DS - 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	173	13	20	NA	NA	206
2	21	NA	145	12	28	206
3	81	NA	47	75	3	206
4	107	NA	51	14	10	182
5	54	NA	48	11	10	123
6	42	NA	14	16	5	77
7	27	NA	11	3	2	43
8	11	NA	1	3	NA	15
9	4	NA	1	NA	2	7

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8398058	0.0631068	0.0970874	NA	NA	206
2	0.1019417	NA	0.7038835	0.0582524	0.1359223	206
3	0.3932039	NA	0.2281553	0.3640777	0.0145631	206
4	0.5879121	NA	0.2802198	0.0769231	0.0549451	182
5	0.4390244	NA	0.3902439	0.0894309	0.0813008	123
6	0.5454545	NA	0.1818182	0.2077922	0.0649351	77

pos_factor	O	P	V	1	S	total
7	0.6279070	NA	0.2558140	0.0697674	0.0465116	43
8	0.7333333	NA	0.0666667	0.2000000	NA	15
9	0.5714286	NA	0.1428571	NA	0.2857143	7

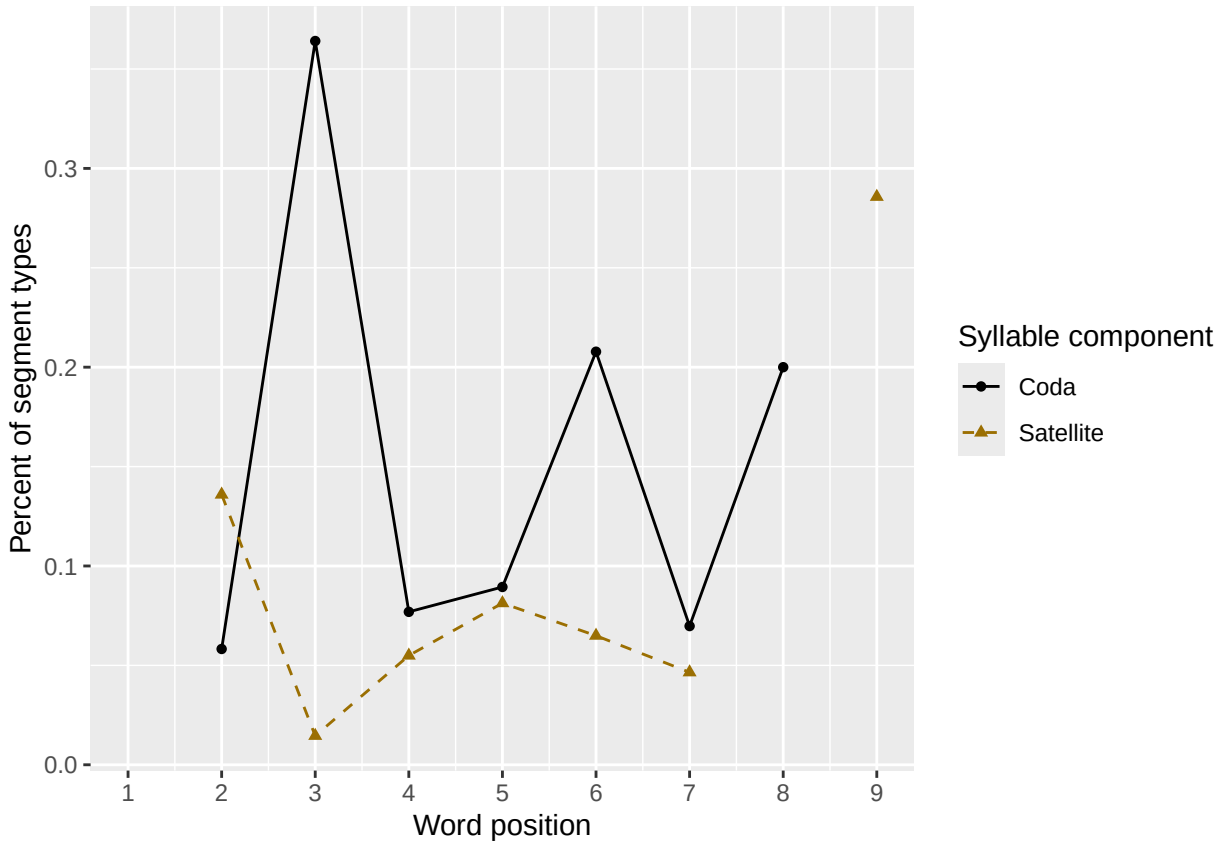
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
syll_comp_plot
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range (`geom_line()`).
## Removed 4 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 0.917 0.938 0.938 NA     NA     NA     NA     NA     NA
## 2       5 0.952 0.879 0.831  0.760 NA     NA     NA     NA     NA
## 3       6 0.924 0.859 0.826  0.793  0.75  NA     NA     NA     NA
## 4       7 0.966 0.922 0.824  0.75   0.730  0.618 NA     NA     NA
## 5       8 1     1     0.982  0.946  0.821  0.875  0.875 NA     NA
## 6       9 1     0.917 0.875  0.917  0.75   0.875  0.875  0.917 NA
## 7      10 0.929 0.857 0.714  0.786  0.786  0.5    0.714  0.571  0.571
```

```
# 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.
```

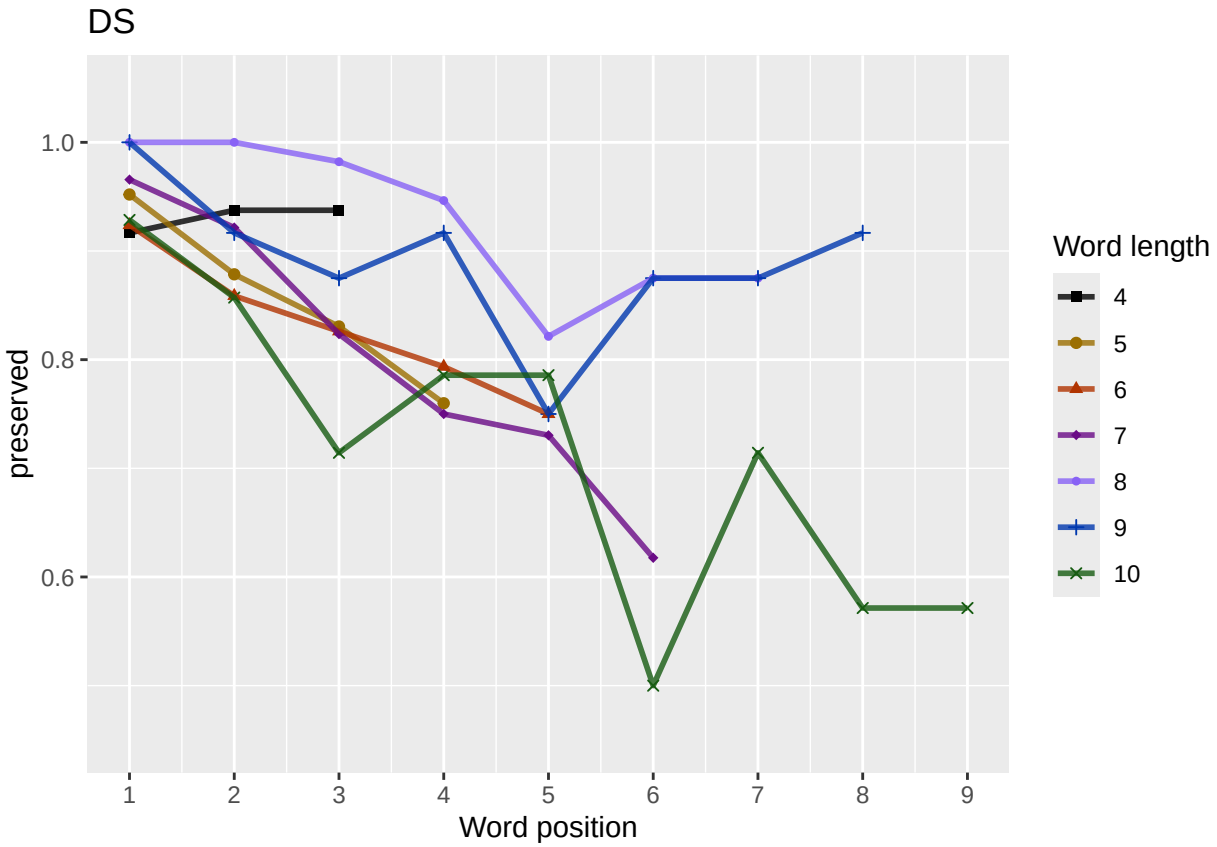
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    24    24    24    NA    NA    NA    NA    NA    NA
## 2       5    59    59    59    59    NA    NA    NA    NA    NA
## 3       6    46    46    46    46    46    NA    NA    NA    NA
## 4       7    34    34    34    34    34    34    NA    NA    NA
## 5       8    28    28    28    28    28    28    28    NA    NA
## 6       9     8     8     8     8     8     8     8     8    NA
## 7      10     7     7     7     7     7     7     7     7     7
```

```
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:  6
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.58949      0.05802     -0.76748
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.1     AIC: 906
## log likelihood:  -402.5375
## Nagelkerke R2:  0.07256528
## % pres/err predicted correctly:  -252.6093
## % of predictable range [ (model-null)/(1-null) ]:  0.03975939
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.93289      0.10689      0.04928     -0.74234
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 802.6     AIC: 906.7
## log likelihood:  -401.3096
## Nagelkerke R2:  0.07659236
## % pres/err predicted correctly:  -252.1566
## % of predictable range [ (model-null)/(1-null) ]:  0.04147366
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          3.78705           0.03910           0.19557          -1.56154          -0.01516
##      stimlen:pos
##          0.07909
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1059 Residual
## Null Deviance:       848.4
## Residual Deviance: 800.5     AIC: 909.1
## log likelihood:  -400.2566
## Nagelkerke R2:  0.08003851
## % pres/err predicted correctly:  -251.7353
## % of predictable range [ (model-null)/(1-null) ]:  0.04306881
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.0072       0.1472      -0.3390
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 807.5     AIC: 910.2
## log likelihood:  -403.7726
## Nagelkerke R2:  0.06850488
## % pres/err predicted correctly:  -253.0697
## % of predictable range [ (model-null)/(1-null) ]:  0.03801637
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.83719      0.03052     -0.57226      0.03102
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 806.4     AIC: 910.6
## log likelihood:  -403.1928
## Nagelkerke R2:  0.07041221
## % pres/err predicted correctly:  -252.7791
## % of predictable range [ (model-null)/(1-null) ]:  0.03911647
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.7343      -0.2709
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 812.4     AIC: 911.4
## log likelihood:  -406.2226
## Nagelkerke R2:  0.06042324
## % pres/err predicted correctly:  -253.9446
## % of predictable range [ (model-null)/(1-null) ]:  0.03470372
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)
##        1.75
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1064 Residual
## Null Deviance:       848.4
## Residual Deviance: 848.4     AIC: 945.3
## log likelihood:  -424.1912
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -263.1102
## % 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
##     2.13258     -0.05735
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 847.3     AIC: 945.6
## log likelihood:  -423.6268
## Nagelkerke R2:  0.001929048
## % pres/err predicted correctly:  -262.7577
## % of predictable range [ (model-null)/(1-null) ]:  0.001334723
## ***************************
```

```
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 ~ I(pos^2) + pos	905.9574	0.0000000	1.0000000	0.4595534	0.0072563	3.589494	NA	-0.7674847	NA	0.0580151	NA
preserved ~ stimlen + I(pos^2) + pos	906.7374	0.7800787	0.6770302	0.3111310	0.0076592	4.932892	-0.1068872	-0.7423399	NA	0.0492795	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)	909.0583	1.1007488	0.3212168	0.2509750	0.2808003	3.5787054	0.0390994	-1.5615425	0.0790870	0.1955663	-0.015155
preserved ~ stimlen + pos	910.1882	2.2308578	0.1205810	0.0554187	0.2706850	4.2900719	0.1471688	-0.3389910	NA	NA	NA
preserved ~ stimlen * pos	910.5614	9.6045009	0.1000388	0.0459707	0.2707041	2.2837186	0.0305209	-0.5722637	0.0310249	NA	NA
preserved ~ pos	911.3872	2.4297954	0.0662107	0.0304278	0.2706042	3.2734260	NA	-0.2708506	NA	NA	NA
preserved ~ 1	945.2782	39.3208458	0.0000000	0.0000000	0.0000000	0.0750027	NA	NA	NA	NA	NA
preserved ~ stimlen	945.5708	39.6126250	0.0000000	0.0000000	0.0019229	0.0132575	-0.0573469	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.58949      0.05802     -0.76748
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.1     AIC: 906
```

```
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
```

