AV - naming - Serial position analysis (v5)

```
# 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
```

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

```
# 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
```

```
# 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())
```

```
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(sy

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	91	10	11	NA	NA	112
2	15	NA	80	5	12	112
3	36	NA	26	49	1	112
4	66	NA	19	9	4	98
5	18	NA	34	9	3	64
6	32	NA	10	10	4	56
7	19	NA	5	1	1	26
8	4	NA	4	2	NA	10
9	5	NA	NA	NA	NA	5

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8125000	0.0892857	0.0982143	NA	NA	112
2	0.1339286	NA	0.7142857	0.0446429	0.1071429	112
3	0.3214286	NA	0.2321429	0.4375000	0.0089286	112
4	0.6734694	NA	0.1938776	0.0918367	0.0408163	98
5	0.2812500	NA	0.5312500	0.1406250	0.0468750	64
6	0.5714286	NA	0.1785714	0.1785714	0.0714286	56

pos_factor	O	P	V	1	S	total
7	0.7307692	NA	0.1923077	0.0384615	0.0384615	26
8	0.4000000	NA	0.4000000	0.2000000	NA	10
9	1.0000000	NA	NA	NA	NA	5

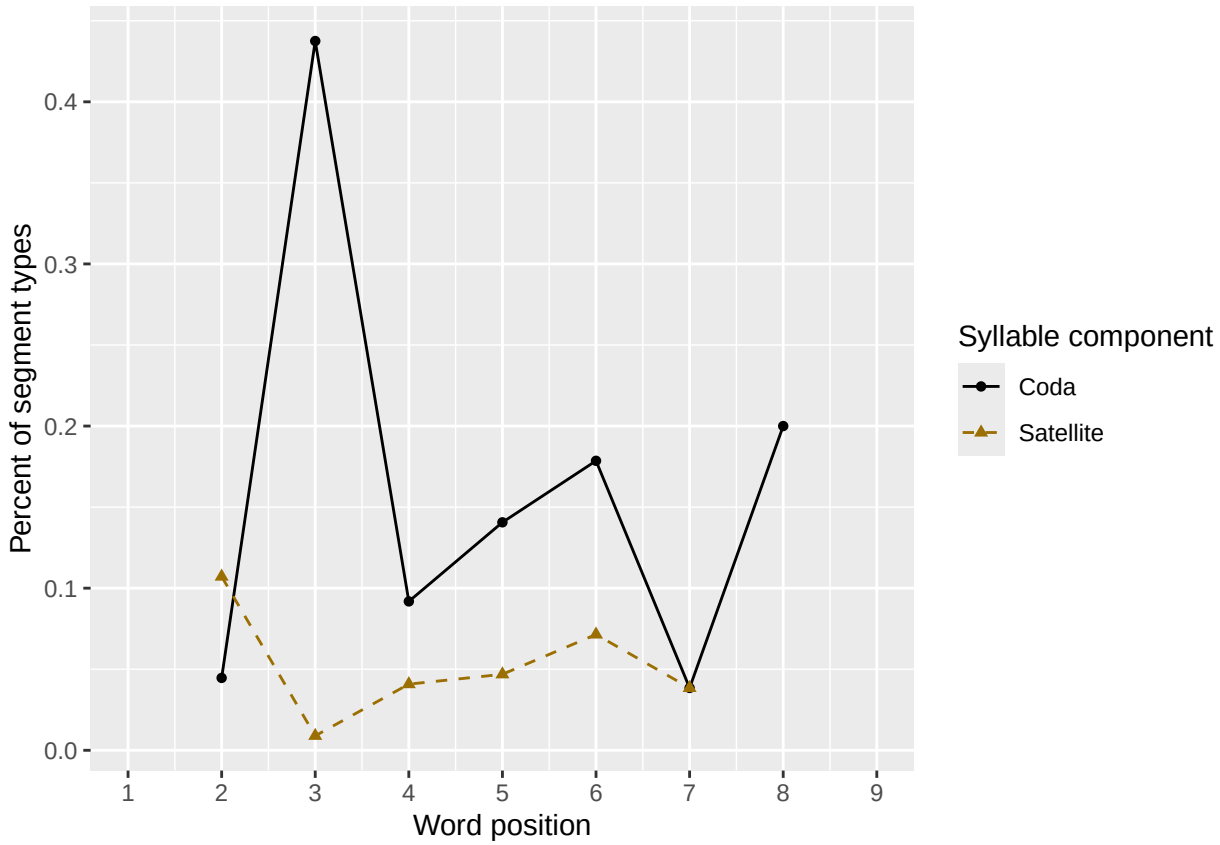
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 5 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
syll_comp_plot
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range (`geom_line()`).
## Removed 5 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 0.786 0.75  0.714 NA     NA     NA     NA        NA    NA
## 2       5 0.765 0.765 0.824  0.794 NA     NA     NA        NA    NA
## 3       6 0.75  0.875 0.875  1      0.875 NA     NA        NA    NA
## 4       7 0.55  0.65  0.828  0.811  0.867  0.85  NA        NA    NA
## 5       8 0.688 0.844 0.812  0.875  0.938  0.906  0.875    NA    NA
## 6       9 0.8   1     0.6    0.8    1      0.8    1         1    NA
## 7      10 0.6   0.8   0.4    0.4    0.6    0.8    0.8       1     1
```

```
# 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.
```

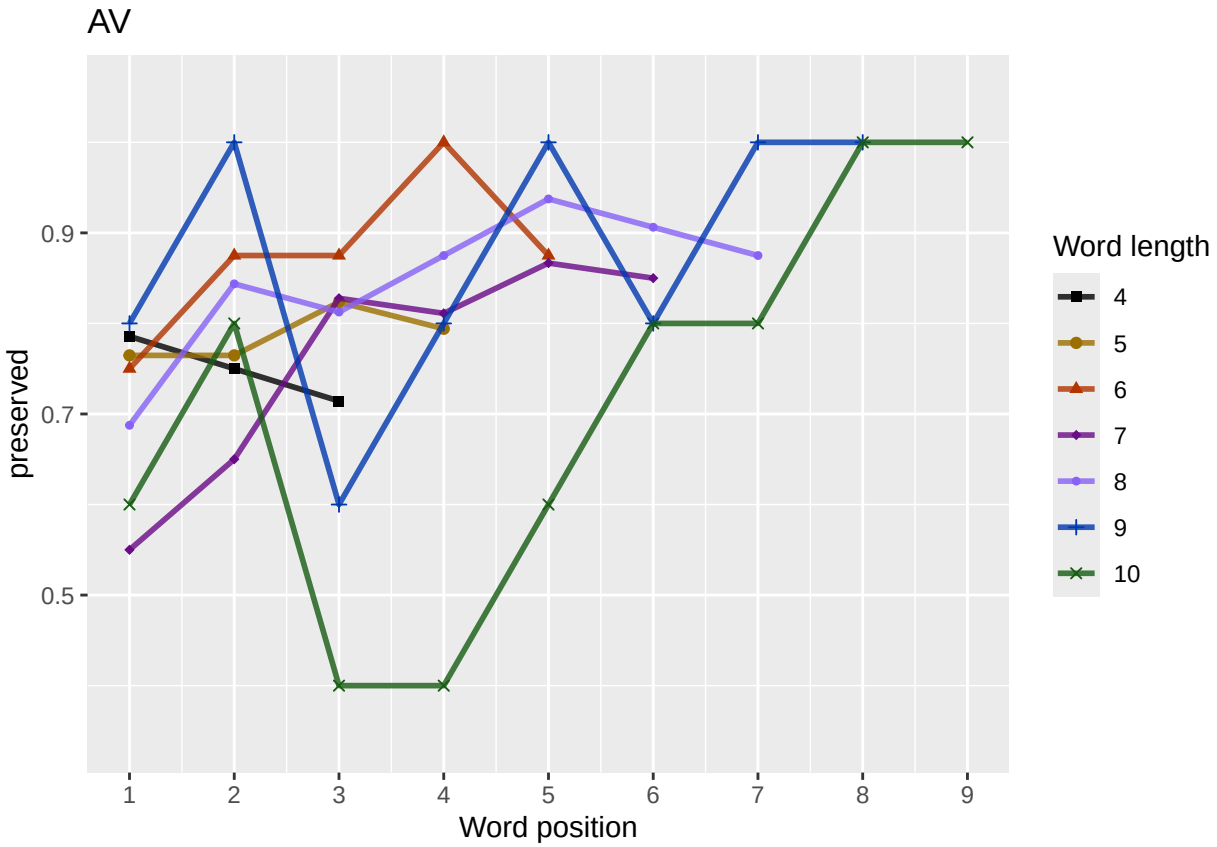
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    14    14    14    NA    NA    NA    NA    NA    NA
## 2       5    34    34    34    34    NA    NA    NA    NA    NA
## 3       6     8     8     8     8     8    NA    NA    NA    NA
## 4       7    30    30    30    30    30    30    NA    NA    NA
## 5       8    16    16    16    16    16    16    16    NA    NA
## 6       9     5     5     5     5     5     5     5     5    NA
## 7      10     5     5     5     5     5     5     5     5     5
```

```
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:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.6049       0.2345
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.9     AIC: 604.4
## log likelihood:  -288.441
## Nagelkerke R2:  0.04415526
## % pres/err predicted correctly:  -186.9453
## % of predictable range [ (model-null)/(1-null) ]:  0.02735039
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     1.12599     -0.09047      0.26335
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 575   AIC: 604.5
## log likelihood:  -287.5124
## Nagelkerke R2:  0.0489543
## % pres/err predicted correctly:  -186.1927
## % of predictable range [ (model-null)/(1-null) ]:  0.03124523
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.34616     -0.09748      0.01867      0.13108
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.6     AIC: 606.1
## log likelihood:  -287.3214
## Nagelkerke R2:  0.04993965
## % pres/err predicted correctly:  -186.2143
## % of predictable range [ (model-null)/(1-null) ]:  0.03113345
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.71027      0.01102      0.15539
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.7     AIC: 606.3
## log likelihood:  -288.3729
## Nagelkerke R2:  0.0445081
## % pres/err predicted correctly:  -186.996
## % of predictable range [ (model-null)/(1-null) ]:  0.02708833
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.35344     -0.12159      0.17540      0.01161
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.9     AIC: 606.4
## log likelihood:  -287.4665
## Nagelkerke R2:  0.04919102
## % pres/err predicted correctly:  -186.1987
## % of predictable range [ (model-null)/(1-null) ]:  0.03121422
## ***************************
## 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)
##        0.6206982         0.0004696        -0.0667706         0.6955866         0.0108288
##      stimlen:pos
##       -0.0738150
##
## Degrees of Freedom: 594 Total (i.e. Null);  589 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.3     AIC: 609.7
## log likelihood:  -287.1633
## Nagelkerke R2:  0.05075454
## % pres/err predicted correctly:  -186.1823
## % of predictable range [ (model-null)/(1-null) ]:  0.03129909
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##       1.347
##
## Degrees of Freedom: 594 Total (i.e. Null);  594 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.7     AIC: 619.2
## log likelihood:  -296.8519
## Nagelkerke R2:  -3.517158e-16
## % pres/err predicted correctly:  -192.2303
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.20314      0.02109
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.6     AIC: 621.1
## log likelihood:  -296.7931
## Nagelkerke R2:  0.0003134394
## % pres/err predicted correctly:  -192.1926
## % of predictable range [ (model-null)/(1-null) ]:  0.0001949289
## ***************************
```

