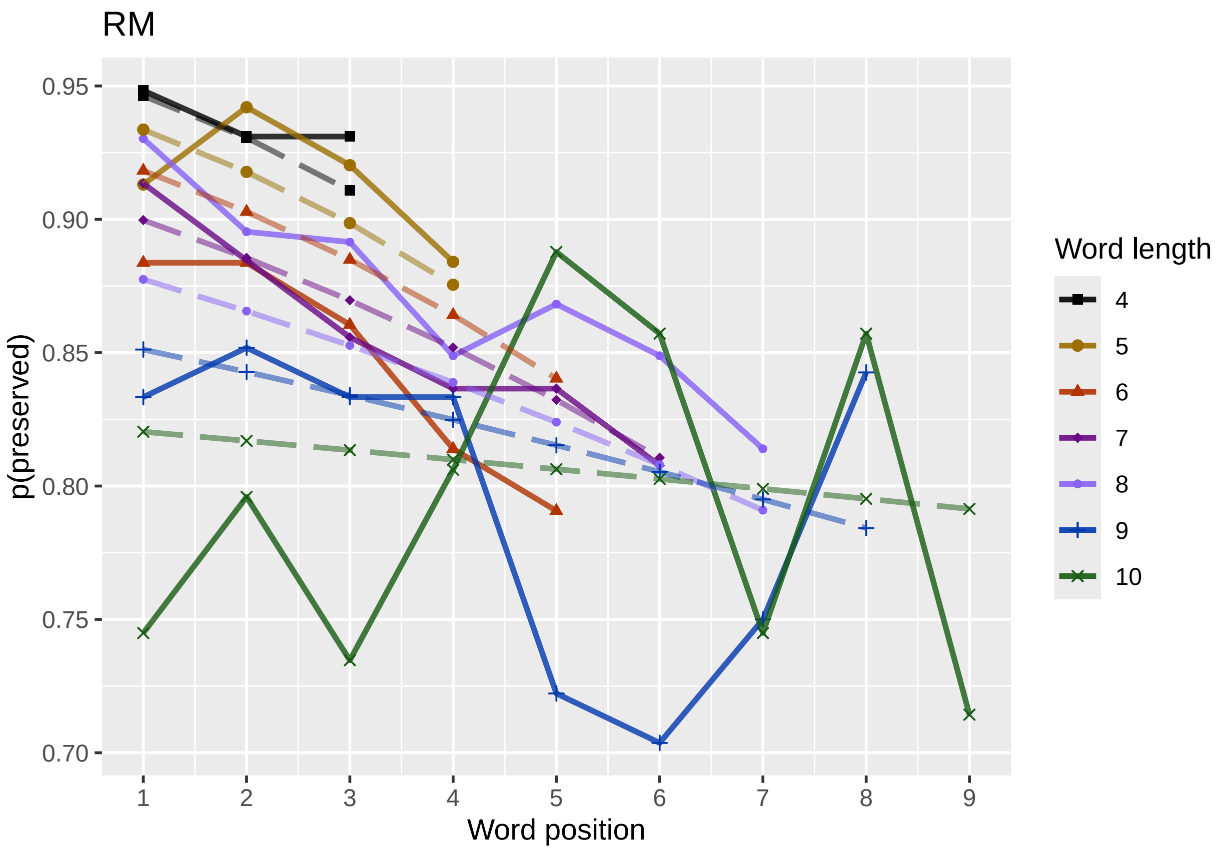
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # 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 0.946 0.931 0.911 NA     NA     NA     NA     NA     NA
## 2       5 0.934 0.918 0.899  0.875 NA     NA     NA     NA     NA
## 3       6 0.918 0.903 0.885  0.864  0.840 NA     NA     NA     NA
## 4       7 0.900 0.886 0.870  0.852  0.832  0.811 NA     NA     NA
## 5       8 0.877 0.866 0.853  0.839  0.824  0.808  0.791 NA     NA
## 6       9 0.851 0.843 0.834  0.825  0.815  0.805  0.795  0.784 NA
## 7      10 0.820 0.817 0.813  0.810  0.806  0.803  0.799  0.795  0.791
```

```
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

```r
# 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
```

```r
# 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       20   268
```

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

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

```r
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 20 / 268 = 7.46 percent"
```

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

```r
# 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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen
##      3.4608      -0.1877
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1406 Residual
## Null Deviance:       931.9
## Residual Deviance: 918.6     AIC: 982.9
## log likelihood:  -459.2808
## Nagelkerke R2:  0.01947209
## % pres/err predicted correctly:  -263.6917
## % of predictable range [ (model-null)/(1-null) ]:  0.01100138
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##    3.460747    -0.189184     0.003031
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1405 Residual
## Null Deviance:       931.9
## Residual Deviance: 918.6     AIC: 984.9
## log likelihood:  -459.2787
## Nagelkerke R2:  0.01947834
## % pres/err predicted correctly:  -263.6905
## % of predictable range [ (model-null)/(1-null) ]:  0.01100574
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    3.579085    -0.193192     0.007326    -0.056369
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1404 Residual
## Null Deviance:       931.9
## Residual Deviance: 918.4     AIC: 986.7
## log likelihood:  -459.2095
## Nagelkerke R2:  0.01967938
## % pres/err predicted correctly:  -263.6743
## % of predictable range [ (model-null)/(1-null) ]:  0.01106643
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##    3.591880    -0.205820    -0.040743     0.005352
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1404 Residual
## Null Deviance:       931.9
## Residual Deviance: 918.5     AIC: 986.8
## log likelihood:  -459.2607
## Nagelkerke R2:  0.0195306
## % pres/err predicted correctly:  -263.6851
## % of predictable range [ (model-null)/(1-null) ]:  0.01102596
## ***************************
## 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)
##         3.438267         -0.174449         -0.002674          0.029541          0.001259
##      stimlen:pos
##        -0.011116
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1402 Residual
## Null Deviance:       931.9
## Residual Deviance: 918.4     AIC: 990.7
## log likelihood:  -459.2034
## Nagelkerke R2:  0.01969719
## % pres/err predicted correctly:  -263.6716
## % of predictable range [ (model-null)/(1-null) ]:  0.01107624
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.118
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1407 Residual
## Null Deviance:       931.9
## Residual Deviance: 931.9     AIC: 995.3
## log likelihood:  -465.9486
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -266.636
## % 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
##     2.34492     -0.06436
```

```
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1406 Residual
## Null Deviance:       931.9
## Residual Deviance: 929.7     AIC: 995.5
## log likelihood:  -464.8584
## Nagelkerke R2:  0.003196444
## % pres/err predicted correctly:  -266.131
## % of predictable range [ (model-null)/(1-null) ]:  0.001886894
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.272335    -0.005657    -0.017550
##
## Degrees of Freedom: 1407 Total (i.e. Null);  1405 Residual
## Null Deviance:       931.9
## Residual Deviance: 929.6     AIC: 997.4
## log likelihood:  -464.8158
## Nagelkerke R2:  0.003321416
## % pres/err predicted correctly:  -266.0975
## % of predictable range [ (model-null)/(1-null) ]:  0.002012095
## ***************************
```

```
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=FALSI
NoFragLPAICSummary <- subset(NoFragLPAICSummary, select = -c(Row.names))

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen	982.8805	0.000000	1.0000000	0.5927733	0.0194723	1.460796	-0.1876984	NA	NA	NA	NA
preserved ~ stimlen + pos	984.8524	1.971912	0.3730824	0.2211533	0.0194783	3.460747	-0.1891838	0.0030314	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	986.6943	3.813778	0.1485410	0.0880516	0.0196793	4.579085	-0.1931920	-0.0563685	NA	0.0073261	NA
preserved ~ stimlen * pos	986.8053	3.924669	0.1405300	0.0833024	0.0195336	3.591880	-0.2058204	-0.0407431	0.0053516	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)	990.6722	2.791630	0.0203260	0.0120492	0.0196973	2.438267	-0.1744489	0.0295411	-0.0111162	-0.0026742	0.0012591
preserved ~ 1	995.3490	12.468497	0.0019610	0.0011625	0.0000000	2.118497	NA	NA	NA	NA	NA
preserved ~ pos	995.4813	12.600756	0.0018356	0.0010881	0.0031964	2.344925	NA	-0.0643591	NA	NA	NA
preserved ~ I(pos^2) + pos	997.3875	14.506996	0.0007077	0.0004195	0.0033214	2.272335	NA	-0.0175497	NA	-0.0056569	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.
```

