DC - repetition - 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	503	28	121	NA	NA	652
2	59	NA	402	90	101	652
3	295	NA	156	187	14	652
4	272	NA	231	62	32	597
5	220	NA	197	70	34	521
6	199	1	135	62	21	418
7	165	NA	100	28	18	311
8	89	NA	53	25	4	171
9	74	NA	2	NA	7	83

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7714724	0.0429448	0.1855828	NA	NA	652
2	0.0904908	NA	0.6165644	0.1380368	0.1549080	652
3	0.4524540	NA	0.2392638	0.2868098	0.0214724	652
4	0.4556114	NA	0.3869347	0.1038526	0.0536013	597
5	0.4222649	NA	0.3781190	0.1343570	0.0652591	521
6	0.4760766	0.0023923	0.3229665	0.1483254	0.0502392	418

pos_factor	O	P	V	1	S	total
7	0.5305466	NA	0.3215434	0.0900322	0.0578778	311
8	0.5204678	NA	0.3099415	0.1461988	0.0233918	171
9	0.8915663	NA	0.0240964	NA	0.0843373	83

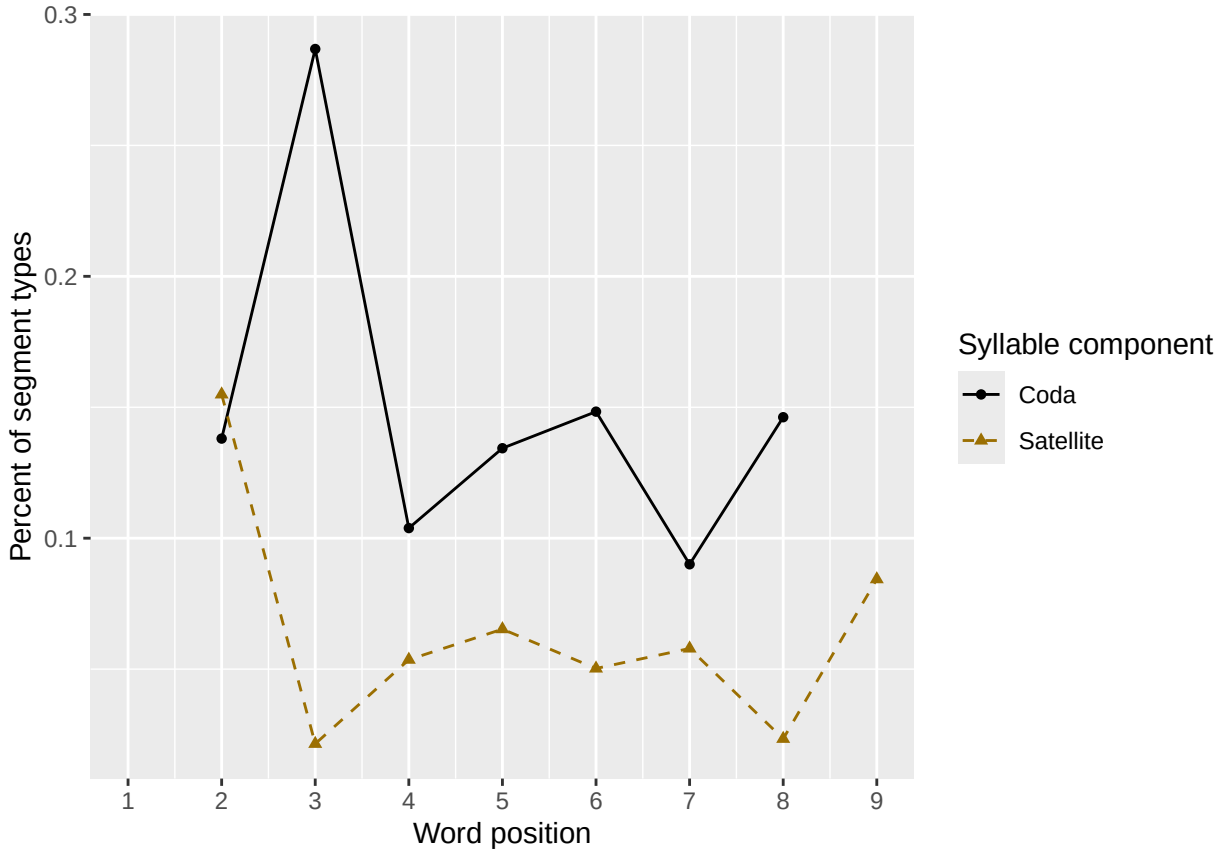
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.591 0.797 0.603 NA     NA     NA     NA     NA     NA
## 2       5 0.607 0.722 0.591  0.521 NA     NA     NA     NA     NA
## 3       6 0.553 0.563 0.474  0.597  0.324 NA     NA     NA     NA
## 4       7 0.482 0.458 0.394  0.494  0.548  0.423 NA     NA     NA
## 5       8 0.542 0.510 0.417  0.393  0.489  0.536  0.378 NA     NA
## 6       9 0.489 0.415 0.255  0.314  0.384  0.412  0.488  0.363 NA
## 7      10 0.460 0.427 0.267  0.317  0.291  0.315  0.338  0.468  0.428
```