```
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 ~ pos	604.4258	0.0000000	1.0000000	0.3075028	0.0044155	0.6048553	NA	0.2344932	NA	NA	NA
preserved ~ stimlen + pos	604.4938	0.0680297	0.9657572	0.2972191	0.0104895	0.1259886	0.0904708	0.2633501	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	606.0842	1.6583958	0.4363992	0.1341940	0.0049937	0.3461616	0.0974772	0.1310831	NA	0.0186730	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	606.2778	1.8519325	0.3961485	0.1218168	0.0445087	1.7102739	NA	0.1553933	NA	0.0110162	NA
preserved ~ stimlen * pos	606.3583	1.9325098	0.3805054	0.1170006	0.0491911	0.3534382	- 0.1215904	0.1754023	0.0116093	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	609.7015	5.2754908	0.0715223	0.0219983	0.0507545	1.6206982	2.0004696	0.6955866	- 0.0738150	- 0.0667706	0.0108288
preserved ~ 1	619.1749	14.7490526	0.0006270	0.0001928	0.0000000	0.3465036	NA	NA	NA	NA	NA
preserved ~ stimlen	621.0718	16.6459614	0.0002429	0.0000747	0.0003134	0.2031372	0.0210938	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.6049       0.2345
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.9     AIC: 604.4
```

```
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.
```

```
# 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.
```

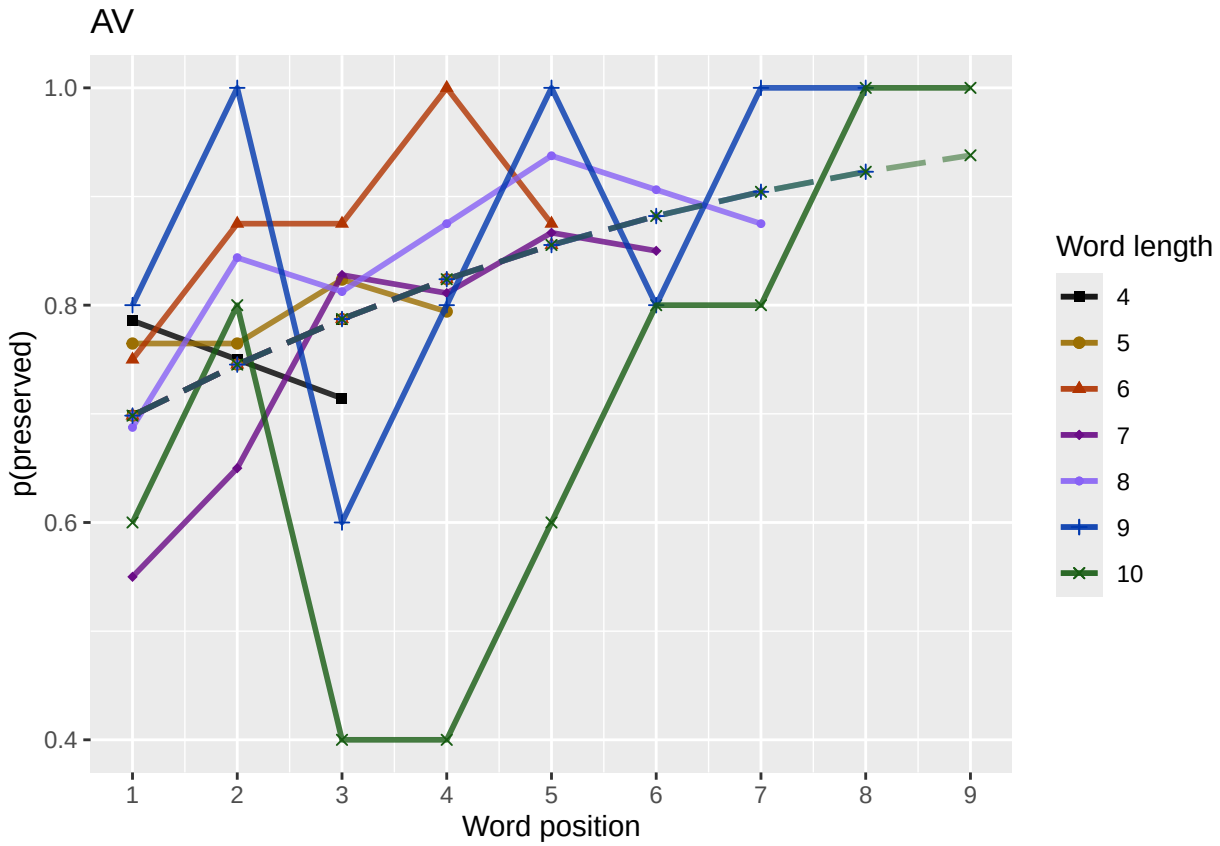
```
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # 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 0.698 0.745 0.787 NA     NA     NA     NA     NA     NA
## 2       5 0.698 0.745 0.787  0.824 NA     NA     NA     NA     NA
## 3       6 0.698 0.745 0.787  0.824  0.855 NA     NA     NA     NA
## 4       7 0.698 0.745 0.787  0.824  0.855  0.882 NA     NA     NA
## 5       8 0.698 0.745 0.787  0.824  0.855  0.882  0.904 NA     NA
## 6       9 0.698 0.745 0.787  0.824  0.855  0.882  0.904  0.923 NA
## 7      10 0.698 0.745 0.787  0.824  0.855  0.882  0.904  0.923  0.938
```

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

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

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

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



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

```
# first number responses, then count resp with fragments - below we will eliminate fragments
# and re-run models
```

```r
# 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        2   112
```

```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 2 / 112 = 1.79 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:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
```

```
##      1.1896      -0.1051       0.2850
##
## Degrees of Freedom: 591 Total (i.e. Null);  589 Residual
## Null Deviance:       584.2
## Residual Deviance: 563   AIC: 592.7
## log likelihood:  -281.5052
## Nagelkerke R2:  0.05600133
## % pres/err predicted correctly:  -181.7504
## % of predictable range [ (model-null)/(1-null) ]:  0.03544604
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.5815       0.2518
##
## Degrees of Freedom: 591 Total (i.e. Null);  590 Residual
## Null Deviance:       584.2
## Residual Deviance: 565.5     AIC: 593.3
## log likelihood:  -282.7365
## Nagelkerke R2:  0.04958913
## % pres/err predicted correctly:  -182.7076
## % of predictable range [ (model-null)/(1-null) ]:  0.0303937
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.45190     -0.11312      0.02293      0.12409
##
## Degrees of Freedom: 591 Total (i.e. Null);  588 Residual
## Null Deviance:       584.2
## Residual Deviance: 562.5     AIC: 594.2
## log likelihood:  -281.2368
## Nagelkerke R2:  0.05739564
## % pres/err predicted correctly:  -181.777
## % of predictable range [ (model-null)/(1-null) ]:  0.03530561
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.30673     -0.12113      0.23907      0.00605
##
## Degrees of Freedom: 591 Total (i.e. Null);  588 Residual
```

```
## Null Deviance:       584.2
## Residual Deviance: 563   AIC: 594.7
## log likelihood:  -281.4932
## Nagelkerke R2:  0.056064
## % pres/err predicted correctly:  -181.7616
## % of predictable range [ (model-null)/(1-null) ]:  0.03538708
## ****************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.71363      0.01413      0.15150
##
## Degrees of Freedom: 591 Total (i.e. Null);  589 Residual
## Null Deviance:       584.2
## Residual Deviance: 565.3     AIC: 595.1
## log likelihood:  -282.6323
## Nagelkerke R2:  0.05013247
## % pres/err predicted correctly:  -182.7752
## % of predictable range [ (model-null)/(1-null) ]:  0.03003709
## ****************************
## 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)
##          0.50709           0.02192          -0.06329           0.78149           0.01166
##      stimlen:pos
##         -0.09143
##
## Degrees of Freedom: 591 Total (i.e. Null);  586 Residual
## Null Deviance:       584.2
## Residual Deviance: 562   AIC: 597.7
## log likelihood:  -281.0105
## Nagelkerke R2:  0.05857021
## % pres/err predicted correctly:  -181.7031
## % of predictable range [ (model-null)/(1-null) ]:  0.03569577
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.371
##
## Degrees of Freedom: 591 Total (i.e. Null);  591 Residual
## Null Deviance:       584.2
```