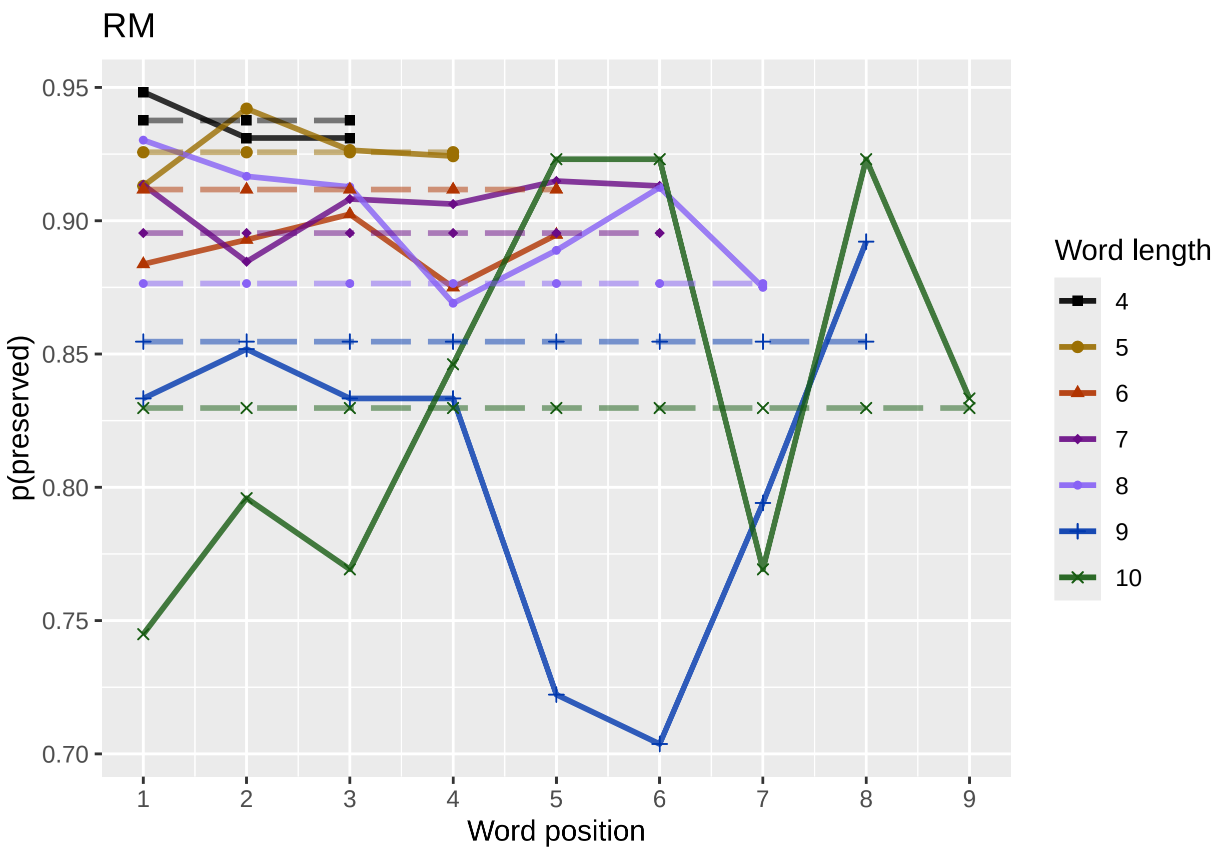
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.938 0.938 0.938 NA     NA     NA     NA     NA     NA
## 2       5 0.926 0.926 0.926  0.926 NA     NA     NA     NA     NA
## 3       6 0.912 0.912 0.912  0.912  0.912 NA     NA     NA     NA
## 4       7 0.895 0.895 0.895  0.895  0.895  0.895 NA     NA     NA
## 5       8 0.876 0.876 0.876  0.876  0.876  0.876  0.876 NA     NA
## 6       9 0.855 0.855 0.855  0.855  0.855  0.855  0.855  0.855 NA
## 7      10 0.830 0.830 0.830  0.830  0.830  0.830  0.830  0.830  0.830
```

```
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: 0.68 - 0.97"
```

```
# 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
```

```r
# 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.
# 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] -0.01305843
```

```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] -0.01116671
```

```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_upwar
  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:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq         I(pos^2)              pos
##         2.59778          0.02500         -0.44886          0.02591         -0.32265
## stimlen:log_freq
##         0.10383
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1455 Residual
## Null Deviance:      1134
## Residual Deviance: 1070  AIC: 1157
## log likelihood:  -534.9029
## Nagelkerke R2:  0.07933585
## % pres/err predicted correctly:  -323.3567
## % of predictable range [ (model-null)/(1-null) ]:  0.05545946
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq              pos  stimlen:log_freq
##         2.15805          0.03726         -0.41056         -0.10606          0.09842
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1456 Residual
## Null Deviance:      1134
## Residual Deviance: 1072  AIC: 1158
## log likelihood:  -535.9937
## Nagelkerke R2:  0.07668583
## % pres/err predicted correctly:  -323.7266
## % of predictable range [ (model-null)/(1-null) ]:  0.05438228
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.155597          0.041921         -0.411130         -0.114027          0.103061
##     log_freq:pos
##        -0.008593
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1455 Residual
## Null Deviance:       1134
## Residual Deviance: 1072  AIC: 1160
## log likelihood:  -535.9457
## Nagelkerke R2:  0.07680241
## % pres/err predicted correctly:  -323.7866
## % of predictable range [ (model-null)/(1-null) ]:  0.05420753
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##           2.79115            0.01374           -0.22530            0.04066           -0.42168
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##           0.09367            0.01387           -0.10573
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1453 Residual
## Null Deviance:       1134
## Residual Deviance: 1069  AIC: 1160
## log likelihood:  -534.2543
## Nagelkerke R2:  0.08090953
## % pres/err predicted correctly:  -322.8127
## % of predictable range [ (model-null)/(1-null) ]:  0.05704363
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.17125          -0.01955          -0.40518           0.09740
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1078  AIC: 1161
## log likelihood:  -539.1569
## Nagelkerke R2:  0.06897868
## % pres/err predicted correctly:  -325.4092
## % of predictable range [ (model-null)/(1-null) ]:  0.04948257
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      2.3194      -0.1089       0.3003
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1083  AIC: 1163
## log likelihood:  -541.2909
## Nagelkerke R2:  0.06376036
## % pres/err predicted correctly:  -327.2054
## % of predictable range [ (model-null)/(1-null) ]:  0.04425178
## ***************************
## 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
##           2.77847            0.04505           -0.43564            0.43661            0.02103
##      pos:log_freq
##          -0.13360
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1455 Residual
## Null Deviance:       1134
## Residual Deviance: 1076  AIC: 1164
## log likelihood:  -538.2409
## Nagelkerke R2:  0.07121388
## % pres/err predicted correctly:  -325.0457
## % of predictable range [ (model-null)/(1-null) ]:  0.05054091
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.25140      -0.08669       0.19376       0.02835
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1081  AIC: 1164
## log likelihood:  -540.6546
## Nagelkerke R2:  0.06531802
## % pres/err predicted correctly:  -326.4242
## % of predictable range [ (model-null)/(1-null) ]:  0.04652669
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           3.07243           -0.04350            0.04842           -0.44774            0.41890
## I(pos^2):log_freq       pos:log_freq
##           0.02154           -0.13519
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1454 Residual
## Null Deviance:       1134
## Residual Deviance: 1076  AIC: 1165
## log likelihood:  -537.9447
## Nagelkerke R2:  0.0719361
## % pres/err predicted correctly:  -324.8106
## % of predictable range [ (model-null)/(1-null) ]:  0.05122554
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.41637     -0.01662     -0.10349      0.29441
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1082  AIC: 1165
## log likelihood:  -541.2456
## Nagelkerke R2:  0.06387127
## % pres/err predicted correctly:  -327.1531
## % of predictable range [ (model-null)/(1-null) ]:  0.04440424
## **************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     2.75861     -0.02831      0.01946     -0.26592      0.29525
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1456 Residual
## Null Deviance:       1134
## Residual Deviance: 1081  AIC: 1166
## log likelihood:  -540.6151
## Nagelkerke R2:  0.06541457
## % pres/err predicted correctly:  -326.9962
## % of predictable range [ (model-null)/(1-null) ]:  0.04486109
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.37343      -0.02112      -0.07935       0.18399       0.02897
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1456 Residual
## Null Deviance:       1134
## Residual Deviance: 1081  AIC: 1166
## log likelihood:  -540.582
## Nagelkerke R2:  0.06549567
## % pres/err predicted correctly:  -326.3419
## % of predictable range [ (model-null)/(1-null) ]:  0.0467664
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.42832     -0.07174      0.29295
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1089  AIC: 1168
## log likelihood:  -544.3328
## Nagelkerke R2:  0.05629565
## % pres/err predicted correctly:  -328.8532
## % of predictable range [ (model-null)/(1-null) ]:  0.03945337
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##      4.2080      -0.2666      -0.4392       0.0416
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1109  AIC: 1194
## log likelihood:  -554.6168
## Nagelkerke R2:  0.0308274
## % pres/err predicted correctly:  -336.2694
## % of predictable range [ (model-null)/(1-null) ]:  0.01785716
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.1107      -0.1265      -0.1011
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1112  AIC: 1194
## log likelihood:  -556.0231
## Nagelkerke R2:  0.02731663
## % pres/err predicted correctly:  -336.6979
## % of predictable range [ (model-null)/(1-null) ]:  0.01660915
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.43010     -0.13787      0.01796     -0.25097
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 1111  AIC: 1195
## log likelihood:  -555.4764
## Nagelkerke R2:  0.02868213
## % pres/err predicted correctly:  -336.5426
## % of predictable range [ (model-null)/(1-null) ]:  0.01706165
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##        3.12        -0.18
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1197
## log likelihood:  -559.0403
## Nagelkerke R2:  0.01976182
## % pres/err predicted correctly:  -338.1971
## % of predictable range [ (model-null)/(1-null) ]:  0.01224341
## ***************************
## 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)
##         4.358565         -0.287046          0.012081         -0.535452         -0.001544
##      stimlen:pos
##         0.054304
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1455 Residual
## Null Deviance:       1134
## Residual Deviance: 1109  AIC: 1198
## log likelihood:  -554.6063
## Nagelkerke R2:  0.03085349
## % pres/err predicted correctly:  -336.2651
## % of predictable range [ (model-null)/(1-null) ]:  0.01786947
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3864      -0.1496
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1200
## log likelihood:  -559.0287
## Nagelkerke R2:  0.01979099
## % pres/err predicted correctly:  -338.2881
## % of predictable range [ (model-null)/(1-null) ]:  0.01197868
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.507068     0.008543    -0.222840
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1201
## log likelihood:  -558.8998
## Nagelkerke R2:  0.02011432
## % pres/err predicted correctly:  -338.2814
## % of predictable range [ (model-null)/(1-null) ]:  0.01199808
## ***************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.835
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1460 Residual
## Null Deviance:       1134
## Residual Deviance: 1134  AIC: 1213
## log likelihood:  -566.8741
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -342.4016
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.4488618	0.1038253	-0.3226499	NA	NA	0.0259123	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.4105598	0.0984229	-0.1060625	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.4111302	0.1030611	-0.1140268	NA	-0.0085933	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2253040	0.0936690	-0.4216802	NA	-0.1057253	0.0406556	NA	0.0138718	NA	NA