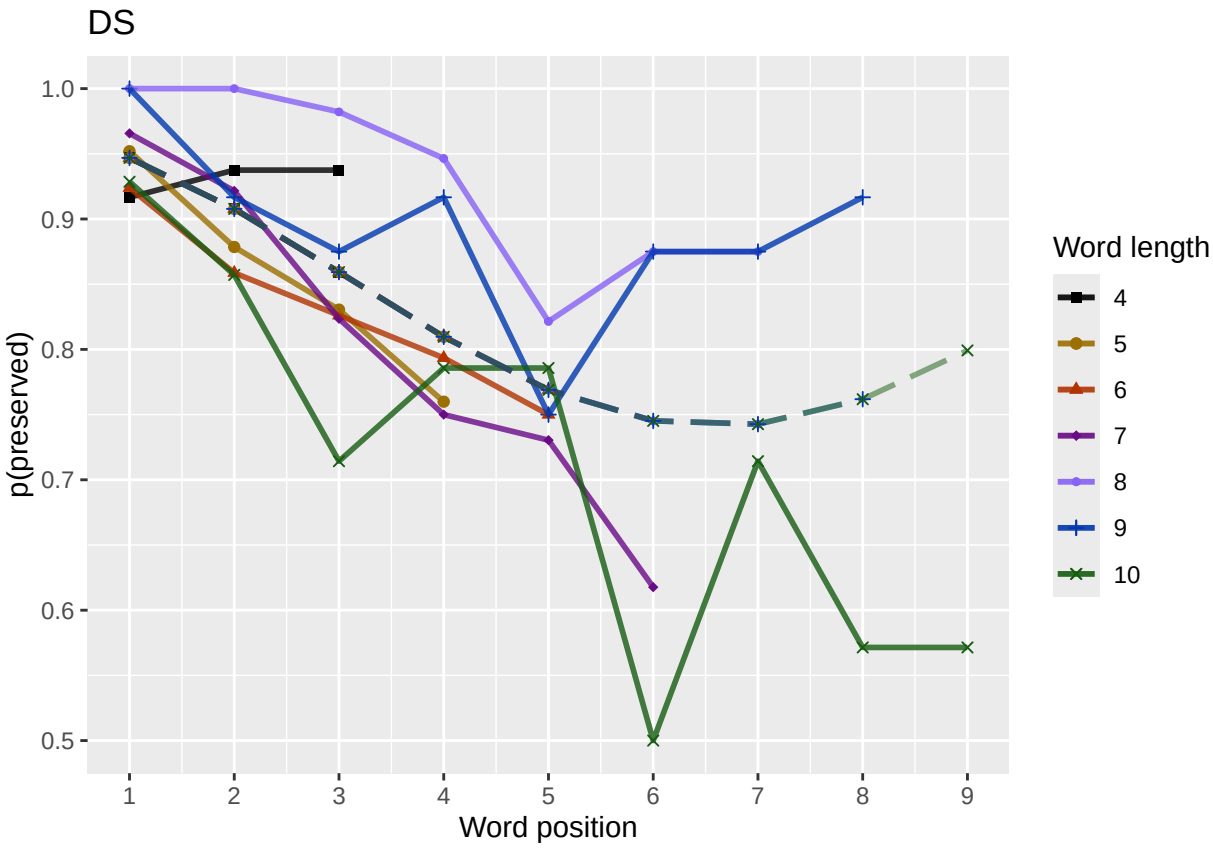
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.947 0.908 0.859 NA     NA     NA     NA     NA     NA
## 2       5 0.947 0.908 0.859  0.810 NA     NA     NA     NA     NA
## 3       6 0.947 0.908 0.859  0.810  0.769 NA     NA     NA     NA
## 4       7 0.947 0.908 0.859  0.810  0.769  0.745 NA     NA     NA
## 5       8 0.947 0.908 0.859  0.810  0.769  0.745  0.743 NA     NA
```

```
## 6       9 0.947 0.908 0.859  0.810  0.769  0.745  0.743  0.762 NA
## 7      10 0.947 0.908 0.859  0.810  0.769  0.745  0.743  0.762  0.799
```

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

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

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

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



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

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

# number responses
resp_num<-0
prev_pos<-9999 # big number to initialize (so first position is smaller)
resp_num_array <- integer(length = nrow(PosDat))
for(i in seq(1,nrow(PosDat))){
  if(PosDat$pos[i] <= prev_pos){ # equal for resp length 1
    resp_num <- resp_num + 1
  }
  resp_num_array[i]<-resp_num
  prev_pos <- PosDat$pos[i]
}
PosDat$resp_num <- resp_num_array
```

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       30   206
```

```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 30 / 206 = 14.56 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:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.65146      0.08474     -0.77575
##
## Degrees of Freedom: 999 Total (i.e. Null);  997 Residual
## Null Deviance:       592.8
## Residual Deviance: 581.3     AIC: 676.7
## log likelihood:  -290.64
## Nagelkerke R2:  0.025559
## % pres/err predicted correctly:  -167.22
## % of predictable range [ (model-null)/(1-null) ]:  0.01090482
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.58545      0.01042      0.08383     -0.77276
##
## Degrees of Freedom: 999 Total (i.e. Null);  996 Residual
## Null Deviance:       592.8
## Residual Deviance: 581.3     AIC: 678.8
## log likelihood:  -290.631
## Nagelkerke R2:  0.02559871
## % pres/err predicted correctly:  -167.2326
## % of predictable range [ (model-null)/(1-null) ]:  0.01083079
## ***************************
## 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)
##          0.57101           0.48360          -0.10550           0.90001           0.02845
##      stimlen:pos
##         -0.25597
##
## Degrees of Freedom: 999 Total (i.e. Null);  994 Residual
## Null Deviance:       592.8
## Residual Deviance: 578.7     AIC: 680.9
## log likelihood:  -289.3532
## Nagelkerke R2:  0.03124049
## % pres/err predicted correctly:  -166.7895
## % of predictable range [ (model-null)/(1-null) ]:  0.01343624
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##       2.636       -0.115
##
## Degrees of Freedom: 999 Total (i.e. Null);  998 Residual
## Null Deviance:       592.8
## Residual Deviance: 588.9     AIC: 681.9
## log likelihood:  -294.4265
## Nagelkerke R2:  0.008754983
## % pres/err predicted correctly:  -168.4631
## % of predictable range [ (model-null)/(1-null) ]:  0.003596161
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        2.25
##
## Degrees of Freedom: 999 Total (i.e. Null);  999 Residual
## Null Deviance:       592.8
## Residual Deviance: 592.8     AIC: 683.9
## log likelihood:  -296.388
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -169.0747
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.3341       0.0573      -0.1380
##
## Degrees of Freedom: 999 Total (i.e. Null);  997 Residual
## Null Deviance:       592.8
## Residual Deviance: 588.3     AIC: 684
## log likelihood:  -294.1509
## Nagelkerke R2:  0.00998264
## % pres/err predicted correctly:  -168.419
## % of predictable range [ (model-null)/(1-null) ]:  0.003855118
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.26030     -0.07266     -0.43510      0.03981
```

```
##
## Degrees of Freedom: 999 Total (i.e. Null);  996 Residual
## Null Deviance:       592.8
## Residual Deviance: 587.2     AIC: 684.9
## log likelihood:  -293.6149
## Nagelkerke R2:  0.01236748
## % pres/err predicted correctly:  -168.1911
## % of predictable range [ (model-null)/(1-null) ]:  0.005195458
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.34270     -0.01398
##
## Degrees of Freedom: 999 Total (i.e. Null);  998 Residual
## Null Deviance:       592.8
## Residual Deviance: 592.7     AIC: 685.7
## log likelihood:  -296.3668
## Nagelkerke R2:  9.508465e-05
## % pres/err predicted correctly:  -169.0573
## % of predictable range [ (model-null)/(1-null) ]:  0.0001021542
## ***************************
```

```
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 ~ I(pos^2) + pos	676.6741	0.000000	1.000000	0.6152054	0.0255593	0.6514609	NA	-0.7757532	NA	0.0847428	NA
preserved ~ stimlen + I(pos^2) + pos	678.7652	2.090939	0.3515267	0.2162601	0.0255987	2.3585445	-0.0104198	-0.7727628	NA	0.0838328	NA
preserved ~ stimlen * (I(pos^2) + pos)	680.9064	4.232074	0.1204676	0.0741123	0.0312405	2.5571010	-0.4835989	9.9000120	-0.2559748	-0.1055050	0.0284511