```
## Residual Deviance: 584.2     AIC: 609.9
## log likelihood:  -292.0894
## Nagelkerke R2:  -3.540094e-16
## % pres/err predicted correctly:  -188.4662
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.27589      0.01401
##
## Degrees of Freedom: 591 Total (i.e. Null);  590 Residual
## Null Deviance:       584.2
## Residual Deviance: 584.1     AIC: 611.8
## log likelihood:  -292.0639
## Nagelkerke R2:  0.0001372079
## % pres/err predicted correctly:  -188.4505
## % of predictable range [ (model-null)/(1-null) ]:  8.296026e-05
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	592.7457	0.0000000	1.0000000	0.3304682	0.0560013	1.1895729	-0.1051192	0.2849598	NA	NA	NA
preserved ~ pos	593.2831	0.5373166	0.7644044	0.2526103	0.0495891	0.5815254	NA	0.2517731	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	594.1846	1.4388603	0.4870297	0.1609478	0.0573956	1.4519034	-0.1131185	0.1240929	NA	0.0229338	NA
preserved ~ stimlen * pos	594.6977	1.9519886	0.3768177	0.1245262	0.0560640	1.3067322	-0.1211304	0.2390707	0.0060502	NA	NA
preserved ~ I(pos^2) + pos	595.0652	2.3194812	0.3135677	0.1036240	0.0501325	1.7136301	NA	0.1514970	NA	0.0141288	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)	597.7014	4.955636	0.0839261	0.0277349	0.9058570	0.2507091	7.021920	8.781490	1 - 0.0914278	- 0.0632921	0.0116632
preserved ~ 1	609.8558	17.11006	3.82E-04	1.26E-06	0.0000000	0.0000000	0.3712301	NA	NA	NA	NA	NA
preserved ~ stimlen	611.8135	19.06772	1.41E-04	4.66E-07	0.0000137	0.1275894	0.0140071	NA	NA	NA	NA	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
```

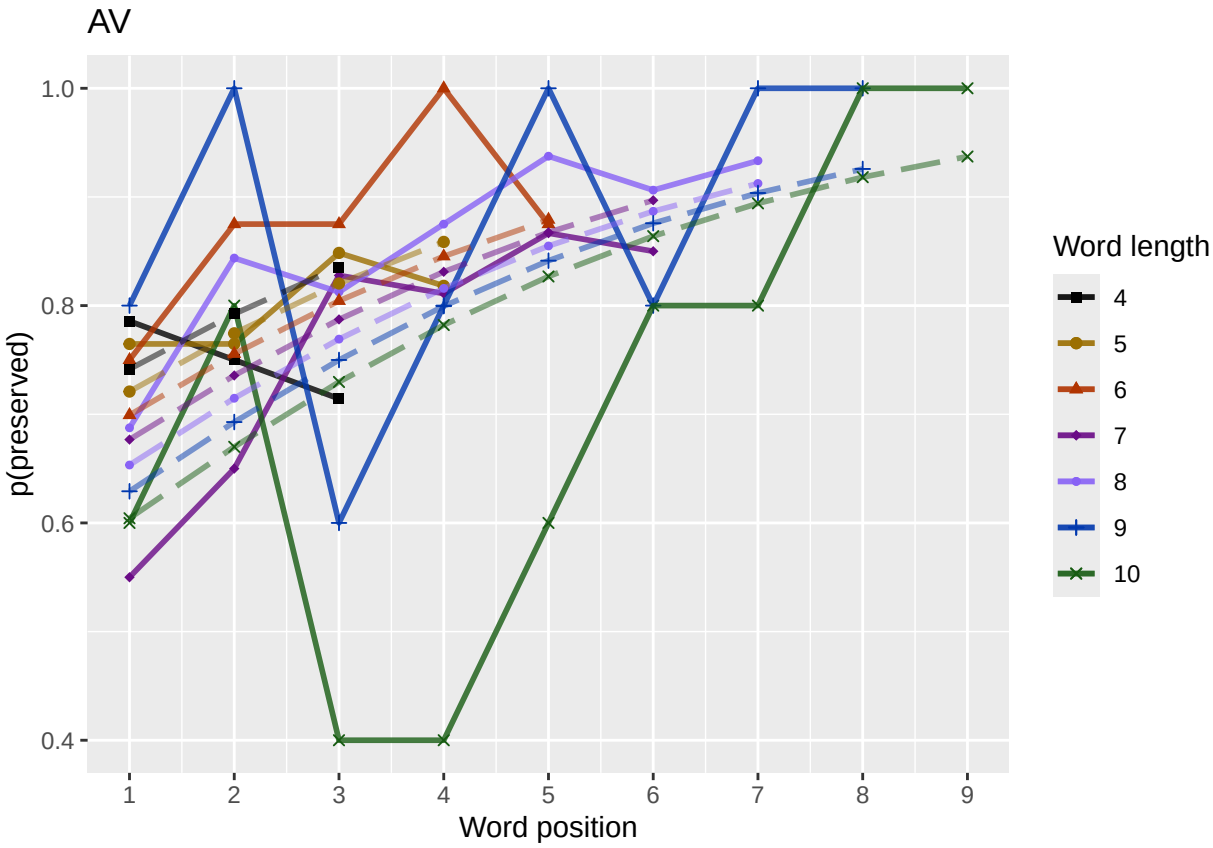
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.742 0.792 0.835 NA     NA     NA     NA     NA     NA
## 2       5 0.721 0.774 0.820  0.859 NA     NA     NA     NA     NA
## 3       6 0.699 0.756 0.804  0.845  0.879 NA     NA     NA     NA
## 4       7 0.677 0.736 0.787  0.831  0.867  0.897 NA     NA     NA
## 5       8 0.653 0.715 0.769  0.816  0.855  0.887  0.912 NA     NA
## 6       9 0.629 0.693 0.750  0.800  0.841  0.876  0.904  0.926 NA
## 7      10 0.604 0.670 0.730  0.782  0.827  0.864  0.894  0.918  0.937
```

```
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: 0.34 - 1.06"
```

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

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

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

# in case min is close to the end or we are not using a min (for non-U-shaped)
```

```
# functions, we need to get differences _first_ and then average those (e.g.
# 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: "
```

```
print(OA_mean_len_diff)
```

```
## [1] 0
```

```
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): "
```

```
print(OA_mean_pos_diff)
```

```
## [1] 0.02994939
```

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

```
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:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)          pos      log_freq  pos:log_freq
##      0.40653      0.32404       0.16660       0.04546
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 554.3     AIC: 584.3
## log likelihood:  -277.1507
## Nagelkerke R2:  0.1014979
## % pres/err predicted correctly:  -179.3478
## % of predictable range [ (model-null)/(1-null) ]:  0.06666887
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      0.4832       0.2872       0.3065
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 556   AIC: 584.3
## log likelihood:  -278.0195
```

```
## Nagelkerke R2:  0.09716222
## % pres/err predicted correctly:  -179.622
## % of predictable range [ (model-null)/(1-null) ]:  0.06524977
## **************************
## 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
##           0.71708            0.03075            0.10161           -0.04366           -0.01439
##      pos:log_freq
##           0.17664
##
## Degrees of Freedom: 594 Total (i.e. Null);  589 Residual
## Null Deviance:       593.7
## Residual Deviance: 551.4     AIC: 584.9
## log likelihood:  -275.7172
## Nagelkerke R2:  0.1086243
## % pres/err predicted correctly:  -178.6002
## % of predictable range [ (model-null)/(1-null) ]:  0.07053805
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      0.05886       0.05854       0.30990       0.19561       0.04488
##
## Degrees of Freedom: 594 Total (i.e. Null);  590 Residual
## Null Deviance:       593.7
## Residual Deviance: 553.7     AIC: 585.6
## log likelihood:  -276.845
## Nagelkerke R2:  0.1030208
## % pres/err predicted correctly:  -179.2041
## % of predictable range [ (model-null)/(1-null) ]:  0.06741266
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     0.11811      0.06143      0.27260      0.33486
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 555.4     AIC: 585.6
## log likelihood:  -277.683
## Nagelkerke R2:  0.0988432
```

```
## % pres/err predicted correctly:  -179.4738
## % of predictable range [ (model-null)/(1-null) ]:  0.06601661
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           0.43062            0.04303            0.02720            0.11523           -0.02142
## I(pos^2):log_freq       pos:log_freq
##          -0.01474            0.17710
##
## Degrees of Freedom: 594 Total (i.e. Null);  588 Residual
## Null Deviance:       593.7
## Residual Deviance: 551.1     AIC: 586.6
## log likelihood:  -275.5557
## Nagelkerke R2:  0.1094247
## % pres/err predicted correctly:  -178.5181
## % of predictable range [ (model-null)/(1-null) ]:  0.07096282
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          0.07013           0.07240           0.17806           0.27348           0.02235
##
## Degrees of Freedom: 594 Total (i.e. Null);  590 Residual
## Null Deviance:       593.7
## Residual Deviance: 555   AIC: 587
## log likelihood:  -277.4871
## Nagelkerke R2:  0.09982061
## % pres/err predicted correctly:  -179.4814
## % of predictable range [ (model-null)/(1-null) ]:  0.06597741
## **************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     0.36111      0.05362      0.02032      0.12788      0.33560
##
## Degrees of Freedom: 594 Total (i.e. Null);  590 Residual
## Null Deviance:       593.7
## Residual Deviance: 554.9     AIC: 587.1
## log likelihood:  -277.4537
## Nagelkerke R2:  0.09998735
## % pres/err predicted correctly:  -179.4464
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.06615843
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         0.050905          0.060960          0.168933          0.308676          0.004622
##     log_freq:pos
##         0.043018
##
## Degrees of Freedom: 594 Total (i.e. Null);  589 Residual
## Null Deviance:       593.7
## Residual Deviance: 553.7     AIC: 587.5
## log likelihood:  -276.8378
## Nagelkerke R2:  0.1030565
## % pres/err predicted correctly:  -179.2151
## % of predictable range [ (model-null)/(1-null) ]:  0.06735564
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          0.33800           0.06544           0.15261           0.02301           0.10937
## stimlen:log_freq
##          0.02616
##
## Degrees of Freedom: 594 Total (i.e. Null);  589 Residual
## Null Deviance:       593.7
## Residual Deviance: 554.4     AIC: 588.3
## log likelihood:  -277.1928
## Nagelkerke R2:  0.1012881
## % pres/err predicted correctly:  -179.4231
## % of predictable range [ (model-null)/(1-null) ]:  0.06627945
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          0.409879           0.047877          -0.077806           0.026479           0.116976
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##          0.008935          -0.015279           0.177261
##
## Degrees of Freedom: 594 Total (i.e. Null);  587 Residual
## Null Deviance:       593.7
## Residual Deviance: 551.1     AIC: 588.5
```

```
## log likelihood:  -275.5293
## Nagelkerke R2:  0.1095559
## % pres/err predicted correctly:  -178.5201
## % of predictable range [ (model-null)/(1-null) ]:  0.07095234
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      0.2255       0.1726       0.3230
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.6     AIC: 602.8
## log likelihood:  -287.2766
## Nagelkerke R2:  0.05017045
## % pres/err predicted correctly:  -185.2273
## % of predictable range [ (model-null)/(1-null) ]:  0.03624147
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          0.17966           0.18320           0.18118           0.02011
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.2     AIC: 604.3
## log likelihood:  -287.1097
## Nagelkerke R2:  0.0510309
## % pres/err predicted correctly:  -185.2352
## % of predictable range [ (model-null)/(1-null) ]:  0.0362005
## **************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.6049       0.2345
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.9     AIC: 604.4
## log likelihood:  -288.441
## Nagelkerke R2:  0.04415526
## % pres/err predicted correctly:  -186.9453
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.02735039
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     1.12599     -0.09047      0.26335
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 575   AIC: 604.5
## log likelihood:  -287.5124
## Nagelkerke R2:  0.0489543
## % pres/err predicted correctly:  -186.1927
## % of predictable range [ (model-null)/(1-null) ]:  0.03124523
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.34616     -0.09748      0.01867      0.13108
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.6     AIC: 606.1
## log likelihood:  -287.3214
## Nagelkerke R2:  0.04993965
## % pres/err predicted correctly:  -186.2143
## % of predictable range [ (model-null)/(1-null) ]:  0.03113345
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.71027      0.01102      0.15539
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.7     AIC: 606.3
## log likelihood:  -288.3729
## Nagelkerke R2:  0.0445081
## % pres/err predicted correctly:  -186.996
## % of predictable range [ (model-null)/(1-null) ]:  0.02708833
## ***************************
## model index:  18
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.35344     -0.12159      0.17540      0.01161
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.9     AIC: 606.4
## log likelihood:  -287.4665
## Nagelkerke R2:  0.04919102
## % pres/err predicted correctly:  -186.1987
## % of predictable range [ (model-null)/(1-null) ]:  0.03121422
## **************************
## 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)
##        0.6206982         0.0004696        -0.0667706         0.6955866         0.0108288
##      stimlen:pos
##       -0.0738150
##
## Degrees of Freedom: 594 Total (i.e. Null);  589 Residual
## Null Deviance:       593.7
## Residual Deviance: 574.3     AIC: 609.7
## log likelihood:  -287.1633
## Nagelkerke R2:  0.05075454
## % pres/err predicted correctly:  -186.1823
## % of predictable range [ (model-null)/(1-null) ]:  0.03129909
## **************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.347
##
## Degrees of Freedom: 594 Total (i.e. Null);  594 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.7     AIC: 619.2
## log likelihood:  -296.8519
## Nagelkerke R2:  -3.517158e-16
## % pres/err predicted correctly:  -192.2303
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  15
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.20314      0.02109
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.6     AIC: 621.1
## log likelihood:  -296.7931
## Nagelkerke R2:  0.0003134394
## % pres/err predicted correctly:  -192.1926
## % of predictable range [ (model-null)/(1-null) ]:  0.0001949289
## ***************************
```