```
# 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.
```

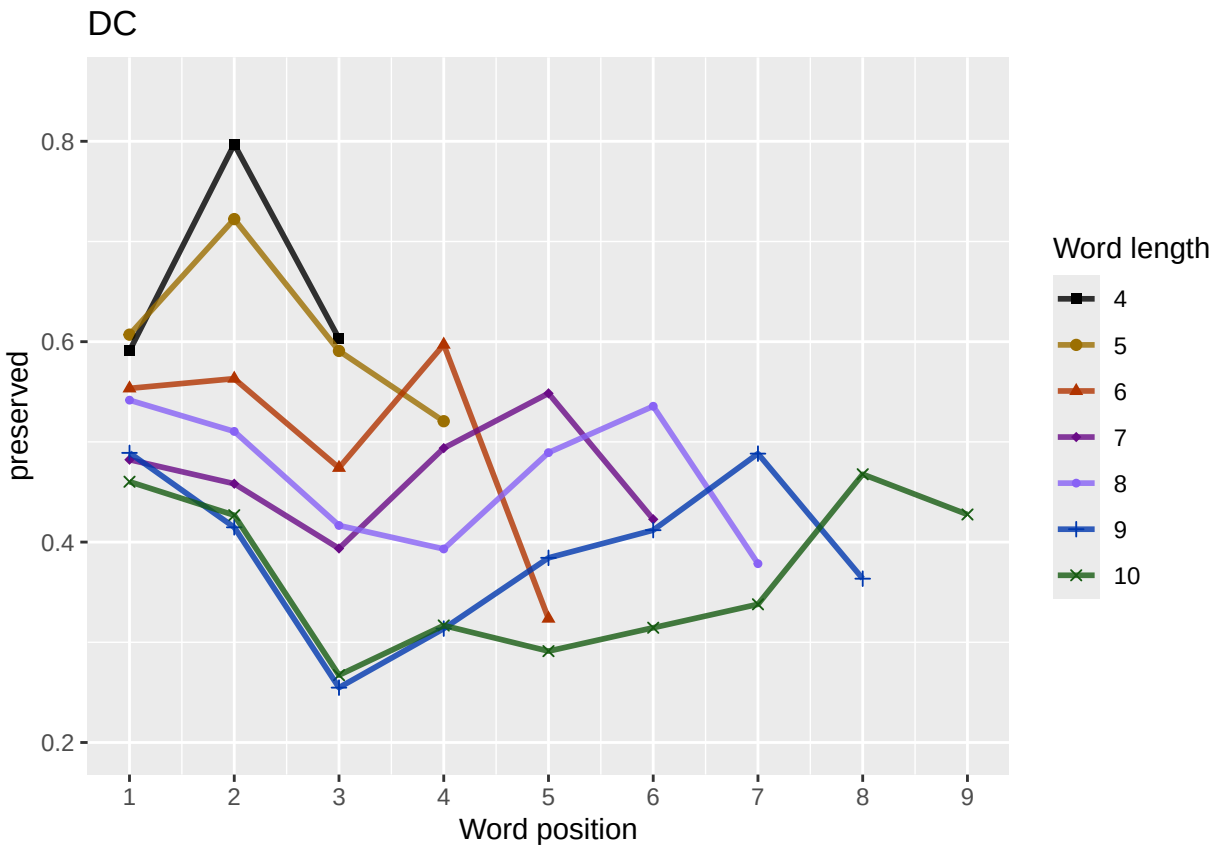
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    55    55    55    NA    NA    NA    NA    NA    NA
## 2       5    76    76    76    76    NA    NA    NA    NA    NA
## 3       6   103   103   103   103   103    NA    NA    NA    NA
## 4       7   107   107   107   107   107   107    NA    NA    NA
## 5       8   140   140   140   140   140   140   140    NA    NA
## 6       9    88    88    88    88    88    88    88    88    NA
## 7      10    83    83    83    83    83    83    83    83    83
```

```
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:  7
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.70199     -0.18321      0.02725     -0.25536
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4853  AIC: 5444
## log likelihood:  -2426.53
## Nagelkerke R2:  0.03777839
## % pres/err predicted correctly:  -1782.233
## % of predictable range [ (model-null)/(1-null) ]:  0.02398138
## **************************
## 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)
##         1.023450         -0.104969         -0.045334          0.250100          0.008039
##      stimlen:pos
##        -0.057021
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4051 Residual
## Null Deviance:       4963
## Residual Deviance: 4850  AIC: 5446
## log likelihood:  -2425.125
## Nagelkerke R2:  0.03873334
## % pres/err predicted correctly:  -1781.105
## % of predictable range [ (model-null)/(1-null) ]:  0.02459853
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.91222     -0.25175     -0.24826      0.02642
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4863  AIC: 5459
## log likelihood:  -2431.476
## Nagelkerke R2:  0.03441088
## % pres/err predicted correctly:  -1785.504
## % of predictable range [ (model-null)/(1-null) ]:  0.02219082
## **************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     1.22534     -0.16811     -0.02424
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4870  AIC: 5462
## log likelihood:  -2434.803
## Nagelkerke R2:  0.03214204
## % pres/err predicted correctly:  -1788.188
## % of predictable range [ (model-null)/(1-null) ]:  0.02072183
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.35743      0.01711     -0.22252
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4930  AIC: 5522
## log likelihood:  -2465.006
## Nagelkerke R2:  0.01136824
## % pres/err predicted correctly:  -1811.734
## % of predictable range [ (model-null)/(1-null) ]:  0.007834223
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      0.1210      -0.0747
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4937  AIC: 5528
## log likelihood:  -2468.424
## Nagelkerke R2:  0.008997405
## % pres/err predicted correctly:  -1813.461
## % of predictable range [ (model-null)/(1-null) ]:  0.006888917
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##     -0.1669
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4056 Residual
## Null Deviance:       4963
## Residual Deviance: 4963  AIC: 5557
## log likelihood:  -2481.346
## Nagelkerke R2:  -3.146332e-16
## % pres/err predicted correctly:  -1826.048
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	5444.036	0.0000000	1.0000000	0.7004757	0.0377784	0.1701985	-	-	NA	0.0272494	NA
						0.1832095	0.2553557				
preserved ~ stimlen * (I(pos^2) + pos)	5445.739	1.7025920	0.4268610	0.2990066	0.0387333	0.0234500	-	0.2500996	-	-	0.0080386
						0.1049690	0.0570207	0.0453338			

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	5459.028	14.98706	0.000055	0.000038	0.034410	1.9122193	-0.2517529	-1.2482588	0.0264163	NA	NA
preserved ~ stimlen + pos	5462.025	17.98924	0.000124	0.000086	0.032142	1.2253391	-0.1681075	-0.5242365	NA	NA	NA
preserved ~ stimlen	5463.507	19.47057	0.000059	0.000041	0.031371	1.2246165	-0.1801101	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	5521.567	77.53282	7.000000	0.000000	0.011368	2.3574333	NA	-0.2225215	NA	0.0171109	NA
preserved ~ pos	5528.487	84.45093	0.000000	0.000000	0.008997	1.1209975	NA	-0.0746982	NA	NA	NA
preserved ~ 1	5557.103	113.06633	0.000000	0.000000	0.000000	-0.1669442	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.70199     -0.18321      0.02725     -0.25536
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4853  AIC: 5444
```

```
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
```