Model	AIC DeltaAIC expAIC AICw NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)	
preserved ~ stimlen * log_freq	1160.[illegible]	-0.0196[illegible]	-0.05[illegible]	0.0973974	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	1163.[illegible]	NA	0.30029[illegible]	NA	-0.1089400	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	1163.[illegible]	NA	0.43660[illegible]	NA	-0.4356[illegible]	-0.[illegible]35975	NA	0.0450[illegible]	0.0[illegible]	NA	NA	NA
preserved ~ pos * log_freq	1164.[illegible]	NA	0.1937[illegible]	NA	-0.0866867	0.02835[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	1164.[illegible]	-0.0434960	0.4189[illegible]	NA	-0.447[illegible]	-0.[illegible]51936	NA	0.048[illegible]	0.0[illegible]	NA	NA	NA
preserved ~ stimlen + pos + log_freq	1165.[illegible]	-0.0166199	0.2944[illegible]	NA	-0.1034946	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	1165.[illegible]	-0.0283100	0.2952[illegible]	NA	-0.2659175	NA	NA	0.0194[illegible]	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	1166.[illegible]	-0.0211193	0.1839[illegible]	NA	-0.0793498	0.0289[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	1168.[illegible]	-0.0717364	0.2929[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	1193.[illegible]	-0.2665926	NA	NA	-0.4391814	NA	NA	NA	NA	NA	0.0416[illegible]	NA
preserved ~ stimlen + pos	1194.[illegible]	-0.1265417	NA	NA	-0.1011372	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1194.[illegible]	-0.1378737	NA	NA	-0.2509742	NA	NA	0.0179[illegible]	NA	NA	NA	NA
preserved ~ stimlen	1197.[illegible]	-0.1800221	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1197.[illegible]	-0.2870460	NA	NA	-0.5354516	NA	NA	0.0120[illegible]	NA	NA	0.0543042	-0.0015439
preserved ~ pos	1199.[illegible]	NA	NA	NA	-0.1495964	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1201.[illegible]	NA	NA	NA	-0.2228404	NA	NA	0.0085[illegible]	NA	NA	NA	NA
preserved ~ 1	1212.[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ stimlen * log_freq + I(pos^2) + pos"
```

```
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)          stimlen         log_freq         I(pos^2)              pos
##          2.59778          0.02500         -0.44886          0.02591         -0.32265
## stimlen:log_freq
##          0.10383
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1455 Residual
## Null Deviance:       1134
## Residual Deviance: 1070  AIC: 1157
```

```
# 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"
```

```
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.
```

```
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.
```

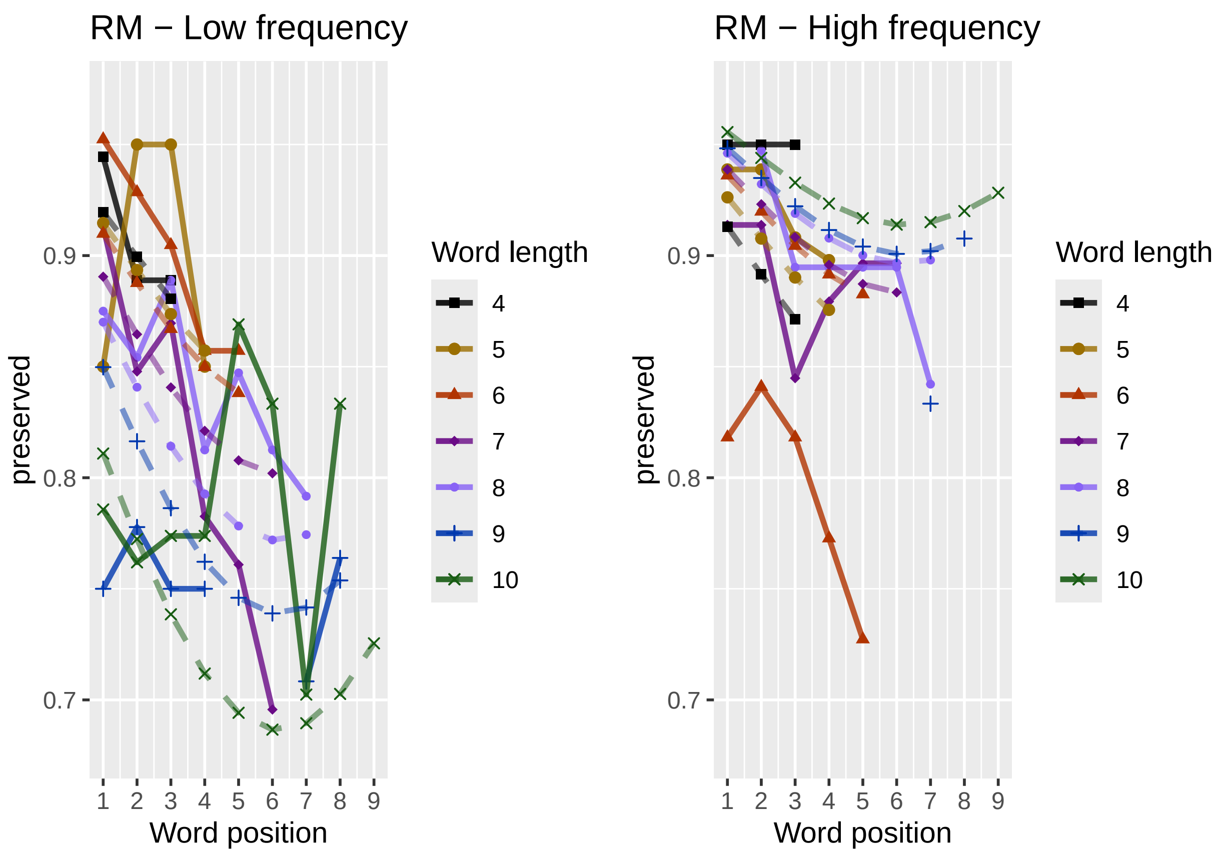
```
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
```

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

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 17 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Warning: Removed 17 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

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



```
# 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
##       2.616       -1.432
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 787   AIC: 856.4
## log likelihood:  -393.503
## Nagelkerke R2:  0.3914156
## % pres/err predicted correctly:  -218.9804
## % of predictable range [ (model-null)/(1-null) ]:  0.3594076
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.1853       0.3776
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1072  AIC: 1151
## log likelihood:  -535.8483
## Nagelkerke R2:  0.0770392
## % pres/err predicted correctly:  -327.0999
## % of predictable range [ (model-null)/(1-null) ]:  0.0445591
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##        3.12        -0.18
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1197
## log likelihood:  -559.0403
## Nagelkerke R2:  0.01976182
## % pres/err predicted correctly:  -338.1971
## % of predictable range [ (model-null)/(1-null) ]:  0.01224341
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)          pos
##      2.3864      -0.1496
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1200
## log likelihood:  -559.0287
## Nagelkerke R2:  0.01979099
## % pres/err predicted correctly:  -338.2881
## % of predictable range [ (model-null)/(1-null) ]:  0.01197868
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.507068     0.008543    -0.222840
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1201
## log likelihood:  -558.8998
## Nagelkerke R2:  0.02011432
## % pres/err predicted correctly:  -338.2814
## % of predictable range [ (model-null)/(1-null) ]:  0.01199808
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.835
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1460 Residual
## Null Deviance:       1134
## Residual Deviance: 1134  AIC: 1213
## log likelihood:  -566.8741
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -342.4016
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```
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)
```