```r
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) … [illegible]								
preserved ~ pos * log_freq	584.2[illegible]	[illegible]	0.324[illegible]		NA	NA	NA	NA	NA
preserved ~ pos + log_freq	584.3[illegible]	0.306[illegible]	0.287[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	584.9[illegible]	-0.0436647 NA	0.1016[illegible]		0.0307467	-0.0143905	NA	NA	NA
preserved ~ stimlen + pos * log_freq	585.5[illegible]	[illegible]	0.3099[illegible]		NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	585.59[illegible]	[illegible]	0.2725[illegible]	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAIC AICexp AICw NagR2 (Intercept) stimlen log_freq stimlen:log_freq pos log_freq:pos I(pos^2) I(pos^2):log_freq stimlen:pos stimlen:I(pos^2)								
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	586.5[illegible] -0.0214198	NA	0.115[illegible] NA	0.0271981	-0.0147444		NA	NA	NA
preserved ~ stimlen * log_freq + pos	587.0[illegible]		[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	587.1[illegible]	NA	0.127[illegible]	NA	0.0203[illegible]	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	587.5[illegible]		[illegible]	0.0430[illegible]	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	588.3[illegible]		[illegible]	NA	0.0230[illegible]	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	588.4[illegible] -0.0778059		0.0089[illegible]	0.177[illegible]	[illegible]	NA	-0.0152795	NA	NA
preserved ~ stimlen + log_freq	602.8[illegible]	NA	NA NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	604.3[illegible]	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	604.4[illegible] NA	NA	0.234[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	604.4[illegible] -0.0904708 NA	NA	0.263[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	606.0[illegible] -0.0974772 NA	NA	0.131[illegible]	NA	0.0186[illegible]	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	606.2[illegible] NA	NA	0.155[illegible]	NA	0.0110[illegible]	NA	NA	NA	NA
preserved ~ stimlen * pos	606.3[illegible] -0.1215904 NA	NA	0.175[illegible]	NA	NA	NA	NA	0.0116[illegible]	NA

Model	AIC DeltaAIC expAICw AICw NagR2 (Intercept) stimlen log_freq stimlen:log_freq pos log_freq:pos I(pos^2) I(pos^2):log_freq stimlen:pos stimlen:I(pos^2)									
preserved ~ stimlen * (I(pos^2) + pos)	609.7251.3446531080.03000056075425069820.4696	NA	0.6955866	NA	NA	-0.0667706	NA	NA	-0.0738150	0.0108288
preserved ~ 1	619.1374920.093600000000001034065NA36	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	621.0361.81700240000000003.12031037210938	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

```
## [1] "preserved ~ pos * log_freq"
```

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)          pos      log_freq  pos:log_freq
##      0.40653      0.32404       0.16660       0.04546
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 554.3     AIC: 584.3
```

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

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

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

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

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

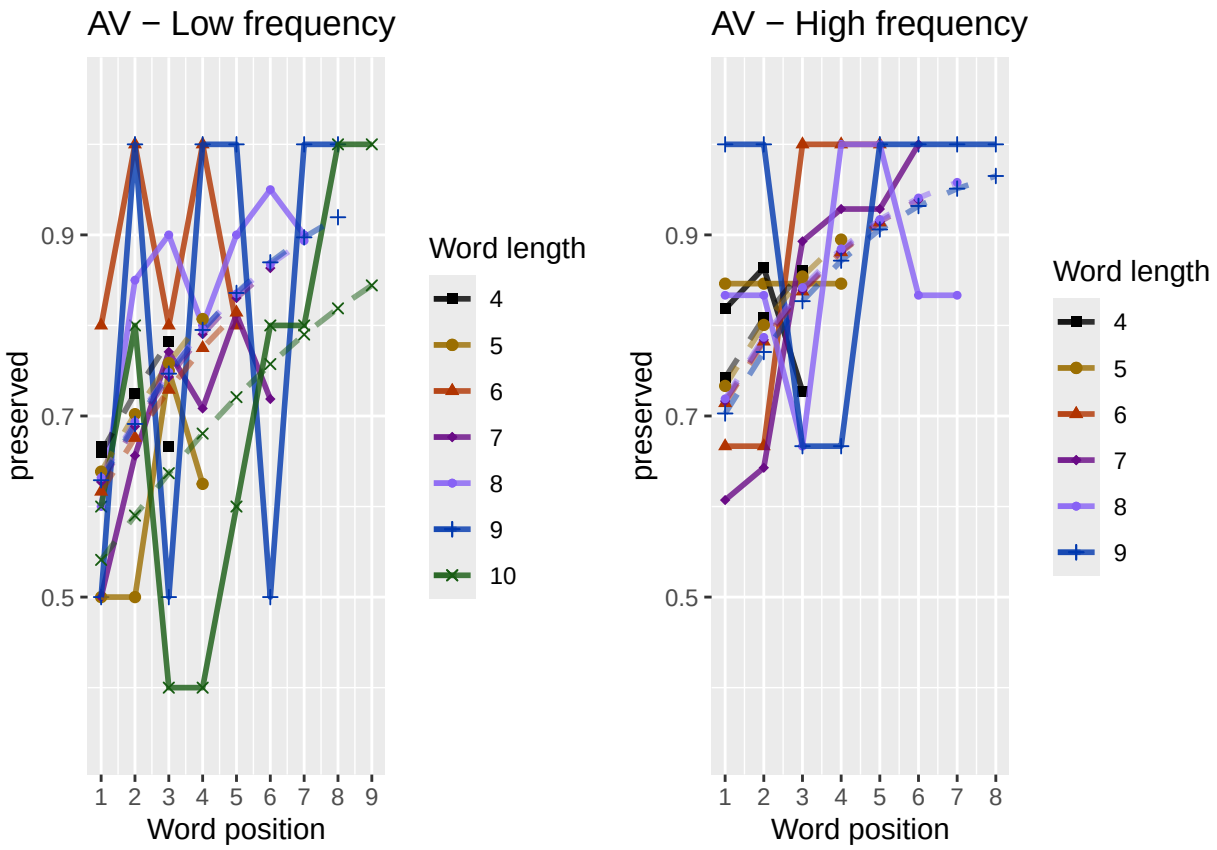
```r
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

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

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

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

```
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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3479       0.8354
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 496.7     AIC: 517.3
## log likelihood:  -248.352
## Nagelkerke R2:  0.2382794
## % pres/err predicted correctly:  -161.6703
## % of predictable range [ (model-null)/(1-null) ]:  0.1581529
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.7742      -0.5673
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 553   AIC: 574.6
## log likelihood:  -276.5075
## Nagelkerke R2:  0.1046995
## % pres/err predicted correctly:  -175.7914
## % of predictable range [ (model-null)/(1-null) ]:  0.08507366
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.6049       0.2345
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.9     AIC: 604.4
## log likelihood:  -288.441
## Nagelkerke R2:  0.04415526
## % pres/err predicted correctly:  -186.9453
## % of predictable range [ (model-null)/(1-null) ]:  0.02735039
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)     I(pos^2)          pos
##     0.71027      0.01102      0.15539
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.7     AIC: 606.3
## log likelihood:  -288.3729
## Nagelkerke R2:  0.0445081
## % pres/err predicted correctly:  -186.996
## % of predictable range [ (model-null)/(1-null) ]:  0.02708833
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.347
##
## Degrees of Freedom: 594 Total (i.e. Null);  594 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.7     AIC: 619.2
## log likelihood:  -296.8519
## Nagelkerke R2:  -3.517158e-16
## % pres/err predicted correctly:  -192.2303
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.20314      0.02109
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.6     AIC: 621.1
## log likelihood:  -296.7931
## Nagelkerke R2:  0.0003134394
## % pres/err predicted correctly:  -192.1926
## % of predictable range [ (model-null)/(1-null) ]:  0.0001949289
## ***************************
```

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

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

```
MEAICSummary$NagR2<-MERes$NagR2

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

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(posˆ2)	pos	stimlen
preserved ~ CumPres	517.2921	0.00000	1	1	0.2382790	0.3479118	0.8353789	NA	NA	NA	NA
preserved ~ CumErr	574.6266	57.33453	0	0	0.1046995	1.7742262	NA	-0.5673213	NA	NA	NA
preserved ~ pos	604.4258	87.13376	0	0	0.0441559	0.6048553	NA	NA	NA	0.2344932	NA
preserved ~ (I(posˆ2) + pos)	606.2778	88.98570	0	0	0.0445080	0.7102739	NA	NA	0.0110162	0.1553933	NA
preserved ~ 1	619.1749	101.88282	0	0	0.0000000	0.3465036	NA	NA	NA	NA	NA
preserved ~ stimlen	621.0718	103.77973	0	0	0.0003134	1.2031372	NA	NA	NA	NA	0.0210938