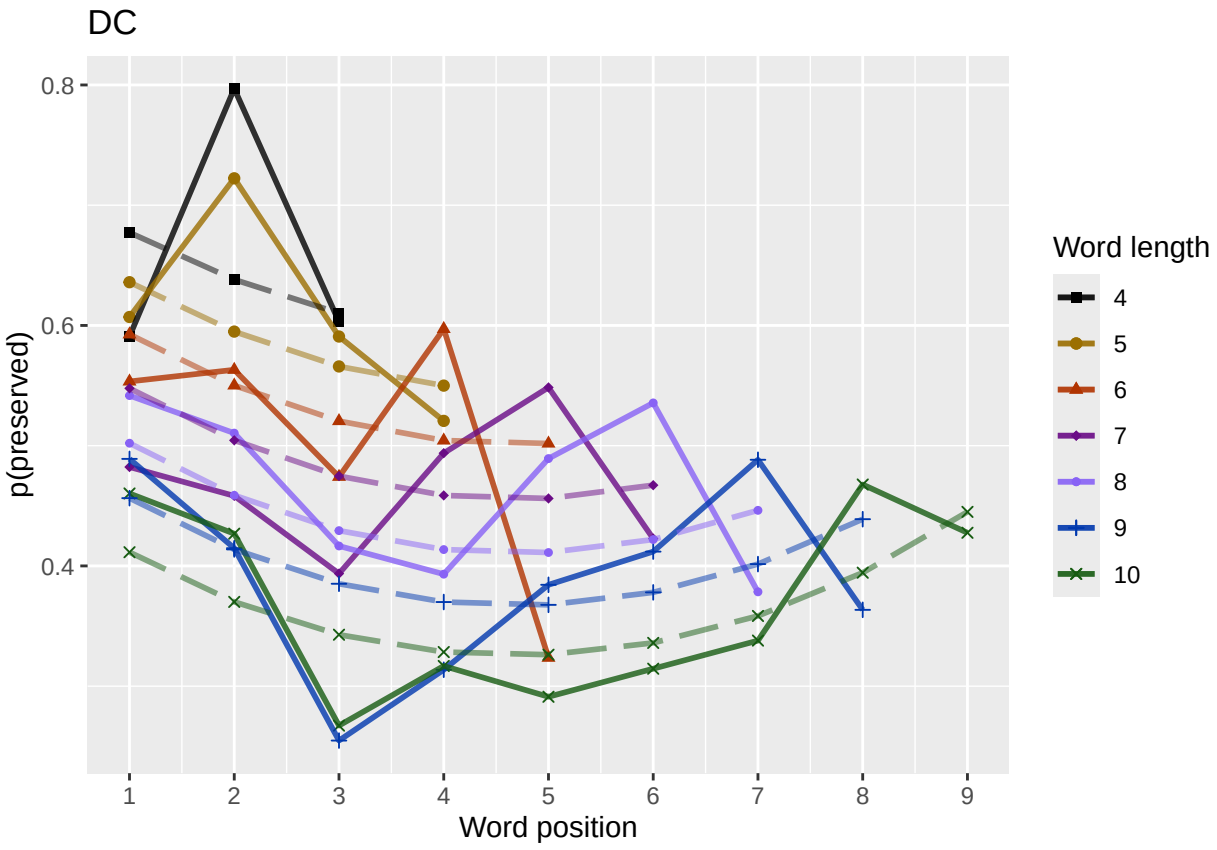
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.677 0.638 0.610 NA     NA     NA     NA     NA     NA
## 2       5 0.636 0.595 0.566  0.550 NA     NA     NA     NA     NA
## 3       6 0.593 0.550 0.520  0.504  0.502 NA     NA     NA     NA
## 4       7 0.548 0.504 0.475  0.459  0.456  0.467 NA     NA     NA
## 5       8 0.502 0.459 0.429  0.414  0.411  0.422  0.446 NA     NA
```

```
## 6       9 0.456 0.414 0.385  0.370  0.368  0.378  0.402  0.439 NA
## 7      10 0.411 0.370 0.343  0.328  0.326  0.336  0.358  0.394  0.445
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       18   652
```

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

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

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

```
## [1] "The number of responses with fragments was 18 / 652 = 2.76 percent"
```

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

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

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

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