```r
MEAICSummary$NagR2<-MERes$NagR2

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

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	856.3598	0.0000	1	1	0.3914152	2.615613	NA	-1.431687	NA	NA	NA
preserved ~ CumPres	1151.2443	294.8845	0	0	0.0770391	1.185251	0.3776145	NA	NA	NA	NA
preserved ~ stimlen	1197.1025	340.7427	0	0	0.0197618	1.120125	NA	NA	NA	NA	-0.1800221
preserved ~ pos	1199.7780	343.4182	0	0	0.0197910	1.386369	NA	NA	NA	-0.1495964	NA
preserved ~ (I(pos^2) + pos)	1201.4951	345.1353	0	0	0.0201143	1.507068	NA	NA	0.0085426	-0.2228404	NA
preserved ~ 1	1212.7408	356.3810	0	0	0.0000000	0.834601	NA	NA	NA	NA	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)
}
```

```
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	0.8322725	202
O	0.8517084	701
P	0.9375000	16
S	0.8873239	71
V	0.8846426	471

```
#  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
##       2.578       -1.465
##
## Degrees of Freedom: 1373 Total (i.e. Null);  1372 Residual
## Null Deviance:       1075
## Residual Deviance: 750.9     AIC: 821.7
## log likelihood:  -375.4516
## Nagelkerke R2:  0.3871039
## % pres/err predicted correctly:  -209.3193
## % of predictable range [ (model-null)/(1-null) ]:  0.3568041
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.1956       0.3842
##
## Degrees of Freedom: 1373 Total (i.e. Null);  1372 Residual
## Null Deviance:       1075
## Residual Deviance: 1019  AIC: 1099
## log likelihood:  -509.5561
## Nagelkerke R2:  0.07331687
## % pres/err predicted correctly:  -312.3047
## % of predictable range [ (model-null)/(1-null) ]:  0.04185526
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       3.066       -0.175
##
## Degrees of Freedom: 1373 Total (i.e. Null);  1372 Residual
## Null Deviance:       1075
## Residual Deviance: 1061  AIC: 1139
## log likelihood:  -530.4072
## Nagelkerke R2:  0.01878798
## % pres/err predicted correctly:  -322.1172
## % of predictable range [ (model-null)/(1-null) ]:  0.01184689
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3619      -0.1475
##
## Degrees of Freedom: 1373 Total (i.e. Null);  1372 Residual
## Null Deviance:       1075
## Residual Deviance: 1060  AIC: 1142
## log likelihood:  -530.1858
## Nagelkerke R2:  0.01937553
## % pres/err predicted correctly:  -322.1068
## % of predictable range [ (model-null)/(1-null) ]:  0.01187862
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.43489      0.00516     -0.19178
##
## Degrees of Freedom: 1373 Total (i.e. Null);  1371 Residual
## Null Deviance:       1075
## Residual Deviance: 1060  AIC: 1144
## log likelihood:  -530.1406
## Nagelkerke R2:  0.01949557
## % pres/err predicted correctly:  -322.1216
## % of predictable range [ (model-null)/(1-null) ]:  0.01183346
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.816
##
## Degrees of Freedom: 1373 Total (i.e. Null);  1373 Residual
## Null Deviance:       1075
## Residual Deviance: 1075  AIC: 1153
## log likelihood:  -537.4473
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -325.991
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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	821.7106	0.0000	1	1	0.3871032	2.577988	NA	-1.46481	NA	NA	NA
preserved ~ CumPres	1098.7198	277.0092	0	0	0.0733169	1.195571	0.384212	NA	NA	NA	NA
preserved ~ stimlen	1139.4982	317.7876	0	0	0.0187883	3.065926	NA	NA	NA	NA	-0.1749745
preserved ~ pos	1141.7193	320.0088	0	0	0.0193755	2.361853	NA	NA	NA	-0.1474893	NA
preserved ~ (I(pos^2) + pos)	1143.6127	321.9021	0	0	0.0194956	2.434889	NA	NA	0.0051595	-0.1917792	NA
preserved ~ 1	1153.4941	331.7836	0	0	0.0000000	1.816459	NA	NA	NA	NA	NA

```r
#  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
##       2.515       -1.572
##
## Degrees of Freedom: 1171 Total (i.e. Null);  1170 Residual
## Null Deviance:       894.6
## Residual Deviance: 665   AIC: 730.5
## log likelihood:  -332.5095
## Nagelkerke R2:  0.3331821
## % pres/err predicted correctly:  -187.2595
## % of predictable range [ (model-null)/(1-null) ]:  0.3044231
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.3698       0.3502
##
## Degrees of Freedom: 1171 Total (i.e. Null);  1170 Residual
## Null Deviance:       894.6
## Residual Deviance: 864.6     AIC: 938.4
## log likelihood:  -432.2767
## Nagelkerke R2:  0.047365
## % pres/err predicted correctly:  -262.368
## % of predictable range [ (model-null)/(1-null) ]:  0.02691416
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.1197      -0.1775
##
## Degrees of Freedom: 1171 Total (i.e. Null);  1170 Residual
## Null Deviance:       894.6
## Residual Deviance: 882.3     AIC: 954.1
## log likelihood:  -441.1376
## Nagelkerke R2:  0.01954776
## % pres/err predicted correctly:  -266.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.01226616
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##       2.394       -0.147
##
## Degrees of Freedom: 1171 Total (i.e. Null);  1170 Residual
## Null Deviance:       894.6
## Residual Deviance: 881.7     AIC: 955.5
## log likelihood:  -440.8443
## Nagelkerke R2:  0.0204753
## % pres/err predicted correctly:  -266.2342
## % of predictable range [ (model-null)/(1-null) ]:  0.01262935
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.482065     0.006414    -0.201829
##
## Degrees of Freedom: 1171 Total (i.e. Null);  1169 Residual
## Null Deviance:       894.6
## Residual Deviance: 881.6     AIC: 957.4
## log likelihood:  -440.781
## Nagelkerke R2:  0.02067549
## % pres/err predicted correctly:  -266.2344
## % of predictable range [ (model-null)/(1-null) ]:  0.0126288
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.857
##
## Degrees of Freedom: 1171 Total (i.e. Null);  1171 Residual
## Null Deviance:       894.6
## Residual Deviance: 894.6     AIC: 966
## log likelihood:  -447.2852
## Nagelkerke R2:  4.159171e-16
## % pres/err predicted correctly:  -269.6524
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

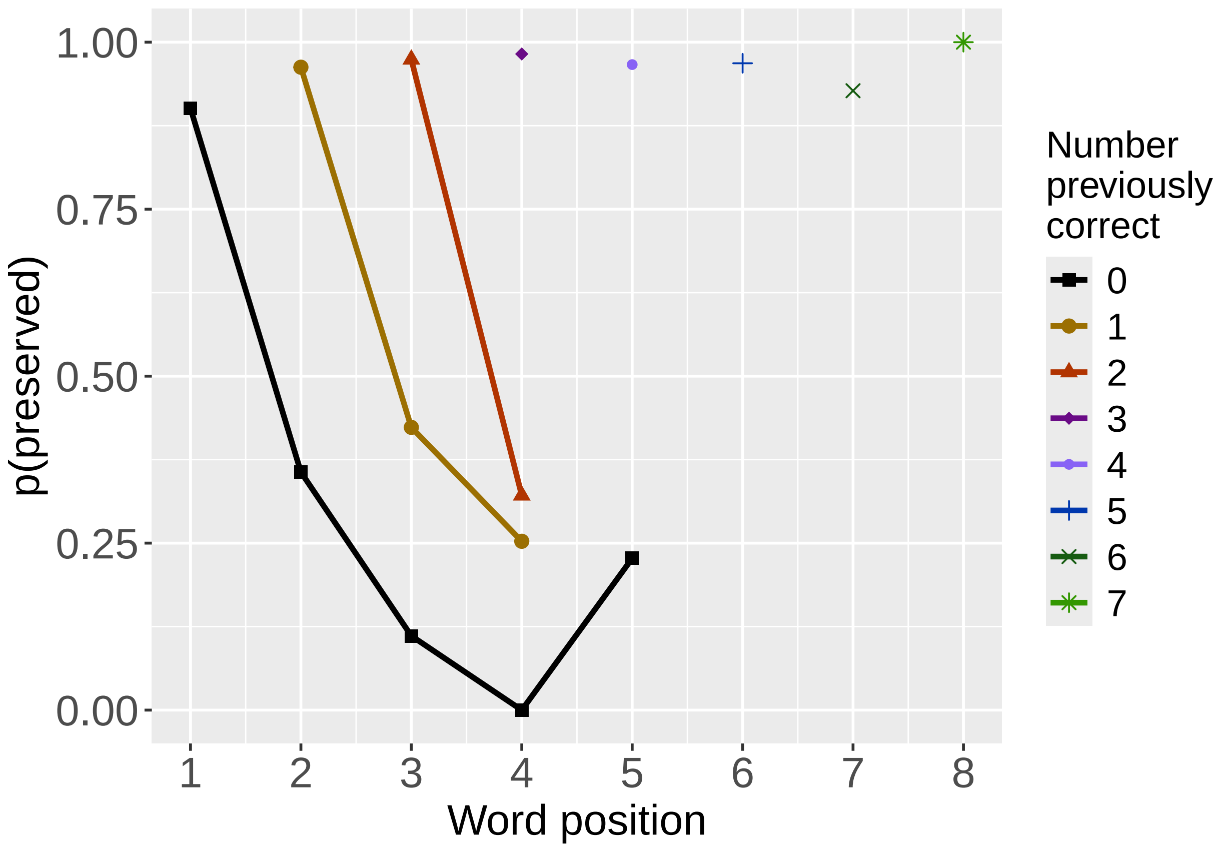
```r
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	730.4847	0.0000	1	1	0.3331822	1.515278	NA	-1.572329	NA	NA	NA
preserved ~ CumPres	938.3816	207.8969	0	0	0.0473650	1.369776	0.3502208	NA	NA	NA	NA
preserved ~ stimlen	954.0704	223.5857	0	0	0.0195478	3.119703	NA	NA	NA	NA	-0.1775061
preserved ~ pos	955.5491	225.0645	0	0	0.0204753	2.394342	NA	NA	NA	-0.1470027	NA
preserved ~ (I(pos^2) + pos)	957.3960	226.9113	0	0	0.0206755	2.482065	NA	NA	0.0064143	-0.2018285	NA
preserved ~ 1	966.0358	235.5511	0	0	0.0000000	1.856975	NA	NA	NA	NA	NA