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ pos	681.9035	4.229257	0.0731950	0.0450299	0.0087552	0.6355395	NA	-0.1149706	NA	NA	NA
preserved ~ 1	683.9175	6.243372	0.0267376	0.0164491	0.0000000	0.2501870	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	683.9730	6.298859	0.0260060	0.0159990	0.0099826	0.3341415	0.0572957	-0.1380001	NA	NA	NA
preserved ~ stimlen * pos	684.8598	8.185709	0.0166915	0.0102688	0.0123675	0.2602974	-0.0726593	-0.4350958	0.0398147	NA	NA
preserved ~ stimlen	685.7214	4.047332	0.0108492	0.0066745	0.0000951	0.3426959	-0.0139842	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.
```

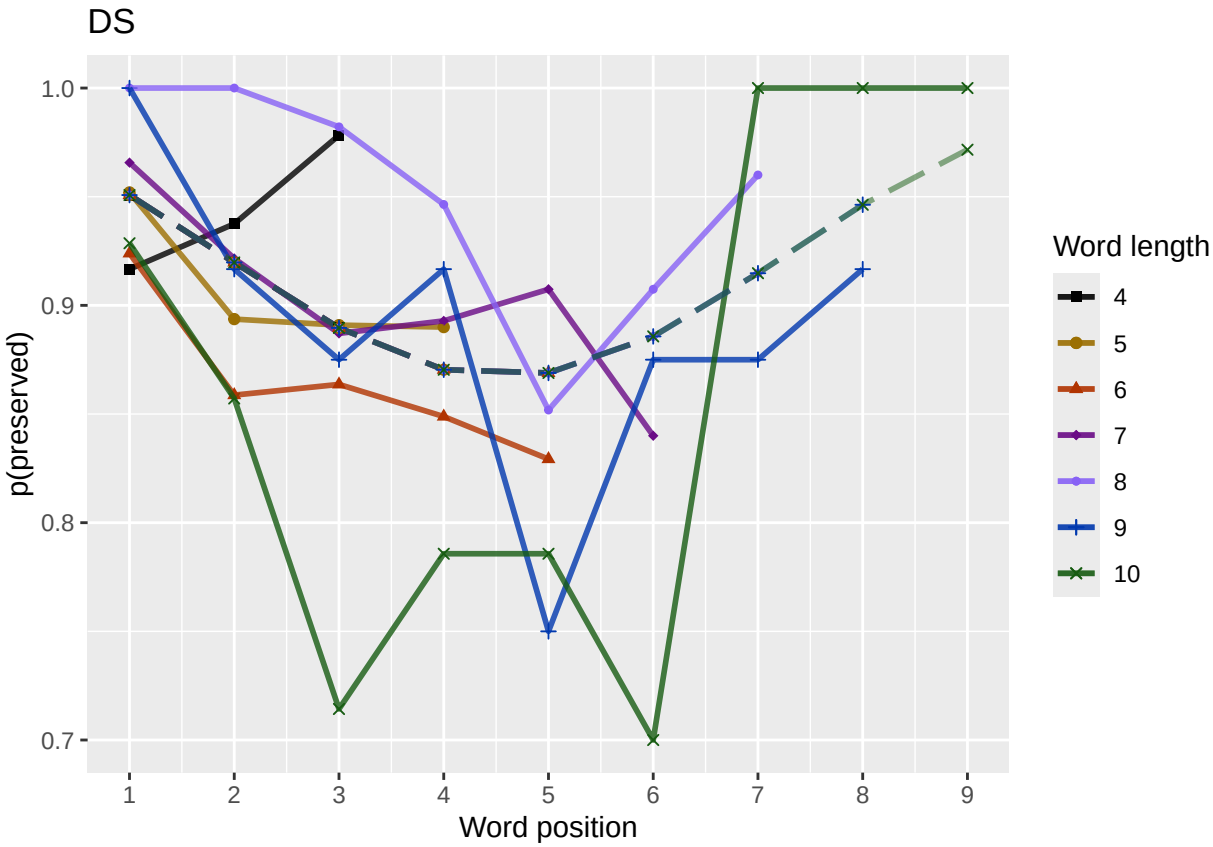
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.951 0.920 0.890 NA     NA     NA     NA     NA     NA
## 2       5 0.951 0.920 0.890  0.870 NA     NA     NA     NA     NA
## 3       6 0.951 0.920 0.890  0.870  0.869 NA     NA     NA     NA
## 4       7 0.951 0.920 0.890  0.870  0.869  0.886 NA     NA     NA
## 5       8 0.951 0.920 0.890  0.870  0.869  0.886  0.915 NA     NA
## 6       9 0.951 0.920 0.890  0.870  0.869  0.886  0.915  0.946 NA
## 7      10 0.951 0.920 0.890  0.870  0.869  0.886  0.915  0.946  0.972
```

```
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: 0.45 - 1.05"
```

```
# 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] -0.03403166
```

```
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] "Average upward change after U minimum"
## [1] 0.02827416
```

```
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] "differences from left max to min for each row: "
## [1] 0.08758786 0.13719097 0.17789013 0.20165854 0.20418998 0.20418998 0.20418998
## [1] "differences from min to right max for each row: "
## [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.01914154 0.05654832
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.20419
## [1] 0.05654832
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.2769397
```

```
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",
```

```r
            "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:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     2.58285      0.16643      0.04945     -0.74731      0.19346
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 791.6     AIC: 897.6
## log likelihood:  -395.8204
## Nagelkerke R2:  0.09448201
```

```
## % pres/err predicted correctly:  -249.0201
## % of predictable range [ (model-null)/(1-null) ]:  0.05334921
## **************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##        2.5823729         0.1666019         0.1905821         0.0494699        -0.7475229
## stimlen:log_freq
##        0.0004237
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1059 Residual
## Null Deviance:       848.4
## Residual Deviance: 791.6     AIC: 899.6
## log likelihood:  -395.8204
## Nagelkerke R2:  0.09448226
## % pres/err predicted correctly:  -249.0198
## % of predictable range [ (model-null)/(1-null) ]:  0.05335034
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##      1.6552       0.2069      -0.3430       0.1937
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 796.5     AIC: 901
## log likelihood:  -398.265
## Nagelkerke R2:  0.08653775
## % pres/err predicted correctly:  -250.0131
## % of predictable range [ (model-null)/(1-null) ]:  0.04958958
## **************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          2.563927           0.167595           0.047917          -0.737686           0.183813
## I(pos^2):log_freq       pos:log_freq
##         -0.001662           0.010472
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1058 Residual
## Null Deviance:       848.4
## Residual Deviance: 791.6     AIC: 901.5
## log likelihood:  -395.8057
```

```
## Nagelkerke R2:  0.09452983
## % pres/err predicted correctly:  -249.0151
## % of predictable range [ (model-null)/(1-null) ]:  0.05336815
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      1.67076       0.20883      -0.35157       0.25560      -0.01551
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 796.2     AIC: 902.6
## log likelihood:  -398.1237
## Nagelkerke R2:  0.08699773
## % pres/err predicted correctly:  -250.051
## % of predictable range [ (model-null)/(1-null) ]:  0.04944581
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         1.667231          0.204429          0.232468         -0.342873         -0.005746
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 796.5     AIC: 903
## log likelihood:  -398.2506
## Nagelkerke R2:  0.0865847
## % pres/err predicted correctly:  -250.0192
## % of predictable range [ (model-null)/(1-null) ]:  0.04956638
## **************************
## 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
##         3.6029873          0.0622256         -0.7833278          0.1372495         -0.0005076
##      pos:log_freq
##         0.0076233
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1059 Residual
## Null Deviance:       848.4
## Residual Deviance: 797.2     AIC: 903.4
## log likelihood:  -398.5941
## Nagelkerke R2:  0.08546542
```

```
## % pres/err predicted correctly:  -250.2438
## % of predictable range [ (model-null)/(1-null) ]:  0.04871583
## **************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          2.550435           0.170423           0.152358           0.047582          -0.736790
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##          0.005260          -0.002077           0.011437
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1057 Residual
## Null Deviance:       848.4
## Residual Deviance: 791.6     AIC: 903.5
## log likelihood:  -395.7978
## Nagelkerke R2:  0.09455543
## % pres/err predicted correctly:  -249.0162
## % of predictable range [ (model-null)/(1-null) ]:  0.05336388
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         1.659490          0.212306          0.223678         -0.353703          0.006629
##     log_freq:pos
##        -0.018740
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1059 Residual
## Null Deviance:       848.4
## Residual Deviance: 796.2     AIC: 904.6
## log likelihood:  -398.1106
## Nagelkerke R2:  0.0870405
## % pres/err predicted correctly:  -250.0526
## % of predictable range [ (model-null)/(1-null) ]:  0.04943975
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      2.6953      -0.2538       0.1472
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.5     AIC: 905.9
## log likelihood:  -402.7744
```

```
## Nagelkerke R2:  0.07178726
## % pres/err predicted correctly:  -251.9295
## % of predictable range [ (model-null)/(1-null) ]:  0.04233355
## **************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.58949      0.05802     -0.76748
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.1     AIC: 906
## log likelihood:  -402.5375
## Nagelkerke R2:  0.07256528
## % pres/err predicted correctly:  -252.6093
## % of predictable range [ (model-null)/(1-null) ]:  0.03975939
## **************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.93289      0.10689      0.04928     -0.74234
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 802.6     AIC: 906.7
## log likelihood:  -401.3096
## Nagelkerke R2:  0.07659236
## % pres/err predicted correctly:  -252.1566
## % of predictable range [ (model-null)/(1-null) ]:  0.04147366
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.71131      -0.25882       0.18773      -0.01031
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.4     AIC: 907.6
## log likelihood:  -402.7086
## Nagelkerke R2:  0.07200338
## % pres/err predicted correctly:  -251.9828
## % of predictable range [ (model-null)/(1-null) ]:  0.04213174
```

```
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          3.78705           0.03910           0.19557          -1.56154          -0.01516
##      stimlen:pos
##          0.07909
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1059 Residual
## Null Deviance:       848.4
## Residual Deviance: 800.5     AIC: 909.1
## log likelihood:  -400.2566
## Nagelkerke R2:  0.08003851
## % pres/err predicted correctly:  -251.7353
## % of predictable range [ (model-null)/(1-null) ]:  0.04306881
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.0072       0.1472      -0.3390
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 807.5     AIC: 910.2
## log likelihood:  -403.7726
## Nagelkerke R2:  0.06850488
## % pres/err predicted correctly:  -253.0697
## % of predictable range [ (model-null)/(1-null) ]:  0.03801637
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.83719      0.03052     -0.57226      0.03102
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 806.4     AIC: 910.6
## log likelihood:  -403.1928
## Nagelkerke R2:  0.07041221
## % pres/err predicted correctly:  -252.7791
## % of predictable range [ (model-null)/(1-null) ]:  0.03911647
## ***************************
```

```
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.7343      -0.2709
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 812.4     AIC: 911.4
## log likelihood:  -406.2226
## Nagelkerke R2:  0.06042324
## % pres/err predicted correctly:  -253.9446
## % of predictable range [ (model-null)/(1-null) ]:  0.03470372
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##    1.793944    -0.002038     0.185389
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 836.7     AIC: 936.8
## log likelihood:  -418.3478
## Nagelkerke R2:  0.01987385
## % pres/err predicted correctly:  -260.0111
## % of predictable range [ (model-null)/(1-null) ]:  0.01173399
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         1.810165         -0.005243          0.235831         -0.007415
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 836.6     AIC: 938.7
## log likelihood:  -418.3224
## Nagelkerke R2:  0.01995983
## % pres/err predicted correctly:  -260.0156
## % of predictable range [ (model-null)/(1-null) ]:  0.0117168
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.75
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1064 Residual
## Null Deviance:       848.4
## Residual Deviance: 848.4     AIC: 945.3
## log likelihood:  -424.1912
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -263.1102
## % 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
##     2.13258     -0.05735
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 847.3     AIC: 945.6
## log likelihood:  -423.6268
## Nagelkerke R2:  0.001929048
## % pres/err predicted correctly:  -262.7577
## % of predictable range [ (model-null)/(1-null) ]:  0.001334723
## ***************************
```

```
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.names
kable(FLPAICSummary)
```