```
## ****************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.71931     -0.18621      0.02797     -0.25484
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4026 Residual
## Null Deviance:       4935
## Residual Deviance: 4826  AIC: 5414
## log likelihood:  -2413.149
## Nagelkerke R2:  0.03768498
## % pres/err predicted correctly:  -1772.342
## % of predictable range [ (model-null)/(1-null) ]:  0.02373749
## ***************************
## 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)
##         1.058817         -0.107282         -0.033829          0.204048          0.007008
##      stimlen:pos
##        -0.053340
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4024 Residual
## Null Deviance:       4935
## Residual Deviance: 4824  AIC: 5416
## log likelihood:  -2412.086
## Nagelkerke R2:  0.03841169
## % pres/err predicted correctly:  -1771.61
## % of predictable range [ (model-null)/(1-null) ]:  0.02414061
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.86939     -0.24844     -0.22635      0.02457
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4026 Residual
## Null Deviance:       4935
## Residual Deviance: 4838  AIC: 5431
## log likelihood:  -2418.945
## Nagelkerke R2:  0.03371372
## % pres/err predicted correctly:  -1776.335
## % of predictable range [ (model-null)/(1-null) ]:  0.02153964
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos
##     1.23173     -0.17083     -0.01787
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4027 Residual
## Null Deviance:       4935
## Residual Deviance: 4844  AIC: 5433
## log likelihood:  -2421.801
## Nagelkerke R2:  0.03175277
## % pres/err predicted correctly:  -1778.653
## % of predictable range [ (model-null)/(1-null) ]:  0.02026316
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2312      -0.1796
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4028 Residual
## Null Deviance:       4935
## Residual Deviance: 4845  AIC: 5433
## log likelihood:  -2422.408
## Nagelkerke R2:  0.0313358
## % pres/err predicted correctly:  -1779.28
## % of predictable range [ (model-null)/(1-null) ]:  0.01991818
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.35367      0.01768     -0.22197
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4027 Residual
## Null Deviance:       4935
## Residual Deviance: 4906  AIC: 5494
## log likelihood:  -2452.819
## Nagelkerke R2:  0.01027628
## % pres/err predicted correctly:  -1802.848
## % of predictable range [ (model-null)/(1-null) ]:  0.006943265
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     0.11005     -0.06947
```