```r
# 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.
```

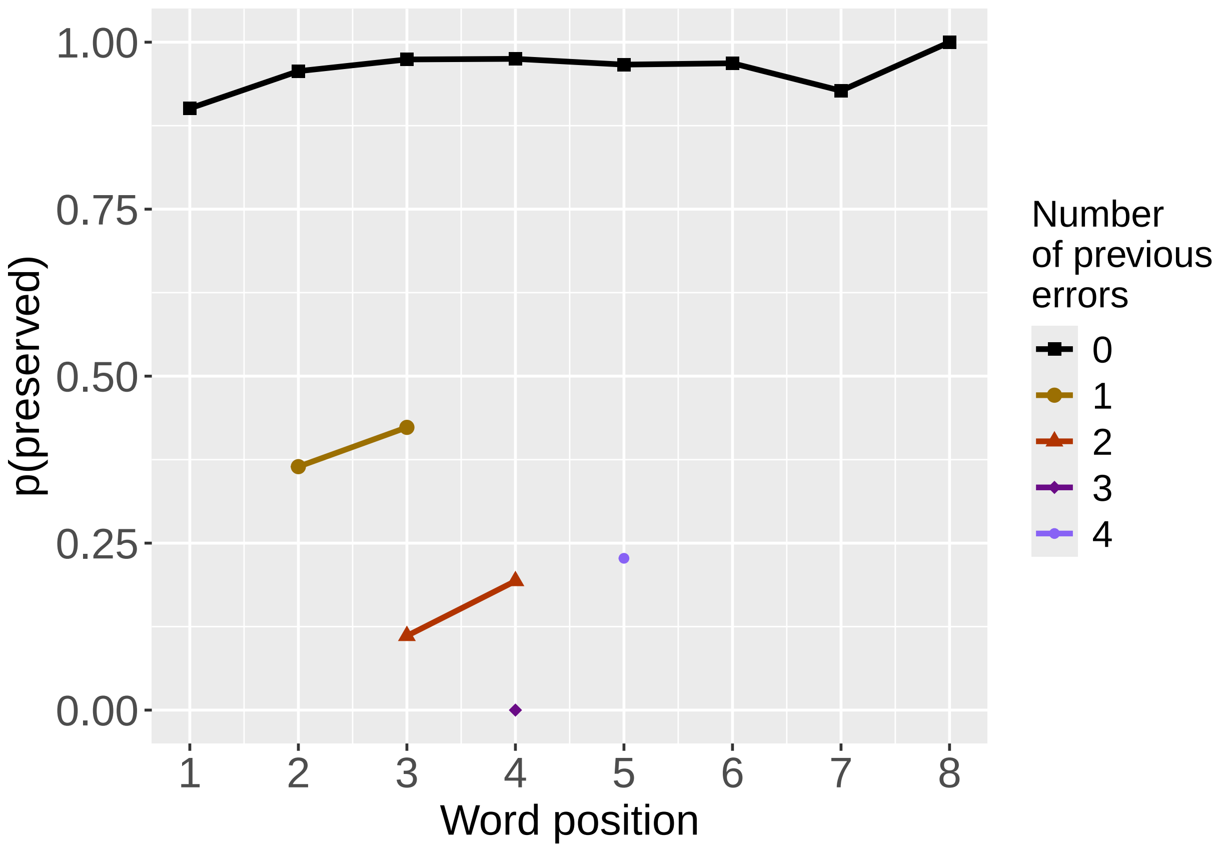
```r
print(PrevCorPlot)
```



```
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
```

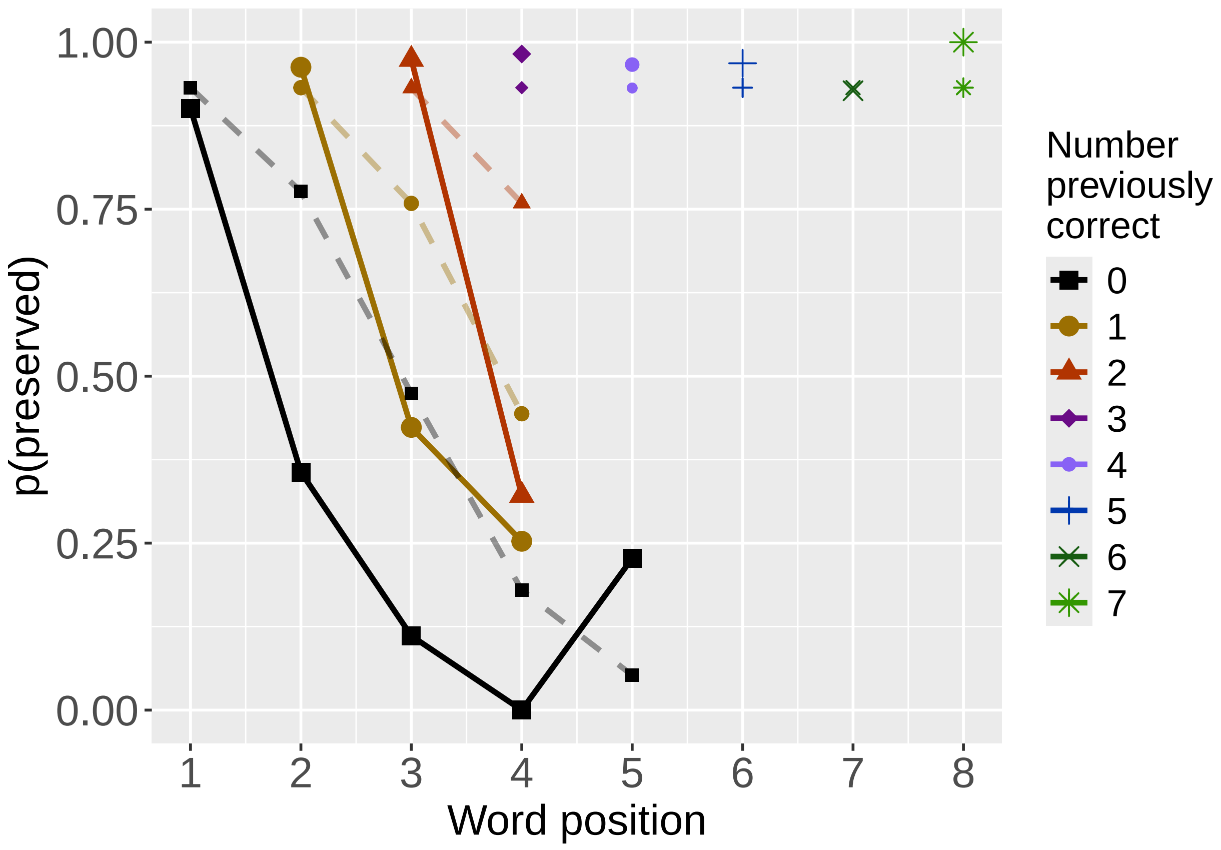
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

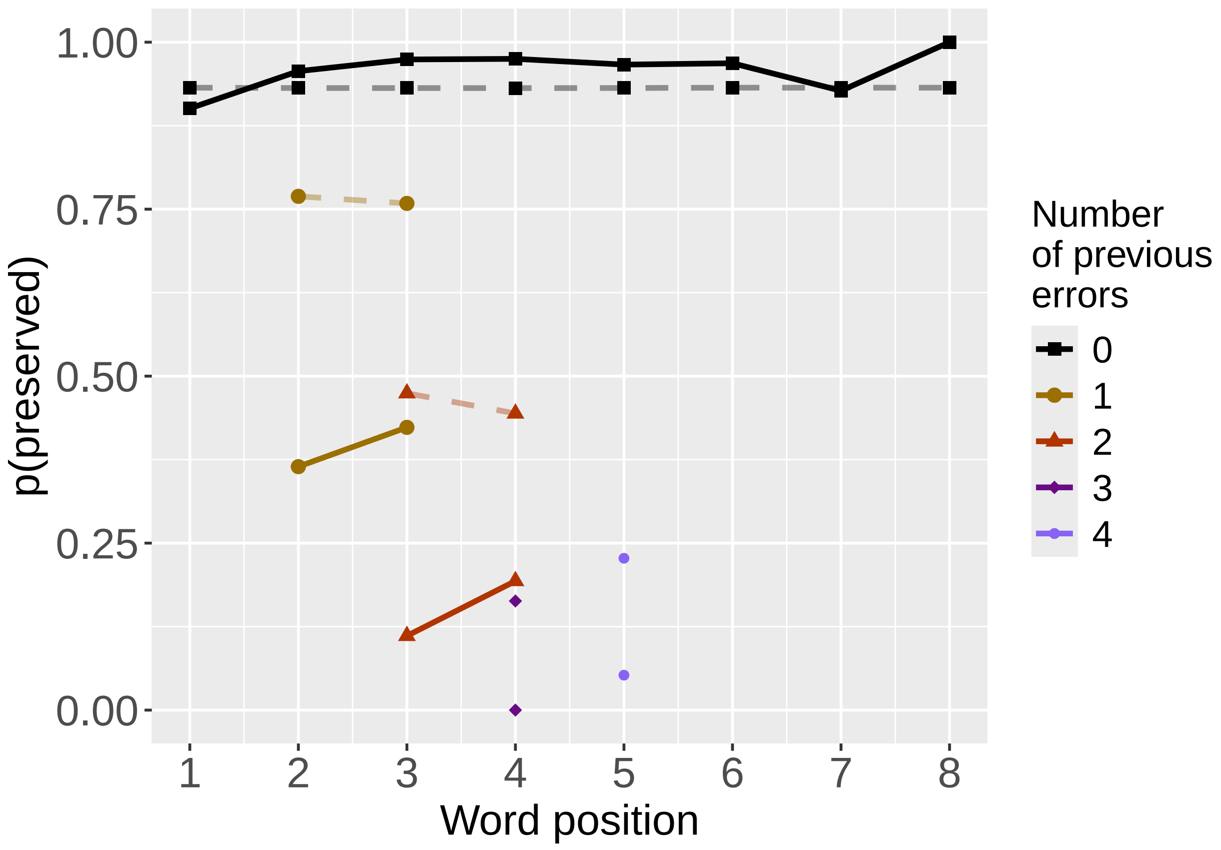
```
print(PrevCorPlot)
```