```
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	0.8000000	85
O	0.7686480	286
P	0.8000000	10
S	0.6000000	25
V	0.8536155	189

```
#  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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3834       0.9432
##
## Degrees of Freedom: 559 Total (i.e. Null);  558 Residual
## Null Deviance:       545
## Residual Deviance: 449.4     AIC: 470.5
## log likelihood:  -224.7226
## Nagelkerke R2:  0.2520416
## % pres/err predicted correctly:  -145.9395
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.1658079
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.8185      -0.5923
##
## Degrees of Freedom: 559 Total (i.e. Null);  558 Residual
## Null Deviance:       545
## Residual Deviance: 508.1     AIC: 530.3
## log likelihood:  -254.0515
## Nagelkerke R2:  0.1023651
## % pres/err predicted correctly:  -160.5534
## % of predictable range [ (model-null)/(1-null) ]:  0.082843
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.5932       0.2540
##
## Degrees of Freedom: 559 Total (i.e. Null);  558 Residual
## Null Deviance:       545
## Residual Deviance: 527.3     AIC: 555.2
## log likelihood:  -263.6335
## Nagelkerke R2:  0.04996717
## % pres/err predicted correctly:  -169.7035
## % of predictable range [ (model-null)/(1-null) ]:  0.03089702
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    0.616088     0.002393     0.236859
##
## Degrees of Freedom: 559 Total (i.e. Null);  557 Residual
## Null Deviance:       545
## Residual Deviance: 527.3     AIC: 557.2
## log likelihood:  -263.6305
## Nagelkerke R2:  0.04998385
## % pres/err predicted correctly:  -169.7218
## % of predictable range [ (model-null)/(1-null) ]:  0.03079277
## ***************************
## model index:  6
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.399
##
## Degrees of Freedom: 559 Total (i.e. Null);  559 Residual
## Null Deviance:       545
## Residual Deviance: 545   AIC: 570.9
## log likelihood:  -272.4753
## Nagelkerke R2:  -3.569274e-16
## % pres/err predicted correctly:  -175.1458
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.19225      0.03049
##
## Degrees of Freedom: 559 Total (i.e. Null);  558 Residual
## Null Deviance:       545
## Residual Deviance: 544.7     AIC: 572.7
## log likelihood:  -272.3639
## Nagelkerke R2:  0.0006394384
## % pres/err predicted correctly:  -175.0768
## % of predictable range [ (model-null)/(1-null) ]:  0.0003919131
## ***************************
```

```
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(posˆ2)	pos	stimlen
preserved ~ CumPres	470.5147	0.00000	1	1	0.2520416	0.3833967	0.9431806	NA	NA	NA	NA
preserved ~ CumErr	530.2600	59.74527	0	0	0.1023651	1.8184861	NA	-0.5923452	NA	NA	NA
preserved ~ pos	555.2343	84.71957	0	0	0.0499672	0.5931555	NA	NA	NA	0.2540330	NA
preserved ~ (I(posˆ2) + pos)	557.2263	86.71154	0	0	0.0499830	0.6160879	NA	NA	0.0023932	0.2368589	NA
preserved ~ 1	570.8622	100.34744	0	0	0.0000000	0.3993664	NA	NA	NA	NA	NA
preserved ~ stimlen	572.6615	102.14674	0	0	0.0006394	1.1922549	NA	NA	NA	NA	0.0304891

```
#  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:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.4431       1.1326
##
## Degrees of Freedom: 474 Total (i.e. Null);  473 Residual
## Null Deviance:       459.9
## Residual Deviance: 380.3     AIC: 402.2
## log likelihood:  -190.1382
## Nagelkerke R2:  0.2487754
## % pres/err predicted correctly:  -124.2064
## % of predictable range [ (model-null)/(1-null) ]:  0.1593955
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.7298      -0.5745
##
## Degrees of Freedom: 474 Total (i.e. Null);  473 Residual
## Null Deviance:       459.9
## Residual Deviance: 439.5     AIC: 462.4
## log likelihood:  -219.7657
## Nagelkerke R2:  0.06761092
## % pres/err predicted correctly:  -139.6434
## % of predictable range [ (model-null)/(1-null) ]:  0.05575524
```