Model	AIC DeltaAICAICexpAICwNagR2(Intercept)stimlenlog_freqstimlen:log_freq	pos	log_freq:pos		I(pos^2)	log_freq:I(pos^2)			stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos + log_freq	897.59.28001000004928401224853828418664031934514	-0.7473140	NA	NA	0.0494459	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	899.59.40107367055129104592485323731666019058204237	-0.7475229	NA	NA	0.0494699	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	901.03.4180241810258921066153575227069259236510	-0.3430394	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	901.53.4954687384365820724325983921767595583810	-0.7376858	0.0104720	NA	0.0479171	-0.0016617	NA	NA	NA
preserved ~ stimlen + pos * log_freq	902.59.3200365820704044785199677076208826559094	-0.3510715	-0.0155109	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	902.95.36301268456323738615867231204022924675 -0.0057463	-0.3428729	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	903.42.67339054090726608534650298A 0.1372495	-0.7833278	0.0076233	NA	0.0622256	-0.0005076	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	903.54.0947073511092510939435540357041233235782600	-0.7367895	NA	0.0114807475821			-0.0020772	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	904.57.4981410300479350287040594901203623607066292	-0.3537032	NA	-0.0187404	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	905.86.2472622159801787641787953A 0.1472266	-0.2538383	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	905.98.346450815026367522735658940A NA NA	-0.7674847	NA	NA	0.0580151 NA		NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	906.73.71446461033505093763928920688412 NA	-0.7423399	NA	NA	0.0492795 NA		NA	NA	NA

Model	AIC DeltaAICexpAICwAICNagR2(Intercept)stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	stimlen:pos	logfreq:pos	log_freq:I(pos^2)	stimlen:I(pos^2)
preserved ~ pos * log_freq	907.6285035686610832622907341301A	0.1877330	NA	- 0.2588191	- 0.0203083	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	909.05814653163238515968033857054390994	NA		- 1.5615425	NA	NA	0.1955663	NA	NA	0.0790876	- 0.015155
preserved ~ stimlen + pos	910.18829564251840590768300971994716588	NA		- 0.3389910	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	910.561969006915269075270418271863052009	NA		- 0.5722637	NA	NA	NA	NA	NA	0.0310249	
preserved ~ pos	911.387279436310106049861042732426NA	NA	NA	- 0.2708506	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	936.829923710510000000019873389944 - 0.0020377	0.1853886		NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	938.741381509980000000001959180165 - 0.0052431	0.2358311	- 0.0074150	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	945.247838654130000000000001750027A	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	945.547097710930000000001924932575 - 0.0573469	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     2.58285      0.16643      0.04945     -0.74731      0.19346
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 791.6     AIC: 897.6
```

```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.
```

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

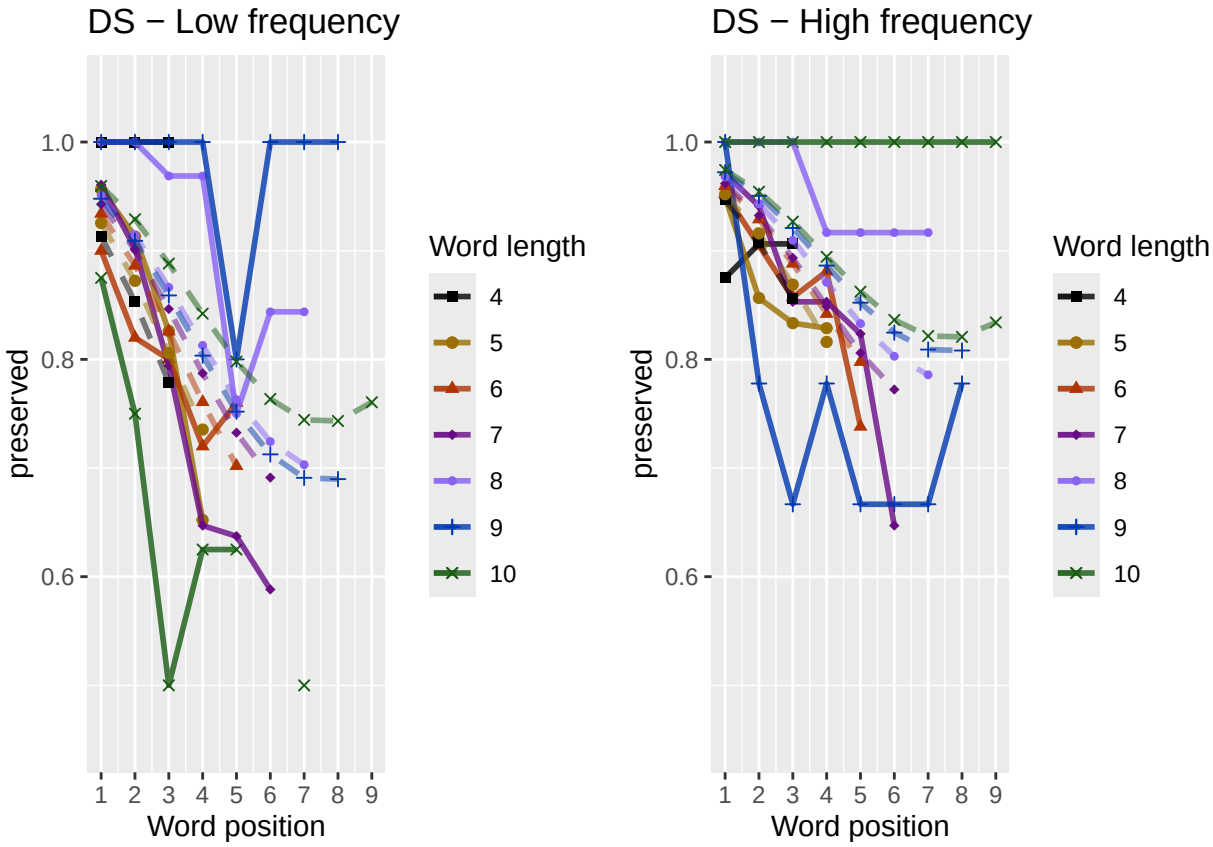
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

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

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

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




```
# only main effects
MEModelEquations<-c(
  "preserved ~ CumPres",
  "preserved ~ CumErr",
  "preserved ~ (I(pos^2)+pos)",
```

```
  "preserved ~ pos",
  "preserved ~ stimlen",
  "preserved ~ 1"
)
MERes<-TestModels(MEModelEquations,PosDat)
```

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.813
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.2     AIC: 659.9
## log likelihood:  -304.5861
## Nagelkerke R2:  0.3663401
## % pres/err predicted correctly:  -176.3984
## % of predictable range [ (model-null)/(1-null) ]:  0.3283165
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.58949      0.05802     -0.76748
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.1     AIC: 906
## log likelihood:  -402.5375
## Nagelkerke R2:  0.07256528
## % pres/err predicted correctly:  -252.6093
## % of predictable range [ (model-null)/(1-null) ]:  0.03975939
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.7343      -0.2709
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 812.4     AIC: 911.4
## log likelihood:  -406.2226
## Nagelkerke R2:  0.06042324
## % pres/err predicted correctly:  -253.9446
## % of predictable range [ (model-null)/(1-null) ]:  0.03470372
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.63981      0.05468
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 847.2     AIC: 945
## log likelihood:  -423.6106
## Nagelkerke R2:  0.001984566
## % pres/err predicted correctly:  -262.681
## % of predictable range [ (model-null)/(1-null) ]:  0.001624954
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.75
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1064 Residual
## Null Deviance:       848.4
## Residual Deviance: 848.4     AIC: 945.3
## log likelihood:  -424.1912
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -263.1102
## % 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
##     2.13258     -0.05735
```

```
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 847.3     AIC: 945.6
## log likelihood:  -423.6268
## Nagelkerke R2:  0.001929048
## % pres/err predicted correctly:  -262.7577
## % of predictable range [ (model-null)/(1-null) ]:  0.001334723
## ***************************
```

```
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(pos^2)	pos	stimlen
preserved ~ CumErr	659.9362	0.0000	1	1	0.3663402	1.435612	NA	-1.812976	NA	NA	NA
preserved ~ (I(pos^2) + pos)	905.9574	246.0212	0	0	0.0725653	3.589494	NA	NA	0.0580151	-0.7674847	NA
preserved ~ pos	911.3872	251.4510	0	0	0.0604232	2.734260	NA	NA	NA	-0.2708506	NA
preserved ~ CumPres	945.0323	285.0961	0	0	0.0019846	1.639807	0.0546842	NA	NA	NA	NA
preserved ~ 1	945.2782	285.3420	0	0	0.0000000	0.750027	NA	NA	NA	NA	NA
preserved ~ stimlen	945.5700	285.6338	0	0	0.0019292	2.132575	NA	NA	NA	NA	-0.0573469

```
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	0.8358209	134
O	0.8320513	520
P	0.9615385	13
S	0.8250000	60
V	0.8895464	338

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

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

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

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.394       -1.875
##
## Degrees of Freedom: 991 Total (i.e. Null);  990 Residual
## Null Deviance:       789.3
## Residual Deviance: 581.5     AIC: 630.9
## log likelihood:  -290.7462
## Nagelkerke R2:  0.3443873
## % pres/err predicted correctly:  -169.1688
## % of predictable range [ (model-null)/(1-null) ]:  0.3074851
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.79508      0.06916     -0.87009
##
## Degrees of Freedom: 991 Total (i.e. Null);  989 Residual
## Null Deviance:       789.3
## Residual Deviance: 745.3     AIC: 839.7
## log likelihood:  -372.6613
## Nagelkerke R2:  0.07896283
## % pres/err predicted correctly:  -234.1432
## % of predictable range [ (model-null)/(1-null) ]:  0.0430669
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.7690      -0.2786
##
## Degrees of Freedom: 991 Total (i.e. Null);  990 Residual
## Null Deviance:       789.3
## Residual Deviance: 754.4     AIC: 846.9
## log likelihood:  -377.203
## Nagelkerke R2:  0.06292473
## % pres/err predicted correctly:  -235.9533
## % of predictable range [ (model-null)/(1-null) ]:  0.03570019
## ***************************
```