```
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
```

```r
CumAICSummary <- NULL
```

```r
########
# 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
##    2.239260    -1.612466     0.032111     0.002909
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 766.8     AIC: 839.4
## log likelihood:  -383.4047
## Nagelkerke R2:  0.4114768
## % pres/err predicted correctly:  -212.1906
## % of predictable range [ (model-null)/(1-null) ]:  0.3791798
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.616       -1.432
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 787   AIC: 856.4
## log likelihood:  -393.503
## Nagelkerke R2:  0.3914156
## % pres/err predicted correctly:  -218.9804
## % of predictable range [ (model-null)/(1-null) ]:  0.3594076
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.507068     0.008543    -0.222840
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1201
## log likelihood:  -558.8998
## Nagelkerke R2:  0.02011432
## % pres/err predicted correctly:  -338.2814
## % of predictable range [ (model-null)/(1-null) ]:  0.01199808
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	839.3582	0.00000	1.0000000	0.9997967	0.4114768	2.239260	-1.612466	0.0321110	0.0029094

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	856.3598	17.00166	0.0002033	0.0002033	0.3914156	2.615613	-1.431687	NA	NA
preserved ~ I(pos^2) + pos	1201.4951	362.13691	0.0000000	0.0000000	0.0201143	2.507068	NA	0.0085426	-0.2228404

```r
########
# 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
##       2.616       -1.432
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 787   AIC: 856.4
## log likelihood:  -393.503
## Nagelkerke R2:  0.3914156
## % pres/err predicted correctly:  -218.9804
## % of predictable range [ (model-null)/(1-null) ]:  0.3594076
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.41381     -1.44142      0.02961
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 786.7     AIC: 858.4
## log likelihood:  -393.3705
## Nagelkerke R2:  0.3916807
## % pres/err predicted correctly:  -218.7445
## % of predictable range [ (model-null)/(1-null) ]:  0.3600947
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##        3.12        -0.18
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1197
## log likelihood:  -559.0403
## Nagelkerke R2:  0.01976182
## % pres/err predicted correctly:  -338.1971
## % of predictable range [ (model-null)/(1-null) ]:  0.01224341
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	856.3598	0.000000	1.0000000	0.7383196	0.3914156	2.615613	-1.431687	NA
preserved ~ CumErr + stimlen	858.4344	2.074506	0.3544269	0.2616804	0.3916807	2.413815	-1.441423	0.0296144
preserved ~ stimlen	1197.1025	340.742658	0.0000000	0.0000000	0.0197618	3.120125	NA	-0.1800221

```
########
# 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
##      2.1458      -1.3751       0.2444
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 768.3     AIC: 838.8
## log likelihood:  -384.1719
## Nagelkerke R2:  0.4099624
## % pres/err predicted correctly:  -213.2251
## % of predictable range [ (model-null)/(1-null) ]:  0.3761673
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.616       -1.432
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 787   AIC: 856.4
## log likelihood:  -393.503
## Nagelkerke R2:  0.3914156
## % pres/err predicted correctly:  -218.9804
## % of predictable range [ (model-null)/(1-null) ]:  0.3594076
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.1853       0.3776
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1072  AIC: 1151
## log likelihood:  -535.8483
## Nagelkerke R2:  0.0770392
## % pres/err predicted correctly:  -327.0999
## % of predictable range [ (model-null)/(1-null) ]:  0.0445591
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	838.8179	0.00000	1.0000000	0.9998449	0.4099624	2.145818	-1.375060	0.2443613
preserved ~ CumErr	856.3598	17.54193	0.0001552	0.0001551	0.3914156	2.615613	-1.431687	NA
preserved ~ CumPres	1151.2444	312.42646	0.0000000	0.0000000	0.0770392	1.185251	NA	0.3776145

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

```r
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
##      1.9015      -1.6194       0.2444
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 768.3     AIC: 838.8
## log likelihood:  -384.1719
## Nagelkerke R2:  0.4099624
## % pres/err predicted correctly:  -213.2251
## % of predictable range [ (model-null)/(1-null) ]:  0.3761673
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.616       -1.432
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 787   AIC: 856.4
## log likelihood:  -393.503
## Nagelkerke R2:  0.3914156
## % pres/err predicted correctly:  -218.9804
## % of predictable range [ (model-null)/(1-null) ]:  0.3594076
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.3864      -0.1496
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 1118  AIC: 1200
## log likelihood:  -559.0287
## Nagelkerke R2:  0.01979099
## % pres/err predicted correctly:  -338.2881
## % of predictable range [ (model-null)/(1-null) ]:  0.01197868
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	838.8179	0.00000	1.0000000	0.9998449	0.4099624	1.901457	-1.619421	0.2443613
preserved ~ CumErr	856.3598	17.54193	0.0001552	0.0001551	0.3914156	2.615613	-1.431687	NA
preserved ~ pos	1199.7780	360.96011	0.0000000	0.0000000	0.0197910	2.386369	NA	-0.1495964