```
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4028 Residual
## Null Deviance:       4935
## Residual Deviance: 4913  AIC: 5502
## log likelihood:  -2456.44
## Nagelkerke R2:  0.007747764
## % pres/err predicted correctly:  -1804.689
## % of predictable range [ (model-null)/(1-null) ]:  0.005929957
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##     -0.1566
##
## Degrees of Freedom: 4029 Total (i.e. Null);  4029 Residual
## Null Deviance:       4935
## Residual Deviance: 4935  AIC: 5526
## log likelihood:  -2467.494
## Nagelkerke R2:  3.144601e-16
## % pres/err predicted correctly:  -1815.46
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	5414.276	0.000000	1.000000	0.7291989	0.0037685	-0.1719313	6 - 0.1862124	- 0.2548374	NA	0.0279744	NA
preserved ~ stimlen * (I(pos^2) + pos)	5416.259	0.983245	0.370974	0.2705121	0.0038411	-0.1705881	66 - 0.1072815	0.2040478 - 0.0533396	-	0.0070077	0.0338291
preserved ~ stimlen * pos	5430.944	16.668059	4.000240	0.0002917	0.0033713	-0.1786938	55 - 0.2484435	- 0.2263529	0.0245688	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	5433.062	28.786457	7.000088e-07	3.000060e-07	0.0317528	2.2317282	-0.1708251	-0.0178667	NA	NA	NA
preserved ~ stimlen	5433.151	18.875655	5.000079e-05	7.000058e-05	0.0313351	2.2312369	-0.1796420	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	5494.277	79.996880	0.000000e+00	0.000000e+00	0.0102763	3.3536660	NA	-0.2219750	NA	0.0176775	NA
preserved ~ pos	5501.628	87.347922	2.000000e-19	0.000000e+00	0.0077478	3.1100524	NA	-0.0694685	NA	NA	NA
preserved ~ 1	5526.071	111.795249	0.000000e+00	0.000000e+00	0.0000000	-0.1565781	NA	NA	NA	NA	NA

```
# plot no fragment data

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

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

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

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

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

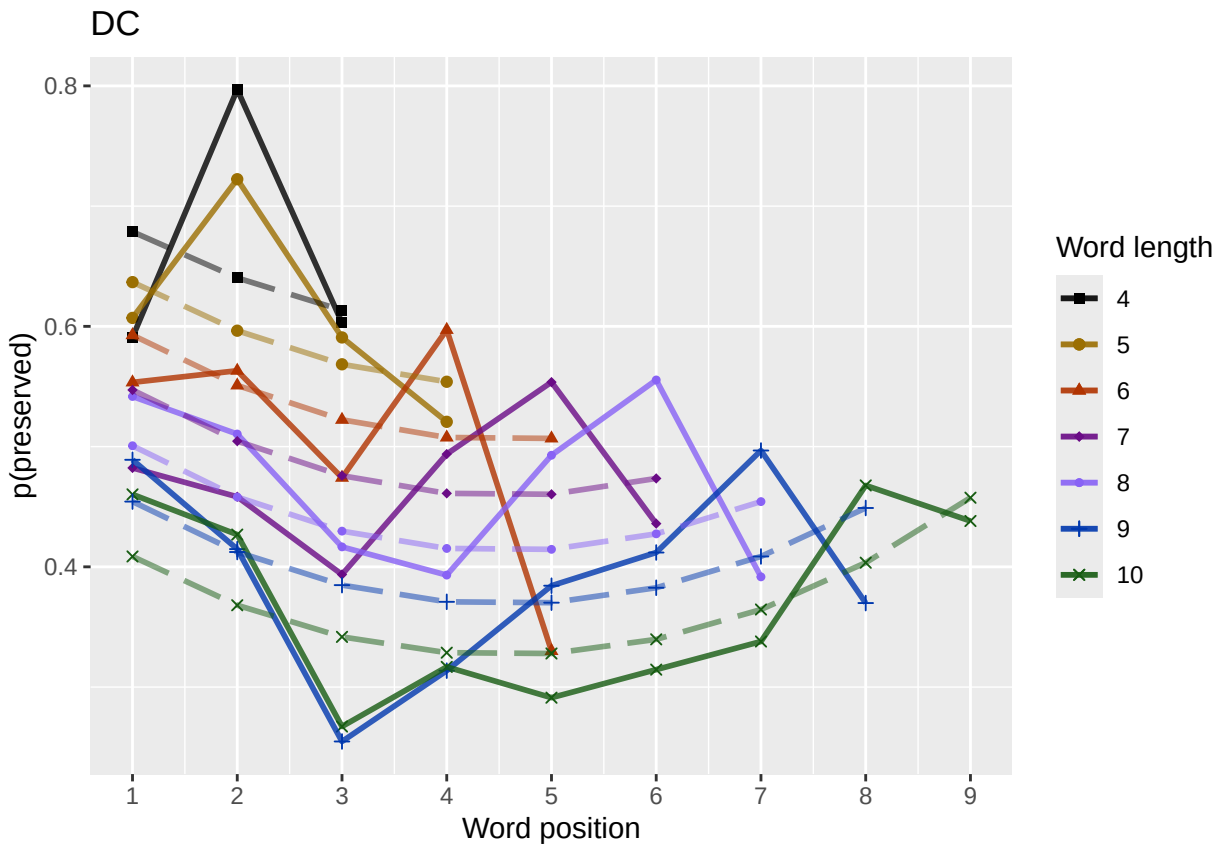
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.679 0.640 0.613 NA     NA     NA     NA     NA     NA
## 2       5 0.637 0.596 0.568  0.554 NA     NA     NA     NA     NA
## 3       6 0.593 0.551 0.522  0.508  0.507 NA     NA     NA     NA
## 4       7 0.547 0.505 0.476  0.461  0.460  0.473 NA     NA     NA
## 5       8 0.501 0.458 0.430  0.415  0.415  0.427  0.454 NA     NA
## 6       9 0.454 0.412 0.385  0.371  0.370  0.383  0.409  0.449 NA
## 7      10 0.409 0.368 0.342  0.329  0.328  0.340  0.364  0.403  0.457
```

```
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: 0.20 - 0.85"
```

```
# 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] -0.04432893
```

```
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] -0.02216607
```

```
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] 0.03028874
```

```
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] 0.06698252 0.08599143 0.09077895 0.09160026 0.09090560 0.08874023 0.08524200
## [1] "differences from min to right max for each row: "
## [1] 0.00000000 0.00000000 0.00000000 0.01103118 0.03507410 0.07124244 0.11879785
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.085242
## [1] 0.1187979
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 1.393654
```

```
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",
```

```r
            "preserved ~ (I(pos^2)+pos)*log_freq",
            "preserved ~ stimlen*log_freq + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen*log_freq + I(pos^2) + pos",
            "preserved ~ stimlen + (I(pos^2) + pos)*log_freq",
            "preserved ~ stimlen + I(pos^2) + pos + log_freq",

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

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

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

```
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          1.515863          -0.162968           0.323449           0.027742          -0.255718
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.030155           0.005031          -0.036741
##
```

```
## Degrees of Freedom: 4056 Total (i.e. Null);  4049 Residual
## Null Deviance:       4963
## Residual Deviance: 4838  AIC: 5435
## log likelihood:  -2418.971
## Nagelkerke R2:  0.04290811
## % pres/err predicted correctly:  -1775.529
## % of predictable range [ (model-null)/(1-null) ]:  0.02765055
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          1.50012          -0.16133           0.23531           0.02632          -0.24768
## stimlen:log_freq
##         -0.02436
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4051 Residual
## Null Deviance:       4963
## Residual Deviance: 4840  AIC: 5436
## log likelihood:  -2420.224
## Nagelkerke R2:  0.04205908
## % pres/err predicted correctly:  -1776.302
## % of predictable range [ (model-null)/(1-null) ]:  0.02722725
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     1.58846     -0.16872      0.02715     -0.25458      0.04583
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4052 Residual
## Null Deviance:       4963
## Residual Deviance: 4846  AIC: 5439
## log likelihood:  -2423.085
## Nagelkerke R2:  0.04011817
## % pres/err predicted correctly:  -1778.961
## % of predictable range [ (model-null)/(1-null) ]:  0.02577204
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          1.588301          -0.168727           0.027432          -0.255831           0.099814
## I(pos^2):log_freq       pos:log_freq
##          0.003312          -0.030770
```

```
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4050 Residual
## Null Deviance:       4963
## Residual Deviance: 4845  AIC: 5441
## log likelihood:  -2422.596
## Nagelkerke R2:  0.04045015
## % pres/err predicted correctly:  -1778.745
## % of predictable range [ (model-null)/(1-null) ]:  0.02589018
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.70199     -0.18321      0.02725     -0.25536
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4853  AIC: 5444
## log likelihood:  -2426.53
## Nagelkerke R2:  0.03777839
## % pres/err predicted correctly:  -1782.233
## % of predictable range [ (model-null)/(1-null) ]:  0.02398138
## ***************************
## 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)
##         1.023450         -0.104969         -0.045334          0.250100          0.008039
##      stimlen:pos
##        -0.057021
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4051 Residual
## Null Deviance:       4963
## Residual Deviance: 4850  AIC: 5446
## log likelihood:  -2425.125
## Nagelkerke R2:  0.03873334
## % pres/err predicted correctly:  -1781.105
## % of predictable range [ (model-null)/(1-null) ]:  0.02459853
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          1.03261          -0.14614           0.25082          -0.02424          -0.02633
##
```