```
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.752
##
## Degrees of Freedom: 991 Total (i.e. Null);  991 Residual
## Null Deviance:       789.3
## Residual Deviance: 789.3     AIC: 879.9
## log likelihood:  -394.631
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -244.7258
## % 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
##     2.08793     -0.05037
##
## Degrees of Freedom: 991 Total (i.e. Null);  990 Residual
## Null Deviance:       789.3
## Residual Deviance: 788.4     AIC: 880.5
## log likelihood:  -394.2237
## Nagelkerke R2:  0.001495787
## % pres/err predicted correctly:  -244.4502
## % of predictable range [ (model-null)/(1-null) ]:  0.001121541
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.69745      0.02824
##
## Degrees of Freedom: 991 Total (i.e. Null);  990 Residual
## Null Deviance:       789.3
## Residual Deviance: 789   AIC: 881.2
## log likelihood:  -394.4956
## Nagelkerke R2:  0.0004976014
## % pres/err predicted correctly:  -244.5949
## % of predictable range [ (model-null)/(1-null) ]:  0.0005326331
## ***************************
write.csv(SimpSyllMEAICSummary,
          paste0(TablesDir,CurPat,"_",CurTask,
```

```
                "_OV1_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	630.8674	0.0000	1	1	0.3443873	3.394388	NA	-1.875446	NA	NA	NA
preserved ~ (I(pos^2) + pos)	839.6860	208.8186	0	0	0.0789628	3.795083	NA	NA	0.069155	-0.8700905	NA
preserved ~ pos	846.8952	216.0278	0	0	0.0629242	2.769015	NA	NA	NA	-0.2786423	NA
preserved ~ 1	879.9262	249.0588	0	0	0.0000000	1.751569	NA	NA	NA	NA	NA
preserved ~ stimlen	880.4935	249.6261	0	0	0.0014958	2.087926	NA	NA	NA	NA	-0.0503735
preserved ~ CumPres	881.2131	250.3457	0	0	0.0004976	1.697449	0.0282413	NA	NA	NA	NA

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.325       -1.945
##
## Degrees of Freedom: 857 Total (i.e. Null);  856 Residual
## Null Deviance:       674.8
## Residual Deviance: 522.4     AIC: 573.9
## log likelihood:  -261.1932
## Nagelkerke R2:  0.2989005
```