```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	838.8179	0.000000	1.0000000	0.9998449	0.4099624	2.145818	-1.375060	NA	NA	NA	0.2443613
preserved ~ CumErr + pos	838.8179	0.000000	1.0000000	0.9998449	0.4099624	1.901457	-1.619421	NA	0.2443613	NA	NA
preserved ~ CumErr + I(pos^2) + pos	839.3582	0.000000	1.0000000	0.9997967	0.4114768	2.239260	-1.612466	0.0321110	0.0029094	NA	NA
preserved ~ CumErr	856.3598	17.001650	0.0002033	0.0002033	0.3914156	2.615613	-1.431687	NA	NA	NA	NA
preserved ~ CumErr	856.3598	0.000000	1.0000000	0.7383196	0.3914156	2.615613	-1.431687	NA	NA	NA	NA
preserved ~ CumErr	856.3598	17.541930	0.0001552	0.0001551	0.3914156	2.615613	-1.431687	NA	NA	NA	NA
preserved ~ CumErr	856.3598	17.541930	0.0001552	0.0001551	0.3914156	2.615613	-1.431687	NA	NA	NA	NA
preserved ~ CumErr + stimlen	858.4344	2.074506	0.3544269	0.2616804	0.3916807	2.413815	-1.441423	NA	NA	0.0296144	NA
preserved ~ CumPres	1151.2443	312.426460	0.0000000	0.0000000	0.0770392	1.185251	NA	NA	NA	NA	0.3776145
preserved ~ stimlen	1197.1025	340.742658	0.0000000	0.0000000	0.0197618	3.120125	NA	NA	NA	-0.1800221	NA
preserved ~ pos	1199.7780	360.960106	0.0000000	0.0000000	0.0197910	2.386369	NA	NA	-0.1495964	NA	NA
preserved ~ I(pos^2) + pos	1201.4953	362.136901	0.0000000	0.0000000	0.0201143	2.507068	NA	0.0085426	-0.2228404	NA	NA

```r
# 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:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq
##      2.1330      -1.3356       0.2545       0.1726
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 760.4     AIC: 831.2
## log likelihood:  -380.2198
## Nagelkerke R2:  0.4177467
## % pres/err predicted correctly:  -211.4771
## % of predictable range [ (model-null)/(1-null) ]:  0.3812575
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres      stimlen     log_freq
##    2.116583    -1.336364     0.253784     0.002613     0.173613
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1456 Residual
## Null Deviance:       1134
## Residual Deviance: 760.4     AIC: 833.2
## log likelihood:  -380.2189
## Nagelkerke R2:  0.4177483
## % pres/err predicted correctly:  -211.472
## % of predictable range [ (model-null)/(1-null) ]:  0.3812723
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      2.1458      -1.3751       0.2444
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 768.3     AIC: 838.8
## log likelihood:  -384.1719
## Nagelkerke R2:  0.4099624
## % pres/err predicted correctly:  -213.2251
## % of predictable range [ (model-null)/(1-null) ]:  0.3761673
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres      stimlen
##     2.52788     -1.35119      0.26164     -0.06111
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 767.3     AIC: 839
## log likelihood:  -383.6589
## Nagelkerke R2:  0.4109752
## % pres/err predicted correctly:  -213.1602
## % of predictable range [ (model-null)/(1-null) ]:  0.3763564
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       1.835
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1460 Residual
## Null Deviance:       1134
## Residual Deviance: 1134  AIC: 1213
## log likelihood:  -566.8741
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -342.4016
## % 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 + log_freq	831.2005	0.000000	1.0000000	0.7112189	0.4177467	2.132995	-1.335608	0.2544609	0.1726201	NA
preserved ~ CumErr + CumPres + stimlen + log_freq	833.2238	2.023298	0.3636189	0.2586126	0.4177483	2.116583	-1.336364	0.2537843	0.1736130	0.0026133
preserved ~ CumErr + CumPres	838.8179	7.617455	0.0221764	0.0157723	0.4099624	2.145818	-1.375060	0.2443613	NA	NA
preserved ~ CumErr + CumPres + stimlen	839.0005	7.800023	0.0202417	0.0143963	0.4109752	2.527880	-1.351186	0.2616423	NA	-0.0611071
preserved ~ 1	1212.7408	381.540308	0.0000000	0.0000000	0.0000000	1.834601	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!
```

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

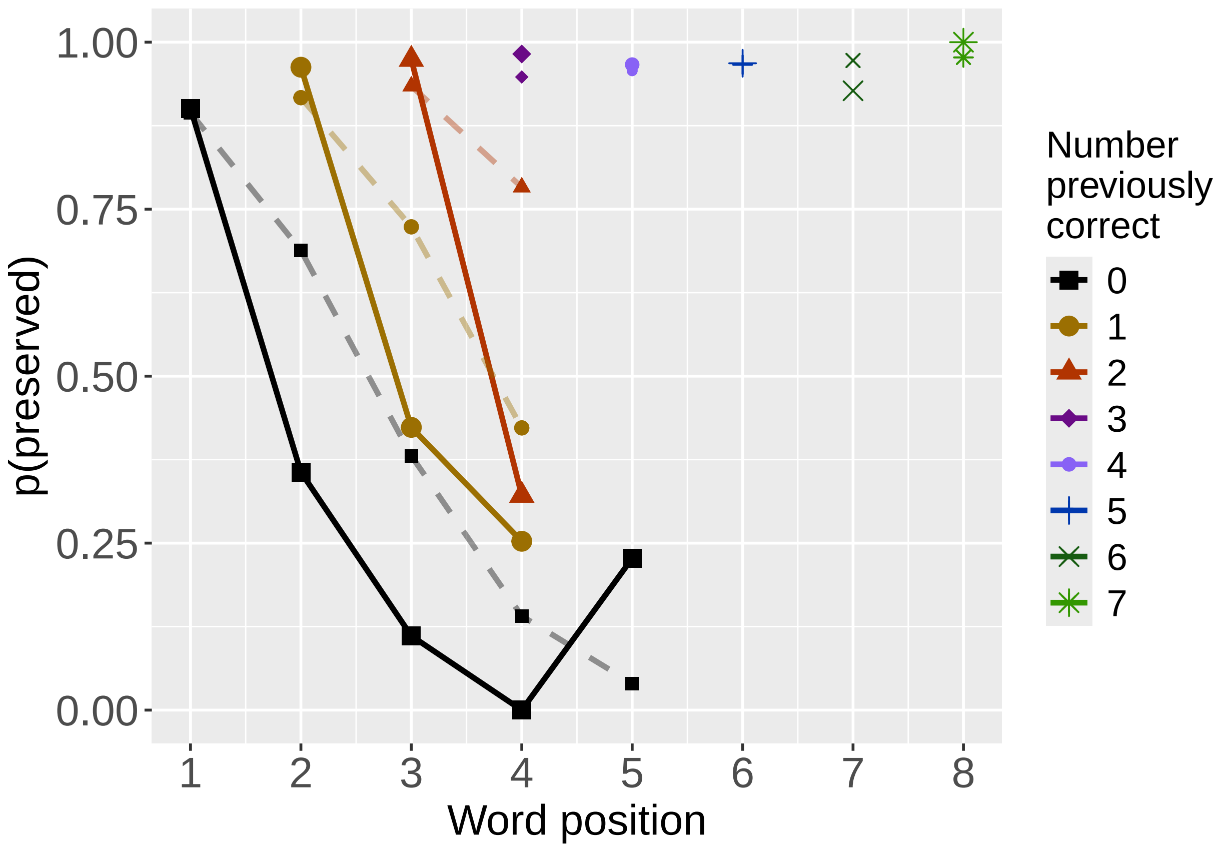
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + CumPres + log_freq
##          Df Deviance     AIC
## CumErr    1  1023.99 1092.75
## CumPres   1   780.97  849.73
## log_freq  1   768.34  837.10
## <none>        760.44  831.20
```

```
################################
# 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.
```

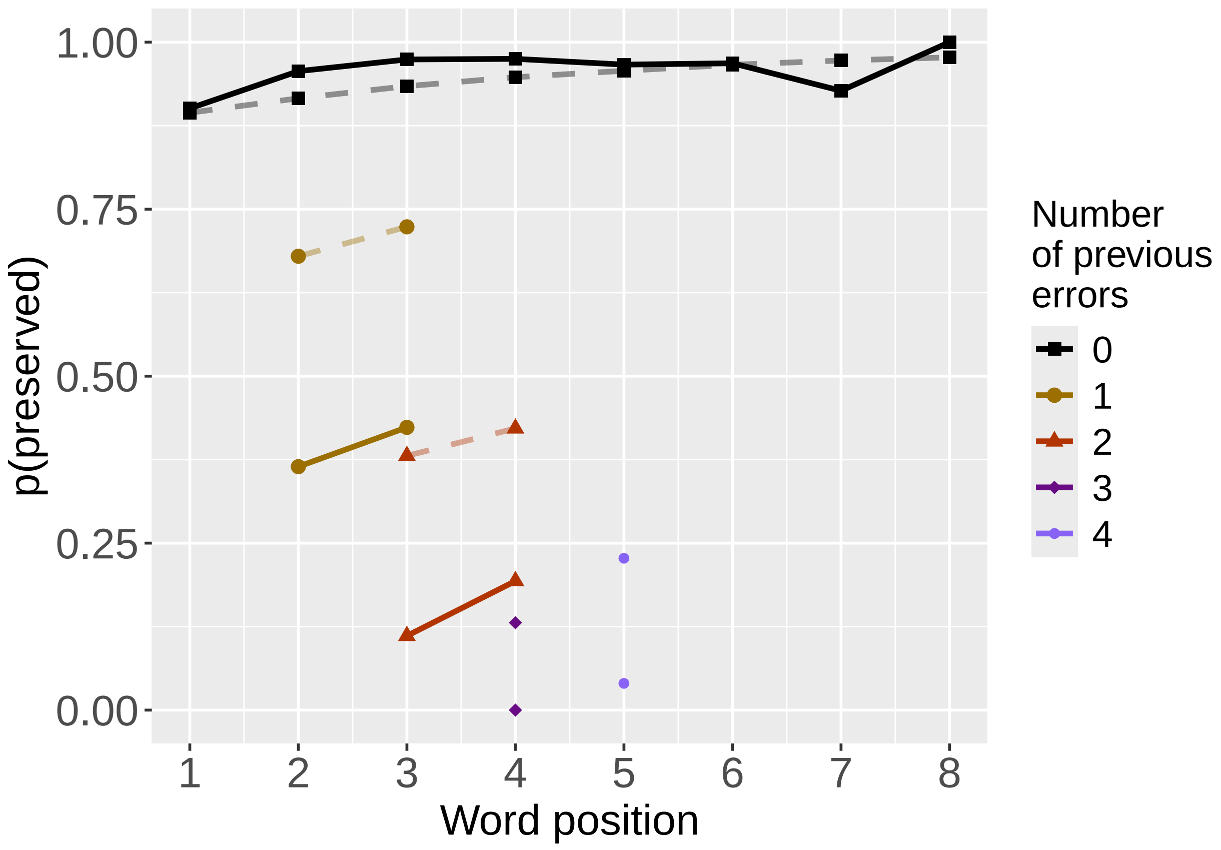
```
print(PrevCorPlot)
```