```
## Degrees of Freedom: 4056 Total (i.e. Null);  4052 Residual
## Null Deviance:       4963
## Residual Deviance: 4856  AIC: 5453
## log likelihood:  -2427.918
## Nagelkerke R2:  0.03683426
## % pres/err predicted correctly:  -1781.743
## % of predictable range [ (model-null)/(1-null) ]:  0.02424914
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          1.03195          -0.15815           0.25080          -0.02633
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4858  AIC: 5454
## log likelihood:  -2429.047
## Nagelkerke R2:  0.0360658
## % pres/err predicted correctly:  -1782.881
## % of predictable range [ (model-null)/(1-null) ]:  0.02362623
## ****************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         1.032594         -0.146368          0.250707         -0.023773         -0.027305
##     log_freq:pos
##         0.001985
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4051 Residual
## Null Deviance:       4963
## Residual Deviance: 4856  AIC: 5454
## log likelihood:  -2427.891
## Nagelkerke R2:  0.03685258
## % pres/err predicted correctly:  -1781.712
## % of predictable range [ (model-null)/(1-null) ]:  0.02426659
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     1.11269     -0.15357     -0.02427      0.04621
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
```

```
## Null Deviance:       4963
## Residual Deviance: 4863  AIC: 5457
## log likelihood:  -2431.287
## Nagelkerke R2:  0.03453984
## % pres/err predicted correctly:  -1784.945
## % of predictable range [ (model-null)/(1-null) ]:  0.0224966
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     1.11203     -0.16560      0.04619
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4865  AIC: 5458
## log likelihood:  -2432.418
## Nagelkerke R2:  0.03376897
## % pres/err predicted correctly:  -1786.063
## % of predictable range [ (model-null)/(1-null) ]:  0.02188489
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     1.103132     -0.152018     -0.025576      0.070099     -0.006175
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4052 Residual
## Null Deviance:       4963
## Residual Deviance: 4862  AIC: 5459
## log likelihood:  -2430.982
## Nagelkerke R2:  0.03474759
## % pres/err predicted correctly:  -1784.679
## % of predictable range [ (model-null)/(1-null) ]:  0.02264248
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.91222     -0.25175     -0.24826      0.02642
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4863  AIC: 5459
## log likelihood:  -2431.476
```

```
## Nagelkerke R2:  0.03441088
## % pres/err predicted correctly:  -1785.504
## % of predictable range [ (model-null)/(1-null) ]:  0.02219082
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     1.22534     -0.16811     -0.02424
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4870  AIC: 5462
## log likelihood:  -2434.803
## Nagelkerke R2:  0.03214204
## % pres/err predicted correctly:  -1788.188
## % of predictable range [ (model-null)/(1-null) ]:  0.02072183
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##          0.336107           0.017831          -0.221633           0.149143           0.002451
##      pos:log_freq
##         -0.030391
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4051 Residual
## Null Deviance:       4963
## Residual Deviance: 4905  AIC: 5502
## log likelihood:  -2452.603
## Nagelkerke R2:  0.01993598
```

```
## % pres/err predicted correctly:  -1801.465
## % of predictable range [ (model-null)/(1-null) ]:  0.01345469
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     0.09220     -0.06750      0.07835
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4915  AIC: 5508
## log likelihood:  -2457.437
## Nagelkerke R2:  0.01660319
## % pres/err predicted correctly:  -1804.186
## % of predictable range [ (model-null)/(1-null) ]:  0.0119655
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      0.09357      -0.06908       0.12205      -0.01147
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4913  AIC: 5509
## log likelihood:  -2456.36
## Nagelkerke R2:  0.01734649
## % pres/err predicted correctly:  -1803.741
## % of predictable range [ (model-null)/(1-null) ]:  0.01220893
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.35743      0.01711     -0.22252
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4930  AIC: 5522
## log likelihood:  -2465.006
## Nagelkerke R2:  0.01136824
## % pres/err predicted correctly:  -1811.734
## % of predictable range [ (model-null)/(1-null) ]:  0.007834223
## ***************************
```

```
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.1210      -0.0747
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4937  AIC: 5528
## log likelihood:  -2468.424
## Nagelkerke R2:  0.008997405
## % pres/err predicted correctly:  -1813.461
## % of predictable range [ (model-null)/(1-null) ]:  0.006888917
## **************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##     -0.1669
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4056 Residual
## Null Deviance:       4963
## Residual Deviance: 4963  AIC: 5557
## log likelihood:  -2481.346
## Nagelkerke R2:  -3.146332e-16
## % pres/err predicted correctly:  -1826.048
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	pos:log_freq	log_freq:pos	I(pos^2)	I(pos^2):log_freq	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	5434.9[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1629679	0.3234488	-0.0301552	-0.2557179	NA	-0.0367411	0.0277417	NA	0.00503[illegible]	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	5435.9[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1613284	0.2353129	-0.02436[illegible]	-0.2476803	NA	NA	0.026323[illegible]	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	5438.4[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1687182	0.0458332	NA	-0.2545782	NA	NA	0.0271509	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	5441.6[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1687271	0.0998139	NA	-0.2558306	-0.0307695	NA	0.02740[illegible]	[illegible]3121	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	5444.9[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1832095	NA	NA	-0.2553557	NA	NA	0.0272494	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	5445.7[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1049690	NA	NA	0.2500996	NA	NA	-0.0453338	NA	NA	-0.0570207	0.0080386
preserved ~ stimlen * log_freq + pos	5452.7[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1461374	0.2508184	-0.02633[illegible]	-0.0242369	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	5454.1[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1581479	0.2508047	-0.0263305	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	5454.4[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1463679	0.2507069	-0.027304[illegible]	-0.0237729	NA	0.0019851	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	5456.7[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1535715	0.0462091	NA	-0.0242723	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	5458.2[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1656000	0.0461851	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	5458.2[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1520184	0.0700992	NA	-0.02557[illegible]	-[illegible]61751	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	5459.2[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2517529	NA	NA	-0.2482588	NA	NA	NA	NA	NA	0.026416[illegible]	NA