```
## % pres/err predicted correctly:  -151.8595
## % of predictable range [ (model-null)/(1-null) ]:  0.2704348
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.79324      0.06429     -0.84719
##
## Degrees of Freedom: 857 Total (i.e. Null);  855 Residual
## Null Deviance:       674.8
## Residual Deviance: 631.2     AIC: 714.6
## log likelihood:  -315.5841
## Nagelkerke R2:  0.09108051
## % pres/err predicted correctly:  -198.1206
## % of predictable range [ (model-null)/(1-null) ]:  0.04964082
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##       2.862       -0.300
##
## Degrees of Freedom: 857 Total (i.e. Null);  856 Residual
## Null Deviance:       674.8
## Residual Deviance: 638.5     AIC: 719.8
## log likelihood:  -319.2553
## Nagelkerke R2:  0.07608134
## % pres/err predicted correctly:  -199.5148
## % of predictable range [ (model-null)/(1-null) ]:  0.04298626
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.772
##
## Degrees of Freedom: 857 Total (i.e. Null);  857 Residual
## Null Deviance:       674.8
## Residual Deviance: 674.8     AIC: 753.4
## log likelihood:  -337.4079
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -208.5214
## % 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
##     2.21738     -0.06675
##
## Degrees of Freedom: 857 Total (i.e. Null);  856 Residual
## Null Deviance:       674.8
## Residual Deviance: 673.6     AIC: 753.7
## log likelihood:  -336.7869
## Nagelkerke R2:  0.002656219
## % pres/err predicted correctly:  -208.1665
## % of predictable range [ (model-null)/(1-null) ]:  0.001693629
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.87045     -0.05871
##
## Degrees of Freedom: 857 Total (i.e. Null);  856 Residual
## Null Deviance:       674.8
## Residual Deviance: 674   AIC: 755.4
## log likelihood:  -337.014
## Nagelkerke R2:  0.001685241
## % pres/err predicted correctly:  -208.4685
## % of predictable range [ (model-null)/(1-null) ]:  0.0002525393
## **************************
```

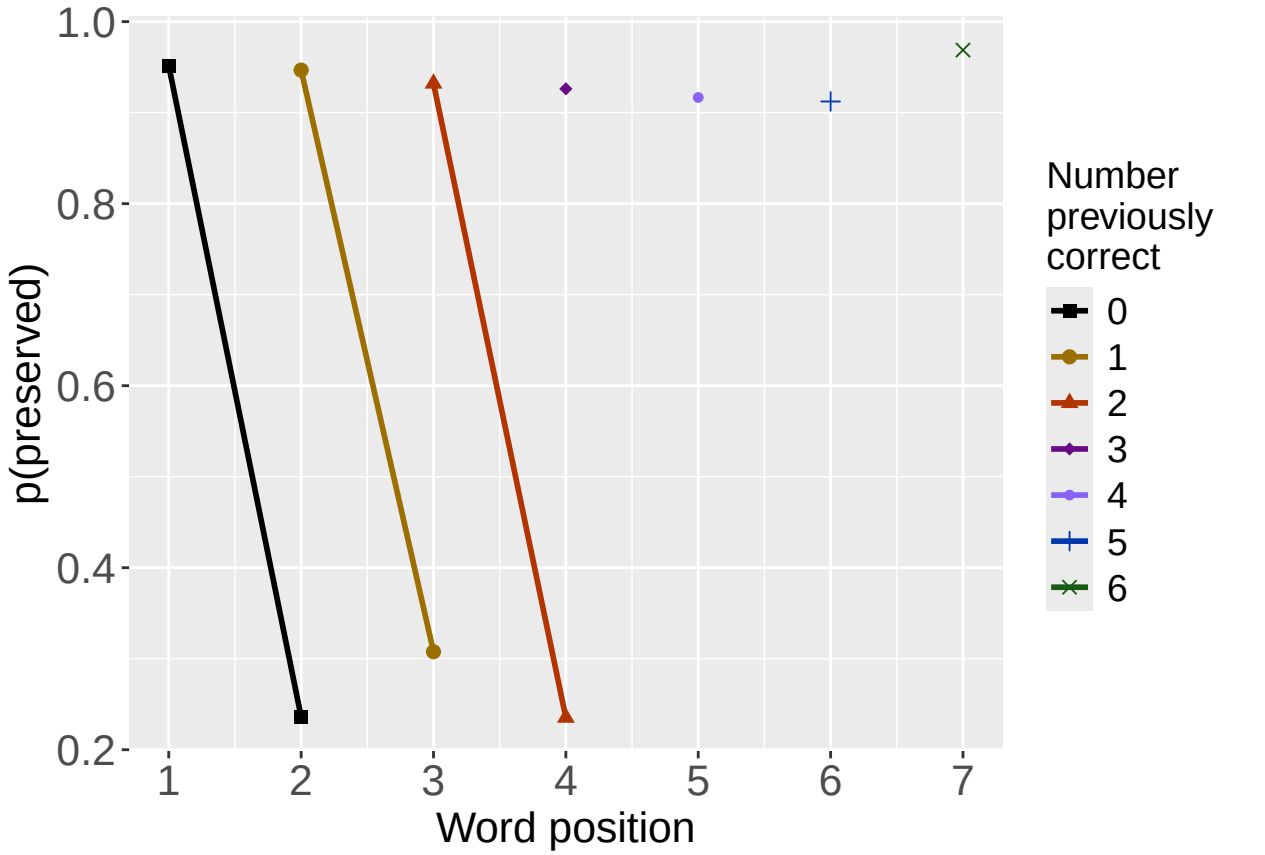
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	573.8891	0.0000	1	1	0.298900	2.324575	NA	-1.944848	NA	NA	NA
preserved ~ (I(pos^2) + pos)	714.5860	140.6969	0	0	0.091080	3.793240	NA	NA	0.0642924	-0.8471890	NA
preserved ~ pos	719.7943	145.9052	0	0	0.076081	3.862056	NA	NA	NA	-0.3000007	NA
preserved ~ 1	753.3907	179.5016	0	0	0.000000	1.771957	NA	NA	NA	NA	NA
preserved ~ stimlen	753.6765	179.7874	0	0	0.002656	2.217376	NA	NA	NA	NA	-0.0667474
preserved ~ CumPres	755.4297	181.5406	0	0	0.001685	2.870455	-0.0587112	NA	NA	NA	NA

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

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

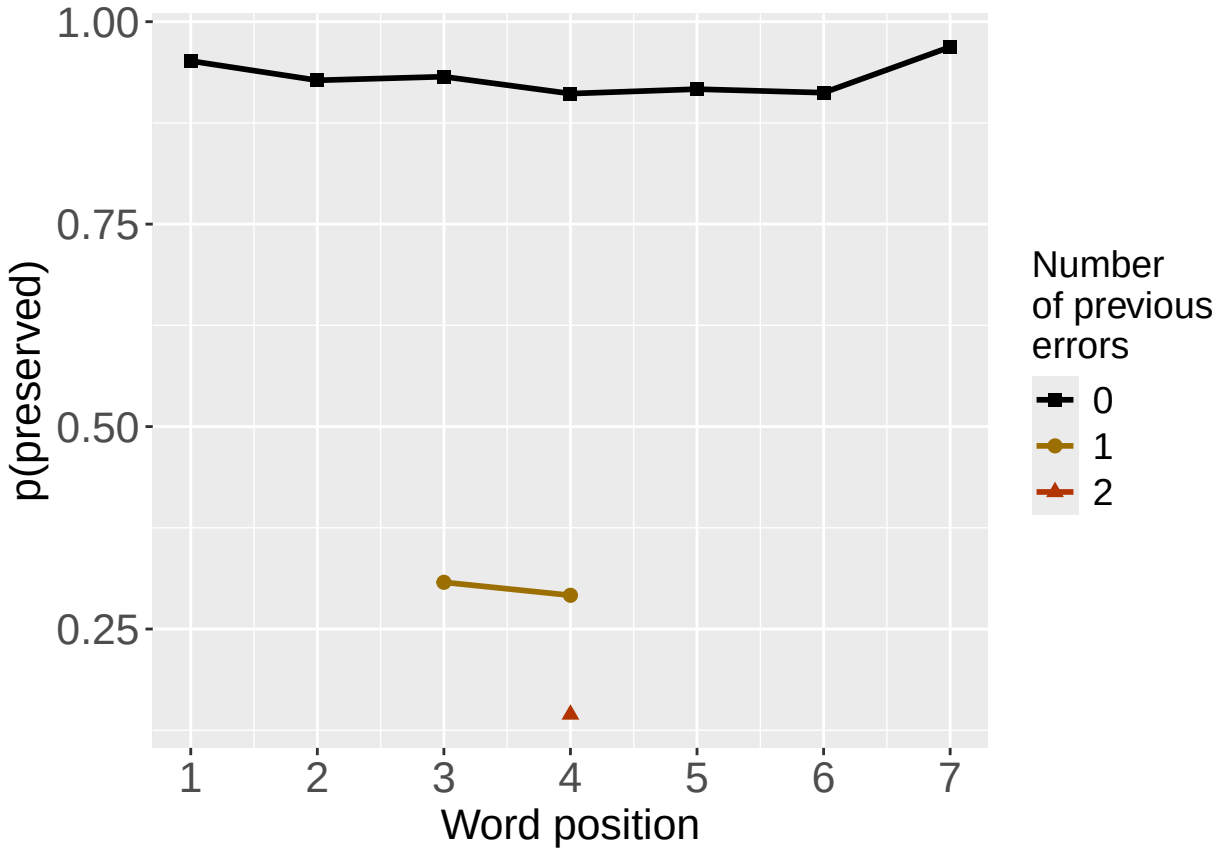
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

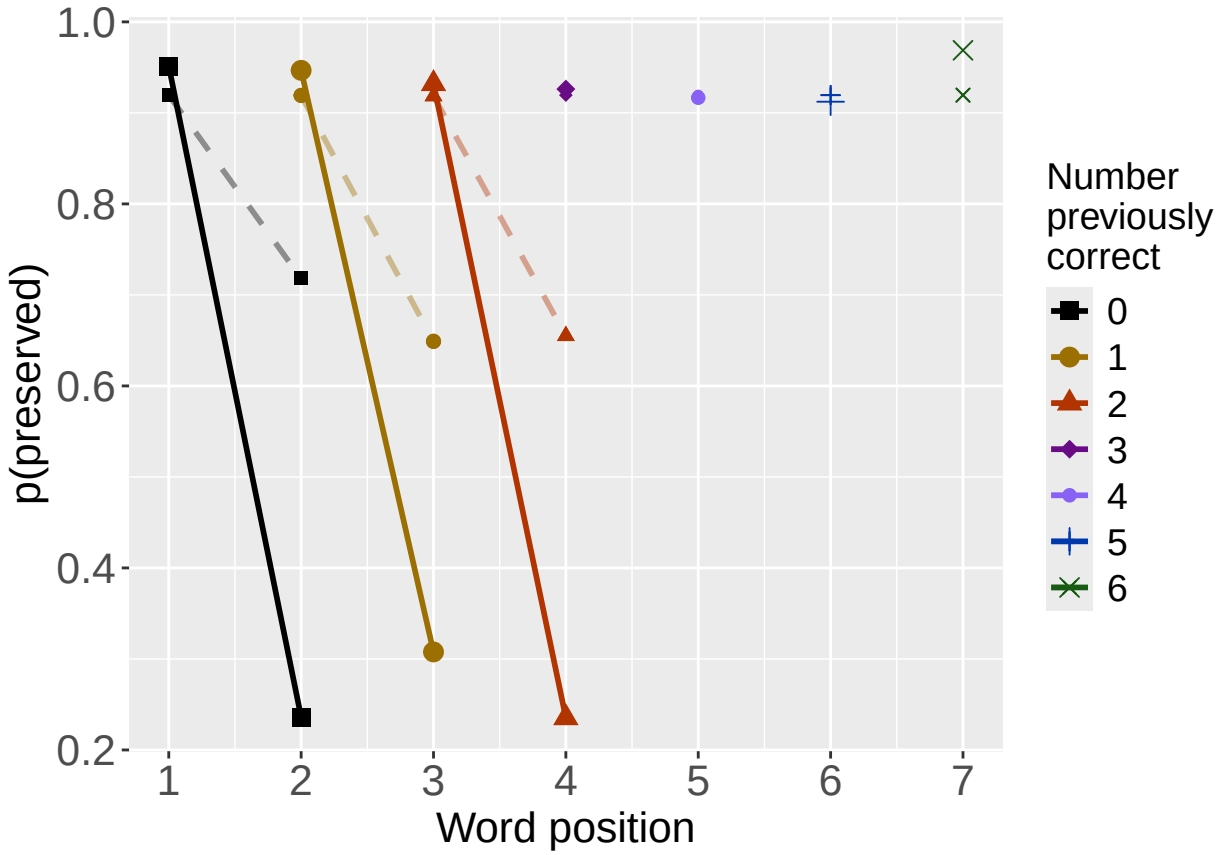
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

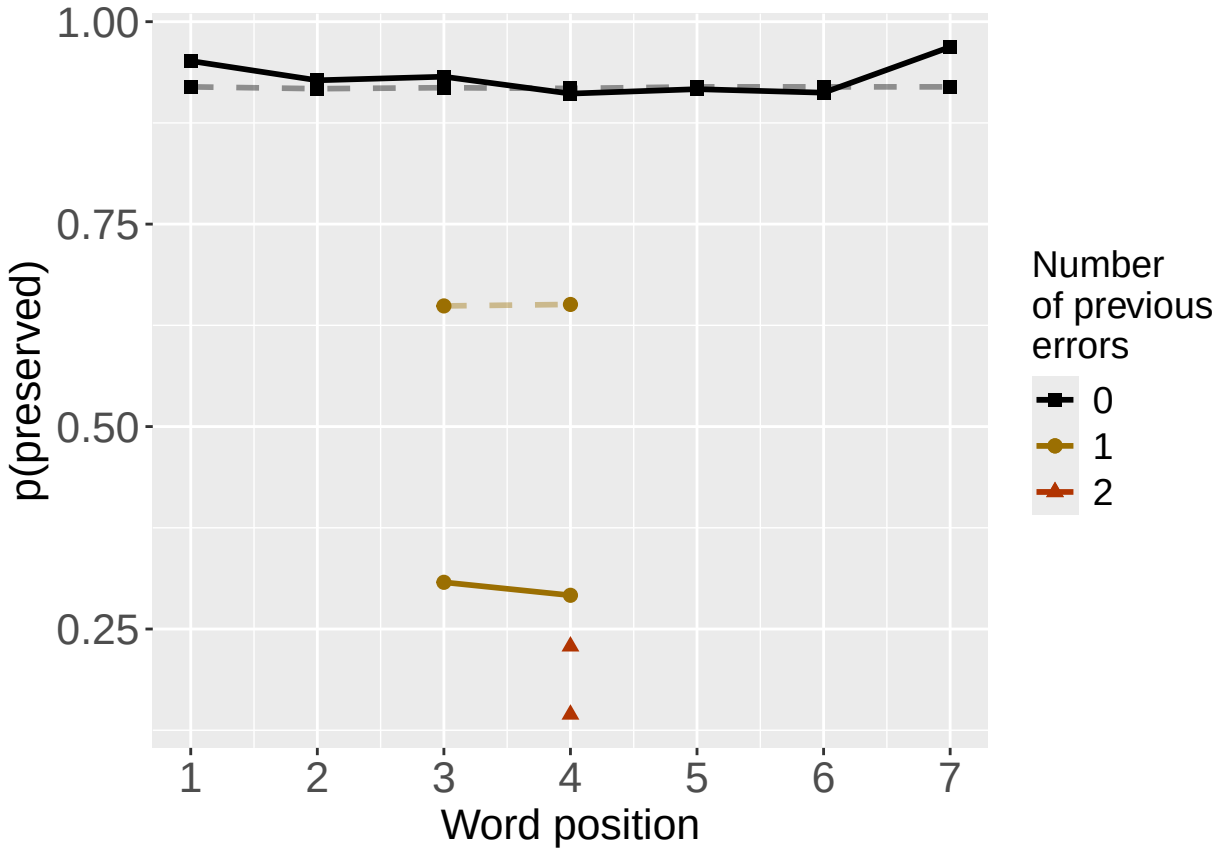
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      3.7089      -1.8620       0.1082      -0.8293
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 594.9     AIC: 648.6
## log likelihood:  -297.4535
## Nagelkerke R2:  0.3856948
## % pres/err predicted correctly:  -172.04
## % of predictable range [ (model-null)/(1-null) ]:  0.3448187
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.813
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.2     AIC: 659.9
## log likelihood:  -304.5861
## Nagelkerke R2:  0.3663401
## % pres/err predicted correctly:  -176.3984
## % of predictable range [ (model-null)/(1-null) ]:  0.3283165
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.58949      0.05802     -0.76748
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 805.1     AIC: 906
## log likelihood:  -402.5375
## Nagelkerke R2:  0.07256528
## % pres/err predicted correctly:  -252.6093
## % of predictable range [ (model-null)/(1-null) ]:  0.03975939
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	648.5635	0.00000	1.0000000	0.9966194	0.3856948	3.708911	-1.861981	0.1081621	-0.8293484

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	659.9362	11.37263	0.0033921	0.0033806	0.3663401	2.435612	-1.812976	NA	NA
preserved ~ I(posˆ2) + pos	905.9574	257.39382	0.0000000	0.0000000	0.0725653	3.589494	NA	0.0580151	-0.7674847

```
########
# 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)       CumErr      stimlen
##       1.722       -1.845         0.111
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 606.4     AIC: 659.9
## log likelihood:  -303.2003
## Nagelkerke R2:  0.3701209
## % pres/err predicted correctly:  -175.7757
## % of predictable range [ (model-null)/(1-null) ]:  0.3306744
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.813
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.2     AIC: 659.9
## log likelihood:  -304.5861
## Nagelkerke R2:  0.3663401
## % pres/err predicted correctly:  -176.3984
## % of predictable range [ (model-null)/(1-null) ]:  0.3283165
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.13258     -0.05735
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 847.3     AIC: 945.6
## log likelihood:  -423.6268
## Nagelkerke R2:  0.001929048
## % pres/err predicted correctly:  -262.7577
## % of predictable range [ (model-null)/(1-null) ]:  0.001334723
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	659.8598	0.0000000	1.000000	0.5095487	0.3701209	1.722165	-1.844903	0.1109654
preserved ~ CumErr	659.9362	0.0763988	0.962521	0.4904513	0.3663401	2.435612	-1.812976	NA
preserved ~ stimlen	945.5700	285.71021530.000000		0.0000000	0.0019290	2.132575	NA	-0.0573469

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.813
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
```

```
## Null Deviance:       848.4
## Residual Deviance: 609.2     AIC: 659.9
## log likelihood:  -304.5861
## Nagelkerke R2:  0.3663401
## % pres/err predicted correctly:  -176.3984
## % of predictable range [ (model-null)/(1-null) ]:  0.3283165
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##     2.41369     -1.81210      0.01052
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.1     AIC: 661.7
## log likelihood:  -304.5716
## Nagelkerke R2:  0.3663796
## % pres/err predicted correctly:  -176.3367
## % of predictable range [ (model-null)/(1-null) ]:  0.3285504
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     1.63981      0.05468
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 847.2     AIC: 945
## log likelihood:  -423.6106
## Nagelkerke R2:  0.001984566
## % pres/err predicted correctly:  -262.681
## % of predictable range [ (model-null)/(1-null) ]:  0.001624954
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr	659.9362	0.000000	1.0000000	0.7083335	0.3663401	2.435612	-1.812976	NA
preserved ~ CumErr + CumPres	661.7108	1.774608	0.4117644	0.2916665	0.3663796	2.413686	-1.812098	0.0105207
preserved ~ CumPres	945.0323	285.096085	0.0000000	0.0000000	0.0019846	1.639807	NA	0.0546842