```
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.6240       0.2513
##
## Degrees of Freedom: 474 Total (i.e. Null);  473 Residual
## Null Deviance:       459.9
## Residual Deviance: 444.1     AIC: 472.3
## log likelihood:  -222.0675
## Nagelkerke R2:  0.0525665
## % pres/err predicted correctly:  -143.0911
## % of predictable range [ (model-null)/(1-null) ]:  0.03260877
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    0.545105    -0.008557     0.312801
##
## Degrees of Freedom: 474 Total (i.e. Null);  472 Residual
## Null Deviance:       459.9
## Residual Deviance: 444.1     AIC: 474.3
## log likelihood:  -222.0328
## Nagelkerke R2:  0.05279435
## % pres/err predicted correctly:  -143.0124
## % of predictable range [ (model-null)/(1-null) ]:  0.03313686
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.402
##
## Degrees of Freedom: 474 Total (i.e. Null);  474 Residual
## Null Deviance:       459.9
## Residual Deviance: 459.9     AIC: 485.8
## log likelihood:  -229.9398
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -147.9481
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.09861      0.04485
##
## Degrees of Freedom: 474 Total (i.e. Null);  473 Residual
## Null Deviance:       459.9
## Residual Deviance: 459.5     AIC: 487.4
## log likelihood:  -229.7306
## Nagelkerke R2:  0.001419442
## % pres/err predicted correctly:  -147.8224
## % of predictable range [ (model-null)/(1-null) ]:  0.0008435548
## ***************************
```

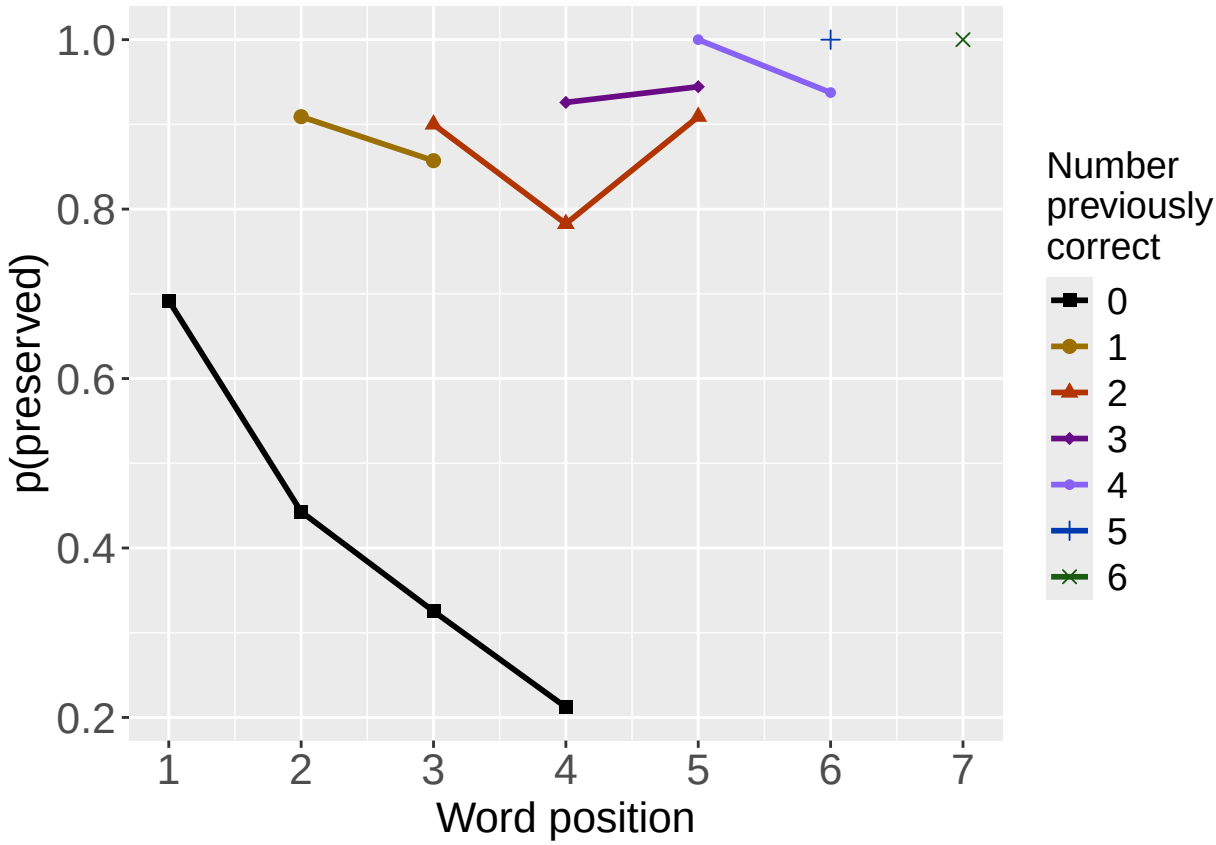
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumPres	402.1693	0.00000	1	1	0.2487754	0.4430621	1.132649	NA	NA	NA	NA
preserved ~ CumErr	462.3758	60.20652	0	0	0.0676109	1.7297542	NA	-0.5745478	NA	NA	NA
preserved ~ pos	472.2873	70.11800	0	0	0.0525665	1.6240230	NA	NA	NA	0.2513208	NA
preserved ~ (I(pos^2) + pos)	474.2528	72.08346	0	0	0.0527948	1.5451055	NA	NA	-0.0085568	0.3128005	NA
preserved ~ 1	485.8106	83.64133	0	0	0.0000000	1.4017166	NA	NA	NA	NA	NA
preserved ~ stimlen	487.4378	85.26849	0	0	0.0014194	1.0986130	NA	NA	NA	NA	0.0448497

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

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

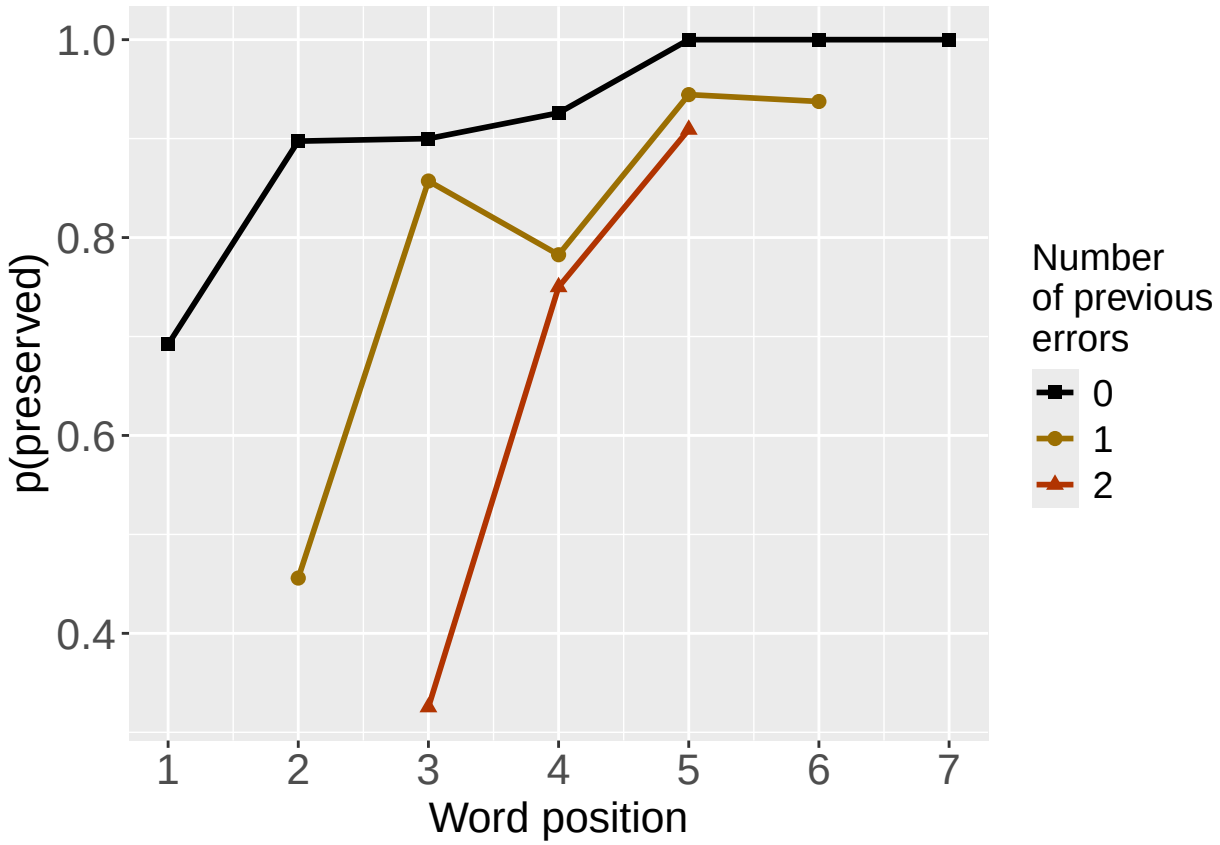
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

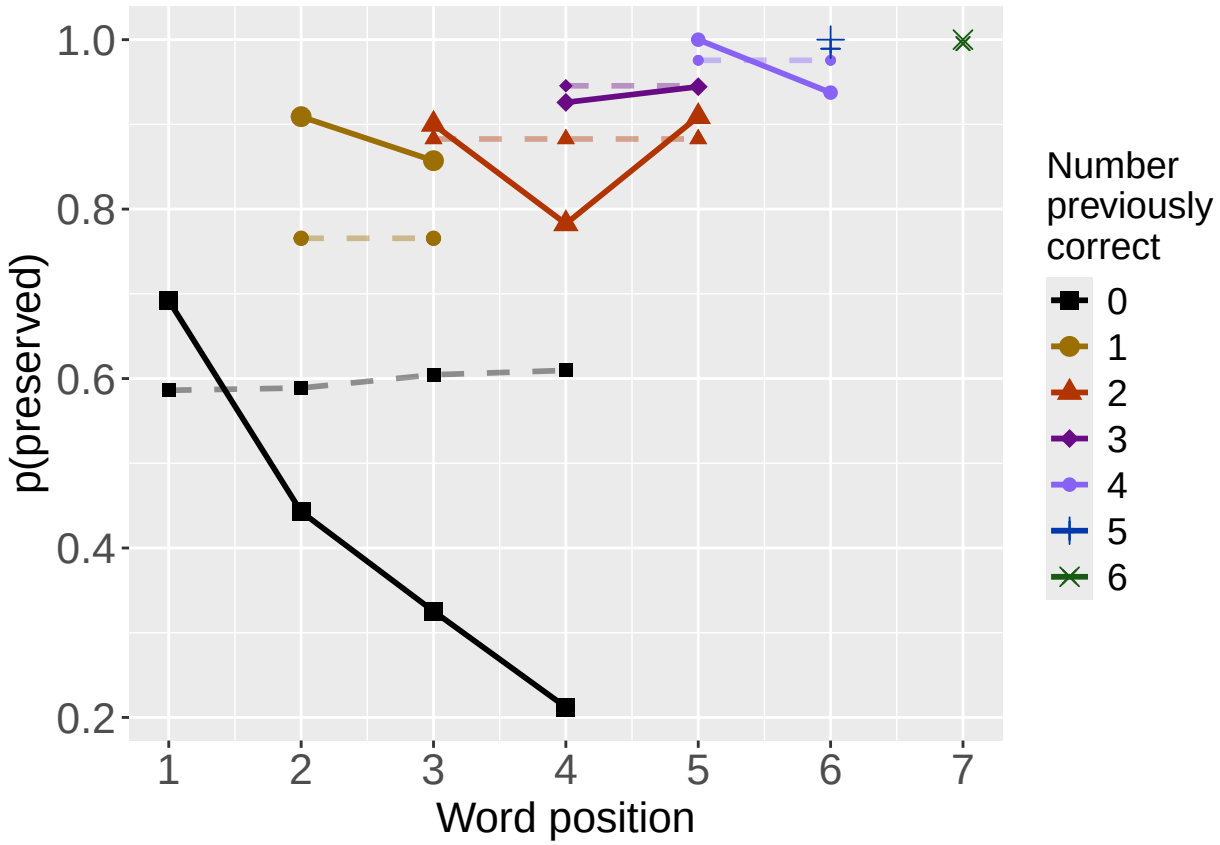
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

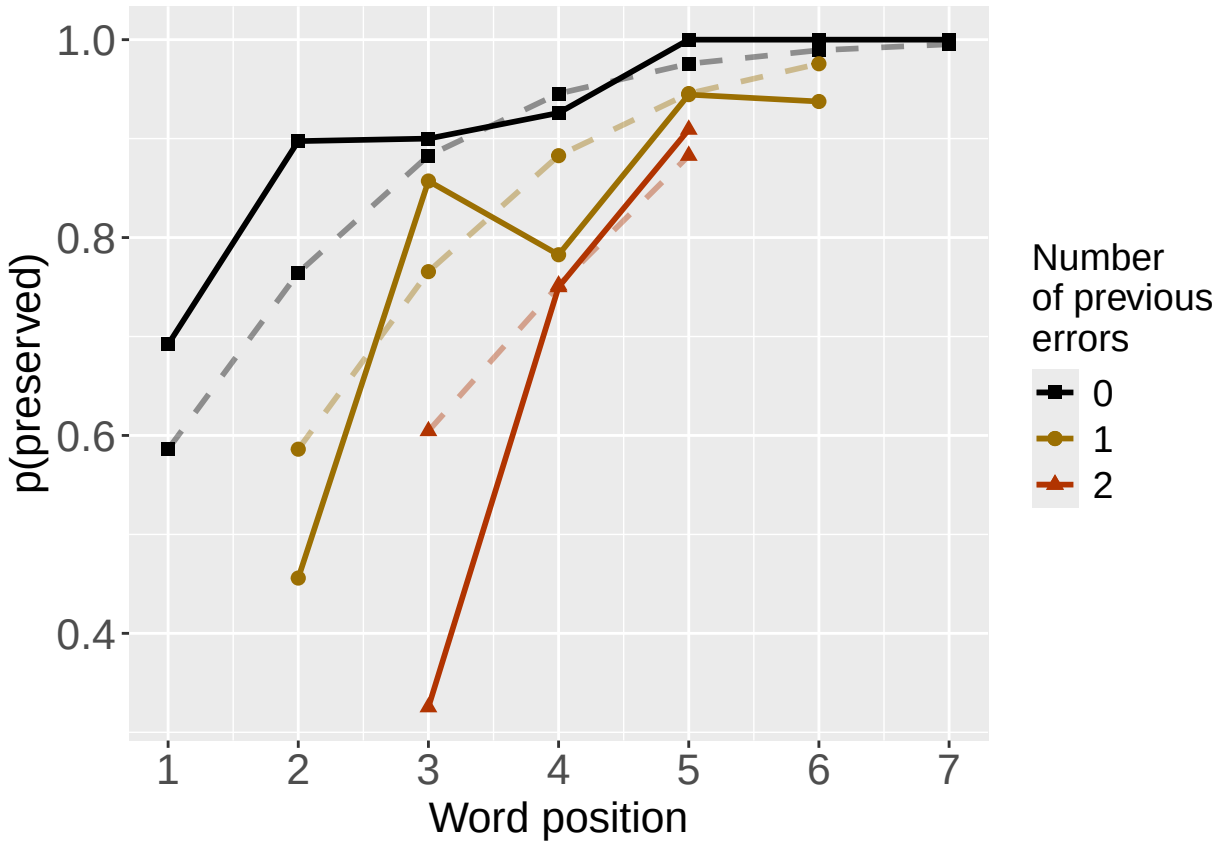
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos
##     1.90417      1.40509      0.07306     -1.03943
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 462.9     AIC: 481.8
## log likelihood:  -231.4263
## Nagelkerke R2:  0.3127038
## % pres/err predicted correctly:  -147.0517
## % of predictable range [ (model-null)/(1-null) ]:  0.2338067
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3479       0.8354
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 496.7     AIC: 517.3
## log likelihood:  -248.352
## Nagelkerke R2:  0.2382794
## % pres/err predicted correctly:  -161.6703
## % of predictable range [ (model-null)/(1-null) ]:  0.1581529
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.71027      0.01102      0.15539
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.7     AIC: 606.3
## log likelihood:  -288.3729
## Nagelkerke R2:  0.0445081
## % pres/err predicted correctly:  -186.996
## % of predictable range [ (model-null)/(1-null) ]:  0.02708833
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos
preserved ~ CumPres + I(pos^2) + pos	481.7578	0.00000	1	1	0.3127038	1.9041714	1.4050937	0.0730597	-1.0394293

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(posˆ2)	pos
preserved ~ CumPres	517.2921	35.53429	0	0	0.2382794	0.3479118	0.8353789	NA	NA
preserved ~ I(posˆ2) + pos	606.2778	124.51998	0	0	0.0445081	0.7102739	NA	0.0110162	0.1553933

```
########
# 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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres      stimlen
##      1.2471       0.8633      -0.1394
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 492.4     AIC: 514.5
## log likelihood:  -246.1861
## Nagelkerke R2:  0.2480408
## % pres/err predicted correctly:  -160.1868
## % of predictable range [ (model-null)/(1-null) ]:  0.1658304
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3479       0.8354
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 496.7     AIC: 517.3
## log likelihood:  -248.352
## Nagelkerke R2:  0.2382794
## % pres/err predicted correctly:  -161.6703
## % of predictable range [ (model-null)/(1-null) ]:  0.1581529
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     1.20314      0.02109
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.6     AIC: 621.1
## log likelihood:  -296.7931
## Nagelkerke R2:  0.0003134394
## % pres/err predicted correctly:  -192.1926
## % of predictable range [ (model-null)/(1-null) ]:  0.0001949289
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	stimlen
preserved ~ CumPres + stimlen	514.5104	0.000000	1.0000000	0.8007244	0.2480408	1.2471209	0.8633349	-0.1393863
preserved ~ CumPres	517.2921	2.781657	0.2488691	0.1992756	0.2382794	0.3479118	0.8353789	NA
preserved ~ stimlen	621.0718	106.561382	0.0000000	0.0000000	0.0003134	1.2031372	NA	0.0210938

```
########
# 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)
}
```