Model	AIC DeltaAICAICexpAICwNagR2(Intercept)stimlenlog_freqstimlen:poslog_freqlog_freq:I(pos^2)I(pos^2)log_freq:stimlen:posstimlen:I(pos^2)											
preserved ~ stimlen + pos	5462.2725100679001300032142253391		-0.1681075	NA	NA	-0.0242365	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	5463.2807582010000060003113721246165		-0.1801101	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	5502.6755229831000000001993661074		NA	0.1491453	NA	-0.2216323803911	-	NA	0.0178308245112		NA	NA	NA
preserved ~ pos + log_freq	5507.7925980056000000001660322019		NA	0.0783503	NA	-0.0675048	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	5508.7935104380000000017346535730		NA	0.1220470	NA	-0.0690841614708	-	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	5521.8669440262000000001103638574353		NA	NA	NA	-0.2225215	NA	NA	0.0171109	NA	NA	NA	NA
preserved ~ pos	5528.9437562375000000000899720975		NA	NA	NA	-0.0746982	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	5557.1223107770000000000000	-0.1669442	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          1.515863          -0.162968           0.323449           0.027742          -0.255718
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.030155           0.005031          -0.036741
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4049 Residual
## Null Deviance:       4963
## Residual Deviance: 4838  AIC: 5435
```

```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"
```

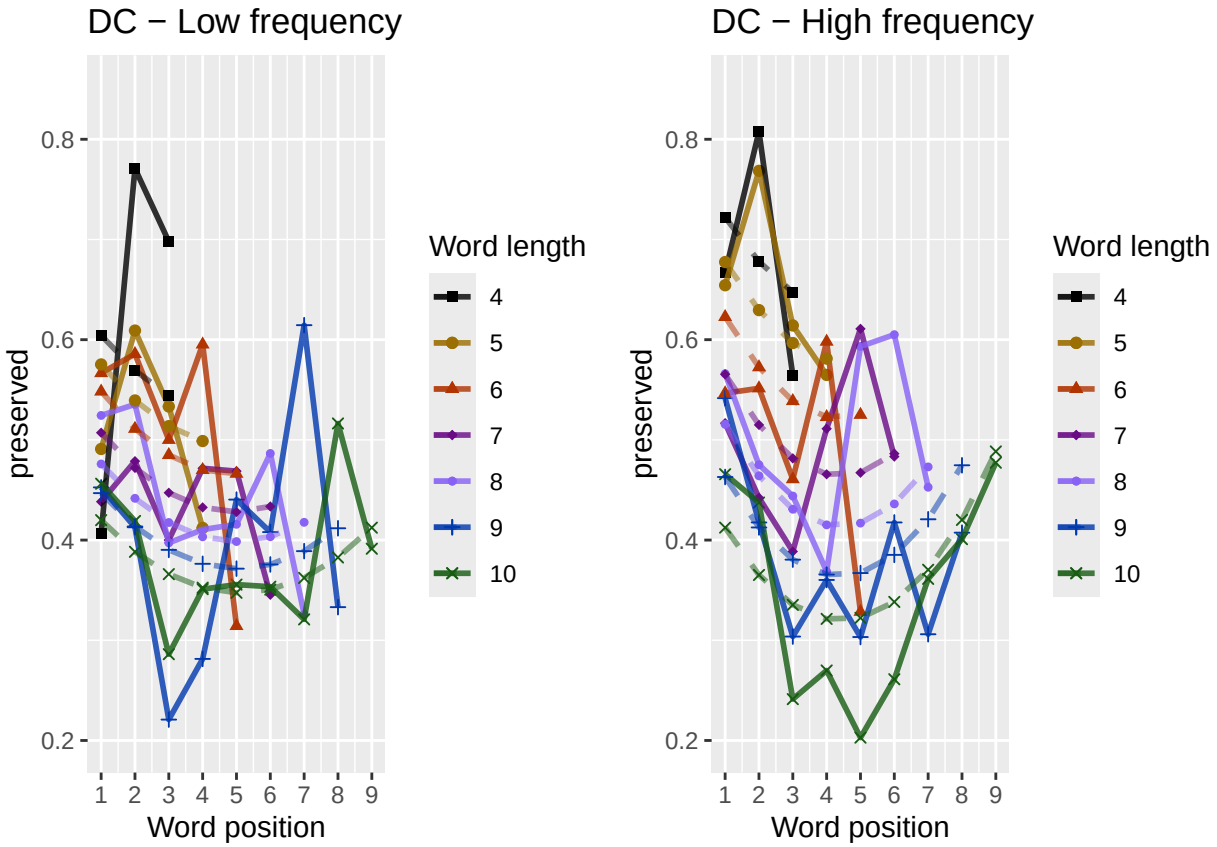
```r
PosDat$FLPFitted<-fitted(BestFLPModel)