```
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
```

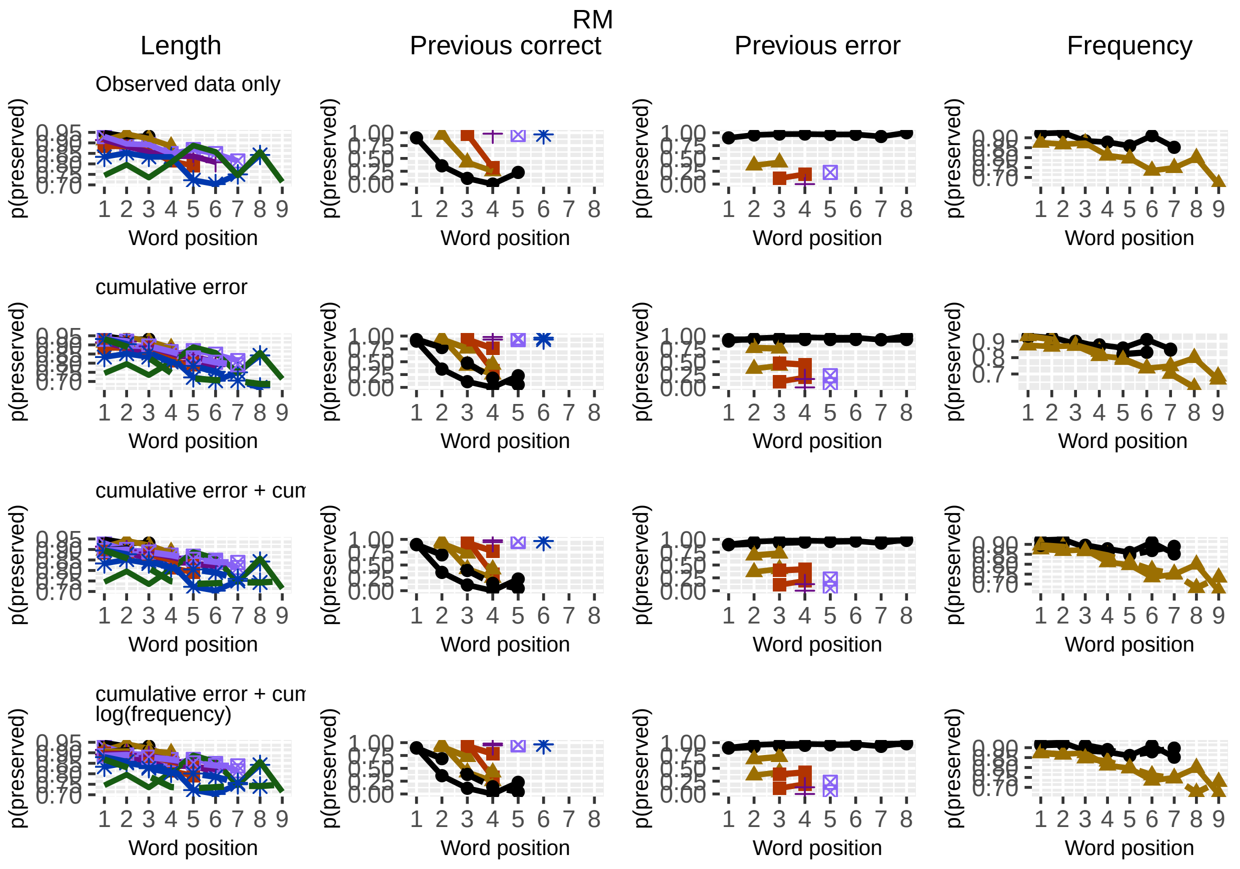
```
if(DoSimulations){
  BestModelFormulaL3Rnd <- BestModelFormulaL3
  if(grepl("CumPres",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumPres","RndCumPres",BestModelFormulaL3Rnd)
  }
  if(grepl("CumErr",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumErr","RndCumErr",BestModelFormulaL3Rnd)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

  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!
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.616       -1.432
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1459 Residual
## Null Deviance:       1134
## Residual Deviance: 787   AIC: 856.4
## log likelihood:  -393.503
## Nagelkerke R2:  0.3914156
## % pres/err predicted correctly:  -218.9804
## % of predictable range [ (model-null)/(1-null) ]:  0.3594076
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      2.1458      -1.3751       0.2444
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1458 Residual
## Null Deviance:       1134
## Residual Deviance: 768.3     AIC: 838.8
## log likelihood:  -384.1719
## Nagelkerke R2:  0.4099624
## % pres/err predicted correctly:  -213.2251
## % of predictable range [ (model-null)/(1-null) ]:  0.3761673
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq
##      2.1330      -1.3356       0.2545       0.1726
##
## Degrees of Freedom: 1460 Total (i.e. Null);  1457 Residual
## Null Deviance:       1134
## Residual Deviance: 760.4     AIC: 831.2
## log likelihood:  -380.2198
## Nagelkerke R2:  0.4177467
## % pres/err predicted correctly:  -211.4771
## % of predictable range [ (model-null)/(1-null) ]:  0.3812575
## ***************************
## `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 8 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 2 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 8 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 2 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 8 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 2 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 8 values. Consider specifying shapes manually if you need that many have
##   them.
## Warning: Removed 2 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 2 rows containing missing values or values outside the scale range (`geom_point()`).
```

```
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!
```

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

	CumErr	CumPres	log_freq
McFadden	0.2611773	0.0355996	0.0233084
SquaredCorrelation	0.1877217	0.0271510	0.0181184
Nagelkerke	0.3332024	0.0481924	0.0321599
Estrella	0.2238788	0.0301304	0.0196277

	deviance	deviance_explained
CumErr + CumPres + log_freq	760.4395	373.3087
CumErr + CumPres	768.3438	365.4044
CumErr	787.0060	346.7422
null	1133.7483	0.0000

```
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!
```

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

```
##                                                   model  deviance deviance_explained
## CumErr + CumPres + log_freq CumErr + CumPres + log_freq  760.4395           373.3087
## CumErr + CumPres                       CumErr + CumPres  768.3438           365.4044
## CumErr                                           CumErr  787.0060           346.7422
## null                                               null 1133.7483             0.0000
##                             percent_explained percent_of_explained_deviance
## CumErr + CumPres + log_freq          32.92695                     100.00000
## CumErr + CumPres                     32.22977                      97.88263
## CumErr                               30.58371                      92.88350
## null                                  0.00000                            NA
##                             increment_in_explained
## CumErr + CumPres + log_freq               2.117374
## CumErr + CumPres                          4.999128
## CumErr                                   92.883498
## null                                      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")
```

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + CumPres + log_freq	32.92695	100.00000	2.117374
CumErr + CumPres	32.22977	97.88263	4.999128
CumErr	30.58371	92.88350	92.883498
null	0.00000	NA	0.000000

```
NagContributions <- t(DAContributionAverage[3,])
NagTotal<-sum(NagContributions)
NagPercents <- NagContributions / NagTotal
NagResultTable <- data.frame(NagContributions = NagContributions, NagPercents = NagPercents)
names(NagResultTable)<-c("NagContributions","NagPercents")
write.csv(NagResultTable,paste0(TablesDir,CurPat,"_",CurTask,"_nagelkerke_contributions_table.csv"),row.names = TRUE)
NagPercents
```

```
##           Nagelkerke
## CumErr    0.80570334
## CumPres   0.11653220
## log_freq  0.07776446
```

```
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
```

```
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       0.8515390       787.0060  0.00000000
## 2           preserved ~ CumErr+CumPres       0.9033475       768.3438  0.05180856
## 3 preserved ~ CumErr+CumPres+log_freq       0.9046307       760.4395  0.05309172
```

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.8515390	787.0060
preserved ~ CumErr+CumPres	0.9033475	768.3438
preserved ~ CumErr+CumPres+log_freq	0.9046307	760.4395

model	diff_CumErr	diff_CumErr+CumPres	diff_CumErr+CumPres+log_freq
preserved ~ CumErr	0.0000000	-0.0518086	-0.0530917
preserved ~ CumErr+CumPres	0.0518086	0.0000000	-0.0012832
preserved ~ CumErr+CumPres+log_freq	0.0530917	0.0012832	0.0000000

```
##   diff_CumErr+CumPres diff_CumErr+CumPres+log_freq
## 1        -0.051808562                 -0.053091720
## 2         0.000000000                 -0.001283159
## 3         0.001283159                  0.000000000
```

```
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")
```