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##       1.282        1.344       -0.524
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 466.6     AIC: 483.9
## log likelihood:  -233.3159
## Nagelkerke R2:  0.3046033
## % pres/err predicted correctly:  -148.389
## % of predictable range [ (model-null)/(1-null) ]:  0.2268859
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3479       0.8354
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 496.7     AIC: 517.3
## log likelihood:  -248.352
## Nagelkerke R2:  0.2382794
## % pres/err predicted correctly:  -161.6703
## % of predictable range [ (model-null)/(1-null) ]:  0.1581529
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.6049       0.2345
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 576.9     AIC: 604.4
## log likelihood:  -288.441
## Nagelkerke R2:  0.04415526
## % pres/err predicted correctly:  -186.9453
## % of predictable range [ (model-null)/(1-null) ]:  0.02735039
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	pos
preserved ~ CumPres + pos	483.9028	0.00000	1e+00	0.9999999	0.3046033	1.2816428	1.3435708	-0.5240003
preserved ~ CumPres	517.2921	33.38922	1e-07	0.0000001	0.2382794	0.3479118	0.8353789	NA
preserved ~ pos	604.4258	120.52299	0e+00	0.0000000	0.0441553	0.6048553	NA	0.2344932

```
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)	CumPres	I(pos^2)	pos	stimlen
preserved ~ CumPres + I(pos^2) + pos	481.7578	0.000000	1.0000000	0.0000000	0.3127033	8.9041714	1.4050937	0.0730597	-1.0394293	NA
preserved ~ CumPres + pos	483.9028	0.000000	1.0000000	0.9999990	0.3046033	1.2816428	1.3435708	NA	-0.5240003	NA
preserved ~ CumPres + stimlen	514.5104	0.000000	1.0000000	0.8007240	0.2480403	8.2471209	0.8633349	NA	NA	-0.1393863
preserved ~ CumPres	517.2921	35.534286	0.0000000	0.0000000	0.2382794	0.3479118	0.8353789	NA	NA	NA
preserved ~ CumPres	517.2921	2.781657	0.2488690	0.1992750	0.2382794	0.3479118	0.8353789	NA	NA	NA
preserved ~ CumPres	517.2921	33.389224	0.0000000	0.0000001	0.2382794	0.3479118	0.8353789	NA	NA	NA
preserved ~ pos	604.4258	120.522988	0.0000000	0.0000000	0.0441553	0.6048553	NA	NA	0.2344932	NA
preserved ~ I(pos^2) + pos	606.2778	124.519983	0.0000000	0.0000000	0.0445080	0.7102739	NA	0.0110162	0.1553933	NA
preserved ~ stimlen	621.0718	106.561382	0.0000000	0.0000000	0.0003134	1.2031372	NA	NA	NA	0.0210938

```
# 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)      CumPres     I(pos^2)          pos     log_freq
##     1.84449      1.35769      0.07987     -1.01729      0.22351
##
## Degrees of Freedom: 594 Total (i.e. Null);  590 Residual
## Null Deviance:       593.7
## Residual Deviance: 453.9     AIC: 474.4
## log likelihood:  -226.945
## Nagelkerke R2:  0.3317095
## % pres/err predicted correctly:  -144.8775
## % of predictable range [ (model-null)/(1-null) ]:  0.2450588
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos      stimlen     log_freq
##     1.61793      1.35535      0.07778     -1.00904      0.03437      0.23949
##
## Degrees of Freedom: 594 Total (i.e. Null);  589 Residual
## Null Deviance:       593.7
## Residual Deviance: 453.7     AIC: 476.2
## log likelihood:  -226.8582
## Nagelkerke R2:  0.3320749
## % pres/err predicted correctly:  -144.77
## % of predictable range [ (model-null)/(1-null) ]:  0.245615
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos
##     1.90417      1.40509      0.07306     -1.03943
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 462.9     AIC: 481.8
```

```
## log likelihood:  -231.4263
## Nagelkerke R2:  0.3127038
## % pres/err predicted correctly:  -147.0517
## % of predictable range [ (model-null)/(1-null) ]:  0.2338067
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres     I(pos^2)          pos      stimlen
##     2.37784      1.40340      0.07843     -1.05333     -0.07329
##
## Degrees of Freedom: 594 Total (i.e. Null);  590 Residual
## Null Deviance:       593.7
## Residual Deviance: 461.9     AIC: 482.7
## log likelihood:  -230.9311
## Nagelkerke R2:  0.3148181
## % pres/err predicted correctly:  -146.9192
## % of predictable range [ (model-null)/(1-null) ]:  0.2344927
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.347
##
## Degrees of Freedom: 594 Total (i.e. Null);  594 Residual
## Null Deviance:       593.7
## Residual Deviance: 593.7     AIC: 619.2
## log likelihood:  -296.8519
## Nagelkerke R2:  -3.517158e-16
## % pres/err predicted correctly:  -192.2303
## % 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))
```

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumPres + I(pos^2) + pos + log_freq	474.3647	0.000000	1.0000000	0.6920596	0.3317095	1.844493	1.357685	0.0798712	-1.017286	0.2235069	NA
preserved ~ CumPres + I(pos^2) + pos + stimlen + log_freq	476.1728	1.808127	0.4049209	0.2802294	0.3320749	1.617927	1.355349	0.0777844	-1.009045	0.2394852	0.0343700
preserved ~ CumPres + I(pos^2) + pos	481.7577	7.393060	0.0248095	0.0171696	0.3127038	1.904171	1.405094	0.0730597	-1.039429	NA	NA
preserved ~ CumPres + I(pos^2) + pos + stimlen	482.7338	8.368742	0.0152308	0.0105401	0.3148181	2.377841	1.403400	0.0784306	-1.053329	NA	-0.0732903
preserved ~ 1	619.1749	144.810163	0.0000000	0.0000000	0.0000000	1.346504	NA	NA	NA	NA	NA

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

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

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

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

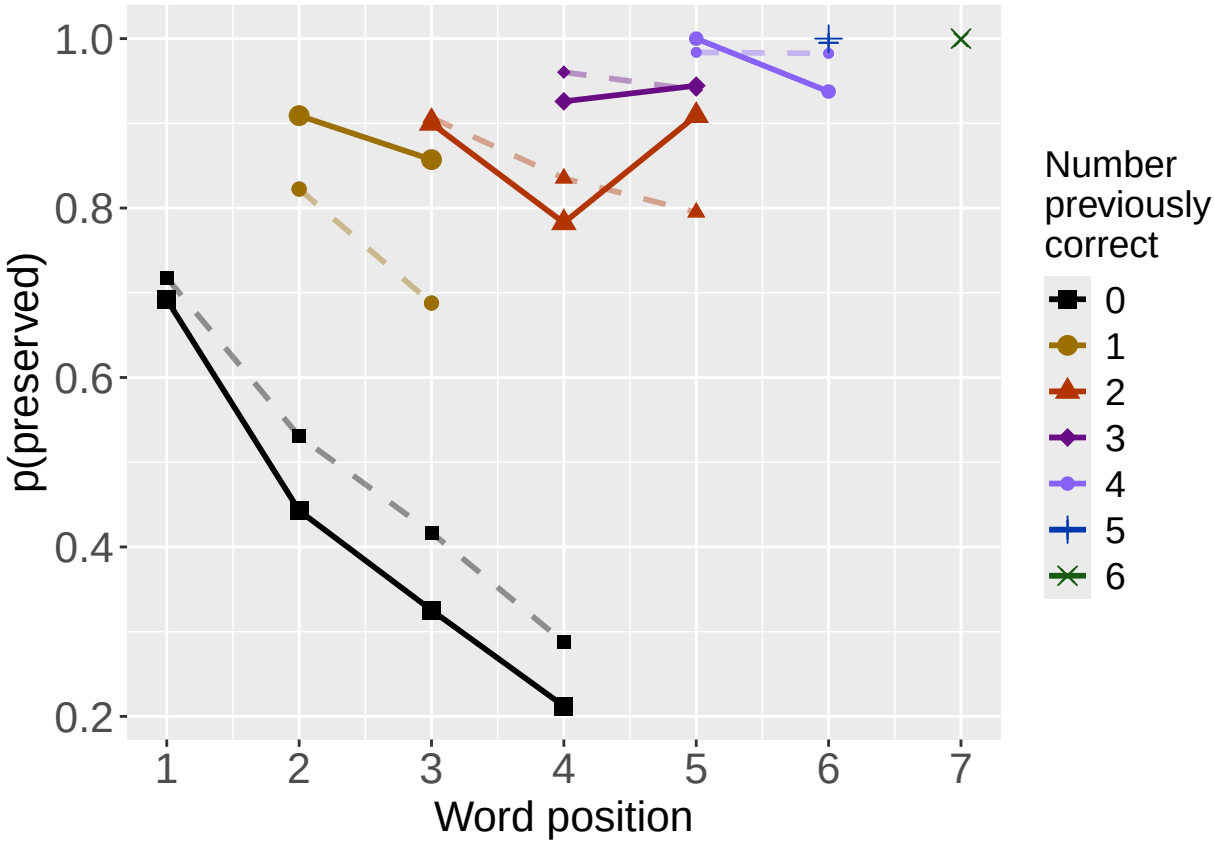
```
## Single term deletions
##
## Model:
## preserved ~ CumPres + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## CumPres   1   555.41 573.89
## pos       1   467.20 485.67
## log_freq  1   462.85 481.33
## I(pos^2)  1   458.49 476.96
## <none>        453.89 474.36
```

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

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

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

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

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

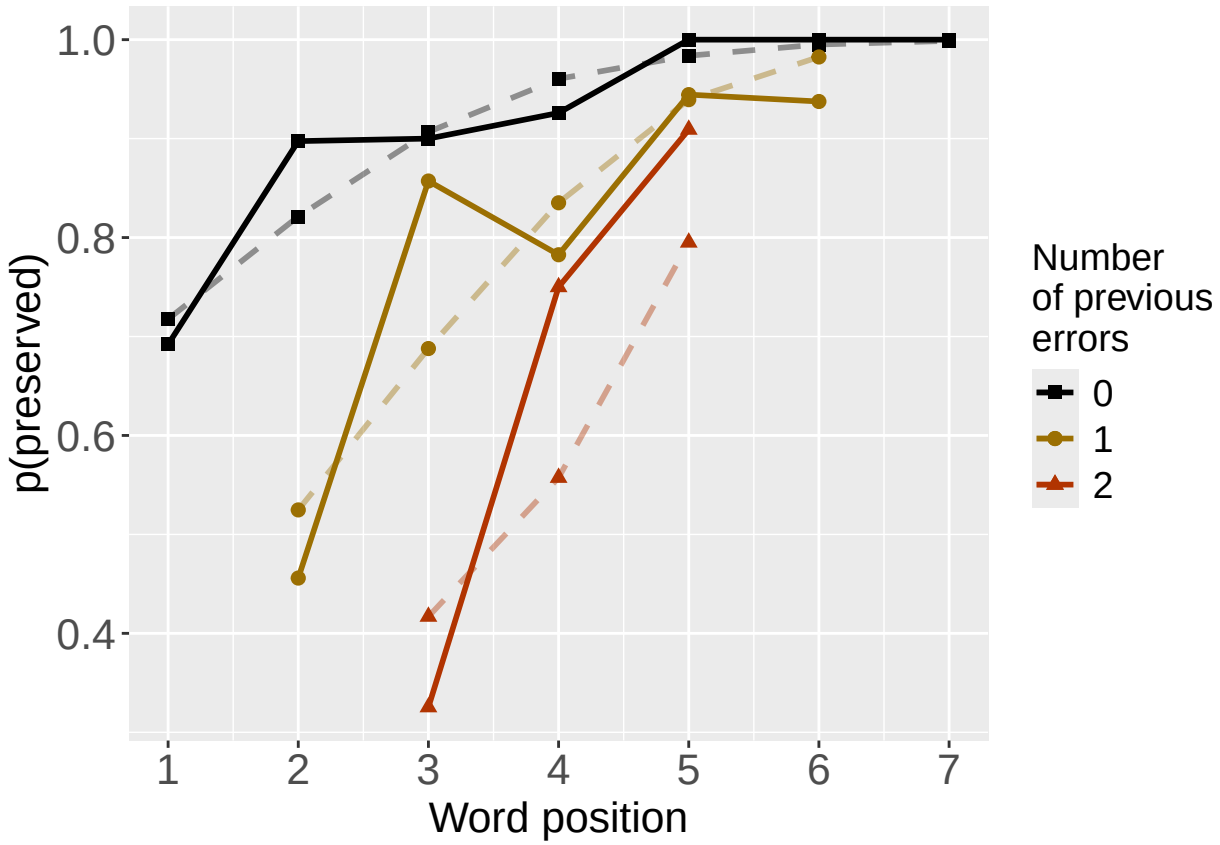
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

  RndModelAIC<-numeric(length=RandomSamples)
  for(rindex in seq(1,RandomSamples)){
  # Shuffle cumulative values
    PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
    PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
    BestModelRnd <- glm(as.formula(BestModelFormulaL3Rnd),
                        family="binomial",data=PosDat)
    RndModelAIC[rindex] <- BestModelRnd$aic
  }
  ModelNames<-c(paste0("***",BestModelFormulaL3),
```

```
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random average"),
                                     AIC=c(mean(RndModelAIC))))
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random SD"),
                                     AIC=c(sd(RndModelAIC))))
  write.csv(BestModelRndDF,paste0(TablesDir,CurPat,"_",CurTask,
                                  "_best_model_with_random_cum_term.csv"),
            row.names = FALSE)
}
```