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

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

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

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

```
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.813
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.2     AIC: 659.9
## log likelihood:  -304.5861
## Nagelkerke R2:  0.3663401
## % pres/err predicted correctly:  -176.3984
## % of predictable range [ (model-null)/(1-null) ]:  0.3283165
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     2.40317     -1.82262      0.01052
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.1     AIC: 661.7
## log likelihood:  -304.5716
## Nagelkerke R2:  0.3663796
## % pres/err predicted correctly:  -176.3367
## % of predictable range [ (model-null)/(1-null) ]:  0.3285504
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.7343      -0.2709
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 812.4     AIC: 911.4
## log likelihood:  -406.2226
## Nagelkerke R2:  0.06042324
## % pres/err predicted correctly:  -253.9446
## % of predictable range [ (model-null)/(1-null) ]:  0.03470372
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr	659.9362	0.000000	1.0000000	0.7083335	0.3663401	2.435612	-1.812976	NA
preserved ~ CumErr + pos	661.7108	1.774608	0.4117644	0.2916665	0.3663796	2.403166	-1.822618	0.0105207
preserved ~ pos	911.3872	251.450987	0.0000000	0.0000000	0.0604232	2.734260	NA	-0.2708506

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	648.5635	0.0000000	1.0000000	0.9966619	0.4385694	2.8708911	-1.861981	0.1081621	-0.8293484	NA	NA
preserved ~ CumErr + stimlen	659.8598	0.0000000	1.0000000	0.0050954	0.3701201	2.9722165	-1.844903	NA	NA	0.1109654	NA
preserved ~ CumErr	659.9362	1.3726285	0.9503392	0.4003380	0.3663402	2.435612	-1.812976	NA	NA	NA	NA
preserved ~ CumErr	659.9362	2.0763988	0.9625210	0.4904513	0.3663402	2.435612	-1.812976	NA	NA	NA	NA
preserved ~ CumErr	659.9362	2.0000000	1.0000000	0.7083335	0.3663402	2.435612	-1.812976	NA	NA	NA	NA
preserved ~ CumErr	659.9362	2.0000000	1.0000000	0.7083335	0.3663402	2.435612	-1.812976	NA	NA	NA	NA
preserved ~ CumErr + pos	661.7108	3.7746078	0.4117644	0.2916665	0.3663796	2.403166	-1.822618	NA	0.0105207	NA	NA
preserved ~ CumErr + CumPres	661.7108	3.7746078	0.4117644	0.2916665	0.3663796	2.6413686	-1.812098	NA	NA	NA	0.0105207
preserved ~ I(pos^2) + pos	905.9574	257.3938199	0.0000000	0.0000000	0.0072565	3.3589494	NA	0.0580151	-0.7674847	NA	NA
preserved ~ pos	911.3872	251.4509869	0.0000000	0.0000000	0.0604232	2.734260	NA	NA	-0.2708506	NA	NA
preserved ~ CumPres	945.0323	285.0960856	0.0000000	0.0000000	0.0001984	6.6639807	NA	NA	NA	NA	0.0546842
preserved ~ stimlen	945.5702	285.7102156	0.0000000	0.0000000	0.0001929	2.0132575	NA	NA	NA	-0.0573469	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##      3.7237      -1.8423       0.1122      -0.8486       0.1092
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 592.3     AIC: 648.1
## log likelihood:  -296.1741
## Nagelkerke R2:  0.3891393
## % pres/err predicted correctly:  -171.375
## % of predictable range [ (model-null)/(1-null) ]:  0.3473367
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      3.7089      -1.8620       0.1082      -0.8293
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 594.9     AIC: 648.6
## log likelihood:  -297.4535
## Nagelkerke R2:  0.3856948
## % pres/err predicted correctly:  -172.04
## % of predictable range [ (model-null)/(1-null) ]:  0.3448187
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##      3.0435      -1.8245       0.1027      -0.8178       0.1088       0.1370
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1059 Residual
## Null Deviance:       848.4
## Residual Deviance: 590.5     AIC: 649.8
## log likelihood:  -295.2531
## Nagelkerke R2:  0.3916138
## % pres/err predicted correctly:  -171.2553
## % of predictable range [ (model-null)/(1-null) ]:  0.34779
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.30113     -1.85357      0.10180     -0.80791      0.06484
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 594.2     AIC: 650.8
## log likelihood:  -297.0985
## Nagelkerke R2:  0.3866515
## % pres/err predicted correctly:  -172.018
## % of predictable range [ (model-null)/(1-null) ]:  0.344902
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##        1.75
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1064 Residual
## Null Deviance:       848.4
## Residual Deviance: 848.4     AIC: 945.3
## log likelihood:  -424.1912
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -263.1102
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + log_freq	648.1178	3.0000000	1.0000000	0.1025770	0.3891393	3.723733	- 1.842276	0.1122436	- 0.8486330	0.1091679	NA
preserved ~ CumErr + I(pos^2) + pos	648.5635	3.4446275	8.8000004	0.8220641	0.3856948	3.708911	- 1.861981	0.1081621	- 0.8293484	NA	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	649.8448	4.7275289	9.4215721	1.6971563	0.3916138	3.843511	- 1.824463	0.1027008	- 0.8178316	0.1370202	0.2087511
preserved ~ CumErr + I(pos^2) + pos + stimlen	650.7929	5.6756575	5.2624148	1.0564243	0.3866535	3.801125	- 1.853569	0.1018000	- 0.8079088	NA	0.0648383
preserved ~ 1	945.2782	297.1609410	5.0000000	0.0000000	0.0000000	1.750027	NA	NA	NA	NA	NA

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

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

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

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

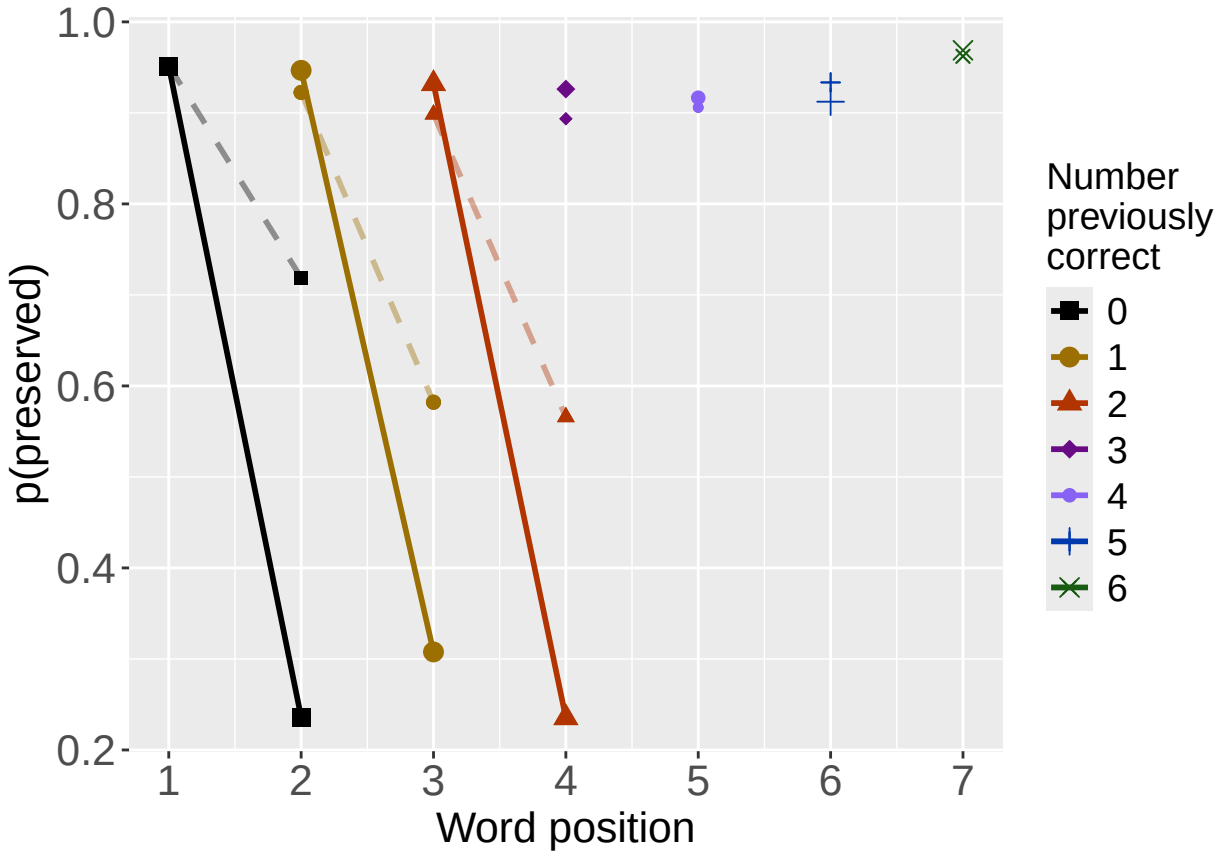
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## CumErr    1   797.20 850.97
## I(pos^2)  1   607.47 661.24
## pos       1   605.65 659.42
## log_freq  1   594.91 648.68
## <none>        592.35 648.12
```