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

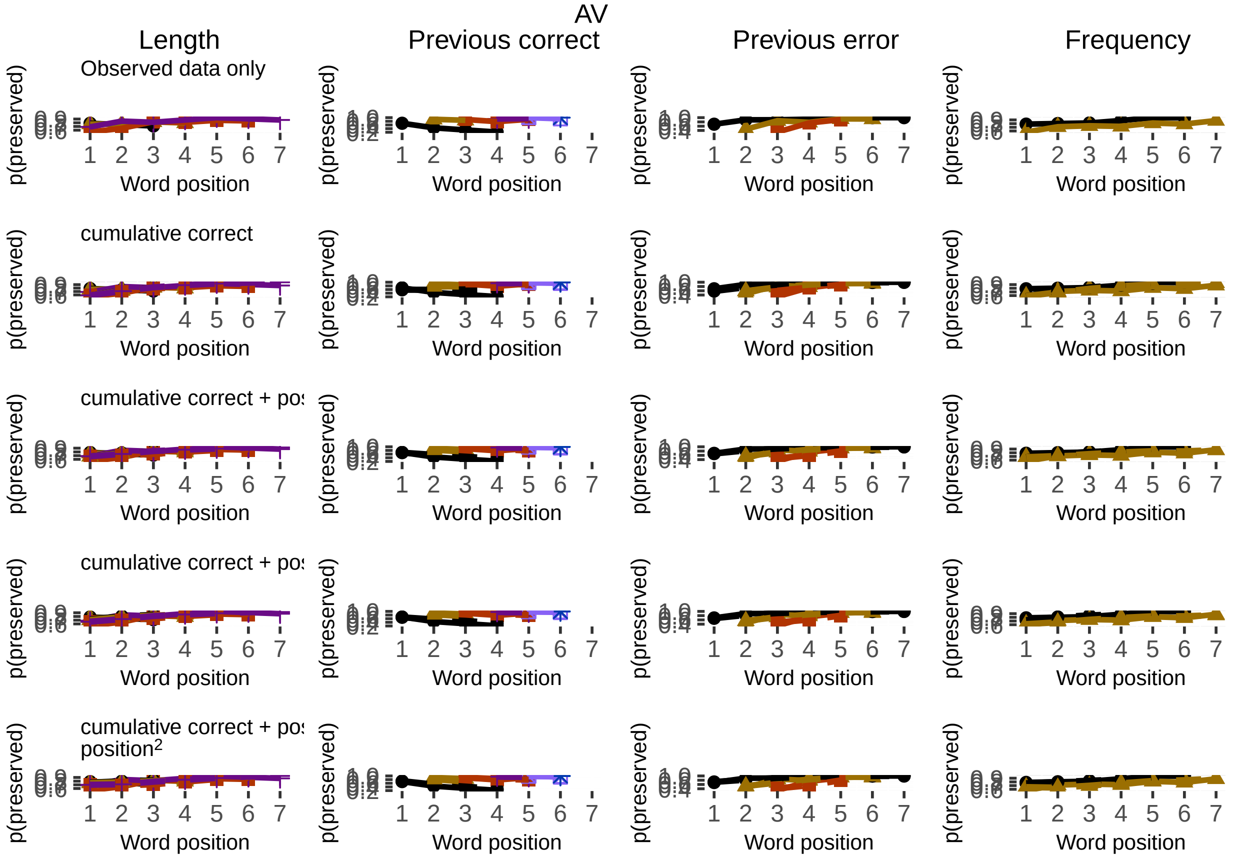
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.3479       0.8354
##
## Degrees of Freedom: 594 Total (i.e. Null);  593 Residual
## Null Deviance:       593.7
## Residual Deviance: 496.7     AIC: 517.3
## log likelihood:  -248.352
```

```
## Nagelkerke R2:  0.2382794
## % pres/err predicted correctly:  -161.6703
## % of predictable range [ (model-null)/(1-null) ]:  0.1581529
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos
##       1.282        1.344       -0.524
##
## Degrees of Freedom: 594 Total (i.e. Null);  592 Residual
## Null Deviance:       593.7
## Residual Deviance: 466.6     AIC: 483.9
## log likelihood:  -233.3159
## Nagelkerke R2:  0.3046033
## % pres/err predicted correctly:  -148.389
## % of predictable range [ (model-null)/(1-null) ]:  0.2268859
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos     log_freq
##      1.1611       1.2929      -0.4538       0.2114
##
## Degrees of Freedom: 594 Total (i.e. Null);  591 Residual
## Null Deviance:       593.7
## Residual Deviance: 458.5     AIC: 477.4
## log likelihood:  -229.2446
## Nagelkerke R2:  0.3219924
## % pres/err predicted correctly:  -146.236
## % of predictable range [ (model-null)/(1-null) ]:  0.2380284
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres          pos     log_freq     I(pos^2)
##     1.84449      1.35769     -1.01729      0.22351      0.07987
##
## Degrees of Freedom: 594 Total (i.e. Null);  590 Residual
## Null Deviance:       593.7
## Residual Deviance: 453.9     AIC: 474.4
## log likelihood:  -226.945
## Nagelkerke R2:  0.3317095
## % pres/err predicted correctly:  -144.8775
## % of predictable range [ (model-null)/(1-null) ]:  0.2450588
```

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

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

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

	CumPres	I(pos^2)	pos	log_freq
McFadden	0.1792822	0.0179274	0.0266834	0.0237033
SquaredCorrelation	0.1635594	0.0167597	0.0239591	0.0222172
Nagelkerke	0.2533532	0.0259608	0.0371127	0.0344144
Estrella	0.1850212	0.0185150	0.0275117	0.0244833

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

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

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

```
##                                                                 model deviance
## CumPres + pos + log_freq + I(pos^2) CumPres + pos + log_freq + I(pos^2) 453.8901
## CumPres + pos + log_freq                       CumPres + pos + log_freq 458.4892
## CumPres + pos                                        CumPres + pos 466.6318
## CumPres                                                    CumPres 496.7040
## null                                                          null 593.7039
##                                     deviance_explained percent_explained
## CumPres + pos + log_freq + I(pos^2)          139.81380          23.54942
## CumPres + pos + log_freq                     135.21470          22.77477
## CumPres + pos                                127.07212          21.40328
## CumPres                                       96.99993          16.33810
## null                                           0.00000           0.00000
##                                     percent_of_explained_deviance increment_in_explained
## CumPres + pos + log_freq + I(pos^2)                     100.00000              3.289447
## CumPres + pos + log_freq                                 96.71055              5.823873
## CumPres + pos                                            90.88668             21.508745
## CumPres                                                  69.37793             69.377934
## null                                                           NA              0.000000
```

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

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

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

	deviance	deviance_explained
CumPres + pos + log_freq + I(pos^2)	453.8901	139.81380
CumPres + pos + log_freq	458.4892	135.21470
CumPres + pos	466.6318	127.07212
CumPres	496.7040	96.99993
null	593.7039	0.00000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumPres + pos + log_freq + I(pos^2)	23.54942	100.00000	3.289447
CumPres + pos + log_freq	22.77477	96.71055	5.823873
CumPres + pos	21.40328	90.88668	21.508745
CumPres	16.33810	69.37793	69.377934
null	0.00000	NA	0.000000

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

```
##          Nagelkerke
## CumPres  0.72213088
## I(pos^2) 0.07399592
## pos      0.10578199
## log_freq 0.09809121
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumPres	0.6814316	496.7040
preserved ~ CumPres+pos+log_freq+I(pos^2)	0.8390450	453.8901
preserved ~ CumPres+pos+log_freq	0.8663860	458.4892
preserved ~ CumPres+pos	0.8678862	466.6318

```
## 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_CumPres
## 1                        preserved ~ CumPres       0.6814316       496.7040    0.0000000
## 2 preserved ~ CumPres+pos+log_freq+I(pos^2)       0.8390450       453.8901    0.1576135
## 3          preserved ~ CumPres+pos+log_freq       0.8663860       458.4892    0.1849545
## 4                    preserved ~ CumPres+pos       0.8678862       466.6318    0.1864546
##   diff_CumPres+pos+log_freq+I(pos^2) diff_CumPres+pos+log_freq diff_CumPres+pos
## 1                        -0.15761346              -0.184954469     -0.186454592
## 2                         0.00000000              -0.027341009     -0.028841132
## 3                         0.02734101               0.000000000     -0.001500123
## 4                         0.02884113               0.001500123      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")
```

model	diff_CumPres	diff_CumPres+pos+log_freq+I(pos^2)	diff_CumPres+pos+log_freq
preserved ~ CumPres	0.0000000	-0.1576135	-0.1849545
preserved ~ CumPres+pos+log_freq+I(pos^2)	0.1576135	0.0000000	-0.0273410
preserved ~ CumPres+pos+log_freq	0.1849545	0.0273410	0.0000000
preserved ~ CumPres+pos	0.1864546	0.0288411	0.0015001