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

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

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

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

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

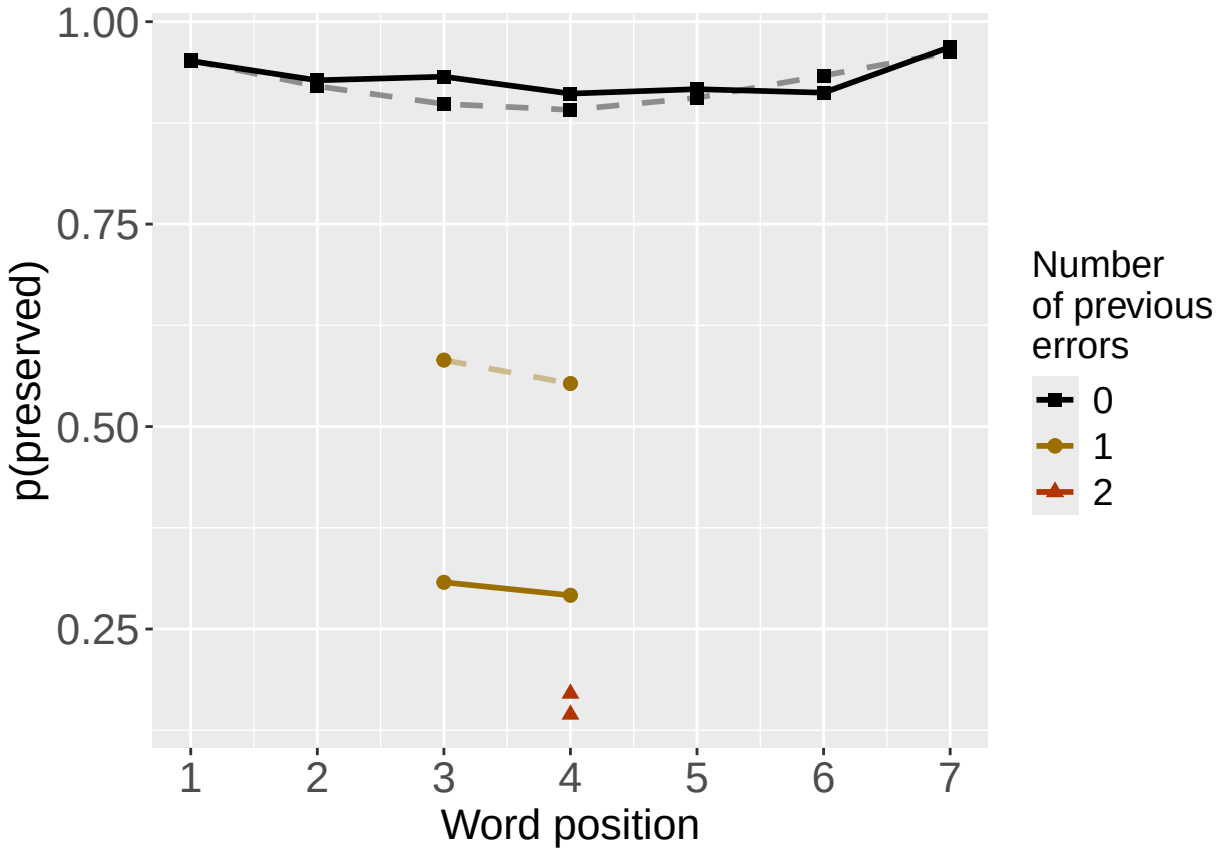
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.813
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1063 Residual
## Null Deviance:       848.4
## Residual Deviance: 609.2     AIC: 659.9
## log likelihood:  -304.5861
```

```
## Nagelkerke R2:  0.3663401
## % pres/err predicted correctly:  -176.3984
## % of predictable range [ (model-null)/(1-null) ]:  0.3283165
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)
##    2.322167    -1.883424     0.009579
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1062 Residual
## Null Deviance:       848.4
## Residual Deviance: 607.7     AIC: 659
## log likelihood:  -303.8489
## Nagelkerke R2:  0.3683524
## % pres/err predicted correctly:  -175.5783
## % of predictable range [ (model-null)/(1-null) ]:  0.3314216
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##      3.7089      -1.8620       0.1082      -0.8293
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1061 Residual
## Null Deviance:       848.4
## Residual Deviance: 594.9     AIC: 648.6
## log likelihood:  -297.4535
## Nagelkerke R2:  0.3856948
## % pres/err predicted correctly:  -172.04
## % of predictable range [ (model-null)/(1-null) ]:  0.3448187
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##      3.7237      -1.8423       0.1122      -0.8486       0.1092
##
## Degrees of Freedom: 1064 Total (i.e. Null);  1060 Residual
## Null Deviance:       848.4
## Residual Deviance: 592.3     AIC: 648.1
## log likelihood:  -296.1741
## Nagelkerke R2:  0.3891393
## % pres/err predicted correctly:  -171.375
## % of predictable range [ (model-null)/(1-null) ]:  0.3473367
```

```
## ***************************
```

```
## `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()`).
```

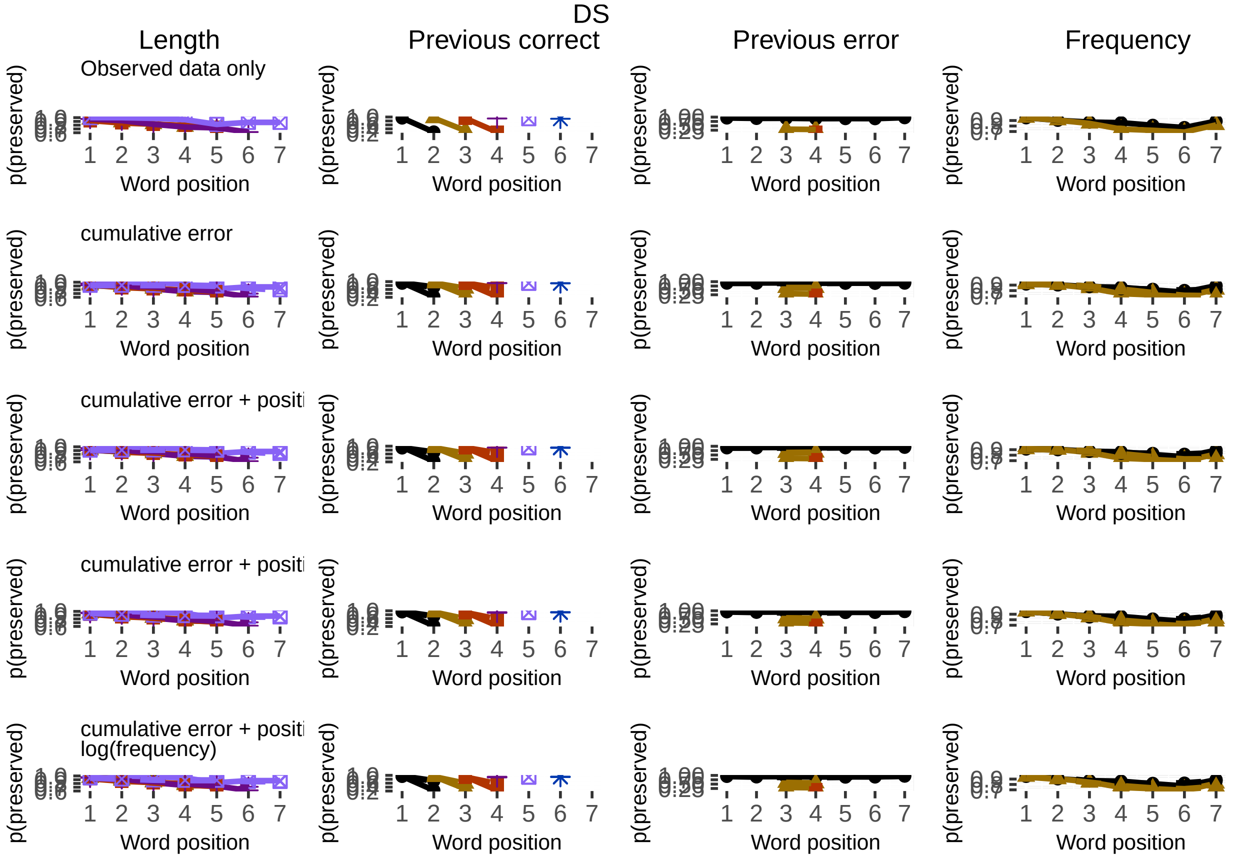
```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
```

```
## 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)
```

	CumErr	I(pos^2)	pos	log_freq
McFadden	0.2807090	0.0163446	0.0199592	0.0064982
SquaredCorrelation	0.2143256	0.0129958	0.0163823	0.0054363
Nagelkerke	0.3647614	0.0221176	0.0278810	0.0092521
Estrella	0.2541616	0.0147372	0.0178988	0.0058095

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

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

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

```
##                                                                 model deviance
## CumErr + I(pos^2) + pos + log_freq CumErr + I(pos^2) + pos + log_freq 592.3482
## CumErr + I(pos^2) + pos                       CumErr + I(pos^2) + pos 594.9071
## CumErr + I(pos^2)                                   CumErr + I(pos^2) 607.6979
## CumErr                                                         CumErr 609.1721
## null                                                             null 848.3825
##                                   deviance_explained percent_explained
## CumErr + I(pos^2) + pos + log_freq          256.0342          30.17911
## CumErr + I(pos^2) + pos                     253.4754          29.87749
## CumErr + I(pos^2)                           240.6846          28.36982
## CumErr                                      239.2103          28.19605
## null                                          0.0000           0.00000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + I(pos^2) + pos + log_freq                     100.00000              0.9994206
## CumErr + I(pos^2) + pos                                 99.00058              4.9957284
## CumErr + I(pos^2)                                       94.00485              0.5758142
## CumErr                                                  93.42904             93.4290368
## null                                                          NA              0.0000000
```

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

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

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

	deviance	deviance_explained
CumErr + I(pos^2) + pos + log_freq	592.3482	256.0342
CumErr + I(pos^2) + pos	594.9071	253.4754
CumErr + I(pos^2)	607.6979	240.6846
CumErr	609.1721	239.2103
null	848.3825	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + I(pos^2) + pos + log_freq	30.17911	100.00000	0.9994206
CumErr + I(pos^2) + pos	29.87749	99.00058	4.9957284
CumErr + I(pos^2)	28.36982	94.00485	0.5758142
CumErr	28.19605	93.42904	93.4290368
null	0.00000	NA	0.0000000

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

```
##          Nagelkerke
## CumErr   0.86026175
## I(pos^2) 0.05216265
## pos      0.06575531
## log_freq 0.02182029
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.8826919	609.1721
preserved ~ CumErr+I(pos^2)	0.8880744	607.6979
preserved ~ CumErr+I(pos^2)+pos	0.9261326	594.9071
preserved ~ CumErr+I(pos^2)+pos+log_freq	0.9274120	592.3482

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

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

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

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

```
##                                   model p_accounted_for model_deviance diff_CumErr
## 1                       preserved ~ CumErr       0.8826919       609.1721 0.000000000
## 2              preserved ~ CumErr+I(pos^2)       0.8880744       607.6979 0.005382504
## 3          preserved ~ CumErr+I(pos^2)+pos       0.9261326       594.9071 0.043440740
## 4 preserved ~ CumErr+I(pos^2)+pos+log_freq       0.9274120       592.3482 0.044720077
##   diff_CumErr+I(pos^2) diff_CumErr+I(pos^2)+pos diff_CumErr+I(pos^2)+pos+log_freq
## 1         -0.005382504             -0.043440740                      -0.044720077
## 2          0.000000000             -0.038058236                      -0.039337573
## 3          0.038058236              0.000000000                      -0.001279337
## 4          0.039337573              0.001279337                       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_CumErr	diff_CumErr+I(pos^2)	diff_CumErr+I(pos^2)+pos
preserved ~ CumErr	0.0000000	-0.0053825	-0.0434407
preserved ~ CumErr+I(pos^2)	0.0053825	0.0000000	-0.0380582
preserved ~ CumErr+I(pos^2)+pos	0.0434407	0.0380582	0.0000000
preserved ~ CumErr+I(pos^2)+pos+log_freq	0.0447201	0.0393376	0.0012793