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_preserv
```

```
## `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)
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:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##     0.02602     -0.12227
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4926  AIC: 5508
## log likelihood:  -2463.128
## Nagelkerke R2:  0.0126685
## % pres/err predicted correctly:  -1809.834
## % of predictable range [ (model-null)/(1-null) ]:  0.008874512
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.35743      0.01711     -0.22252
##
```

```
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4930  AIC: 5522
## log likelihood:  -2465.006
## Nagelkerke R2:  0.01136824
## % pres/err predicted correctly:  -1811.734
## % of predictable range [ (model-null)/(1-null) ]:  0.007834223
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.1210      -0.0747
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4937  AIC: 5528
## log likelihood:  -2468.424
## Nagelkerke R2:  0.008997405
## % pres/err predicted correctly:  -1813.461
## % of predictable range [ (model-null)/(1-null) ]:  0.006888917
## ****************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##     -0.1669
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4056 Residual
## Null Deviance:       4963
## Residual Deviance: 4963  AIC: 5557
## log likelihood:  -2481.346
## Nagelkerke R2:  -3.146332e-16
## % pres/err predicted correctly:  -1826.048
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##    -0.12513     -0.03281
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4961  AIC: 5559
```

```
## log likelihood:  -2480.574
## Nagelkerke R2:  0.0005388407
## % pres/err predicted correctly:  -1824.831
## % of predictable range [ (model-null)/(1-null) ]:  0.0006658744
## ***************************
```

```
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 ~ stimlen	5463.507	0.00000	1	1	0.0313710	0.2246165	NA	NA	NA	NA	-0.1801101
preserved ~ CumErr	5508.397	44.89017	0	0	0.0126685	0.0260191	NA	-0.1222723	NA	NA	NA
preserved ~ (I(pos^2) + pos)	5521.569	58.06225	0	0	0.0113682	0.3574333	NA	NA	0.0171109	-0.2225215	NA
preserved ~ pos	5528.487	64.98036	0	0	0.0089970	0.1209975	NA	NA	NA	-0.0746982	NA
preserved ~ 1	5557.103	93.59576	0	0	0.0000000	-0.1669442	NA	NA	NA	NA	NA
preserved ~ CumPres	5559.274	95.76750	0	0	0.0005388	-0.1251316	-0.0328123	NA	NA	NA	NA

```
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	0.3520500	524
O	0.4087118	1876
P	0.1551724	29
S	0.3850031	231
V	0.5833328	1397

```
#  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:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       1.260       -0.181
##
## Degrees of Freedom: 3796 Total (i.e. Null);  3795 Residual
## Null Deviance:       4647
## Residual Deviance: 4561  AIC: 5127
## log likelihood:  -2280.268
## Nagelkerke R2:  0.03174888
## % pres/err predicted correctly:  -1676.034
## % of predictable range [ (model-null)/(1-null) ]:  0.01994154
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##     0.07442     -0.14858
##
## Degrees of Freedom: 3796 Total (i.e. Null);  3795 Residual
## Null Deviance:       4647
## Residual Deviance: 4601  AIC: 5155
## log likelihood:  -2300.328
## Nagelkerke R2:  0.01703759
## % pres/err predicted correctly:  -1690.8
## % of predictable range [ (model-null)/(1-null) ]:  0.01131229
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.50593      0.02276     -0.28192
##
## Degrees of Freedom: 3796 Total (i.e. Null);  3794 Residual
## Null Deviance:       4647
## Residual Deviance: 4604  AIC: 5168
## log likelihood:  -2301.791
## Nagelkerke R2:  0.01595901
## % pres/err predicted correctly:  -1692.444
## % of predictable range [ (model-null)/(1-null) ]:  0.0103513
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     0.19077     -0.08523
##
## Degrees of Freedom: 3796 Total (i.e. Null);  3795 Residual
## Null Deviance:       4647
## Residual Deviance: 4615  AIC: 5180
## log likelihood:  -2307.513
## Nagelkerke R2:  0.01173039
## % pres/err predicted correctly:  -1695.941
## % of predictable range [ (model-null)/(1-null) ]:  0.008307547
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##     -0.1397
##
## Degrees of Freedom: 3796 Total (i.e. Null);  3796 Residual
## Null Deviance:       4647
## Residual Deviance: 4647  AIC: 5215
## log likelihood:  -2323.299
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1710.157
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##    -0.07459     -0.05315
##
## Degrees of Freedom: 3796 Total (i.e. Null);  3795 Residual
## Null Deviance:       4647
## Residual Deviance: 4643  AIC: 5216
## log likelihood:  -2321.493
## Nagelkerke R2:  0.001346771
## % pres/err predicted correctly:  -1707.64
## % of predictable range [ (model-null)/(1-null) ]:  0.00147087
## ***************************
```

```
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 ~ stimlen	5127.010	0.00000	1e+00	0.9999991	0.0317489	0.2595040	NA	NA	NA	NA	-0.1809759
preserved ~ CumErr	5154.794	27.78393	9e-07	0.0000009	0.0170376	0.0744184	NA	-0.1485833	NA	NA	NA
preserved ~ (I(pos^2) + pos)	5167.927	40.91674	0e+00	0.0000000	0.0159590	0.5059305	NA	NA	0.0227556	-0.2819219	NA
preserved ~ pos	5180.325	53.31505	0e+00	0.0000000	0.0117309	0.1907737	NA	NA	NA	-0.0852347	NA
preserved ~ 1	5215.405	88.39460	0e+00	0.0000000	0.0000000	-0.1396700	NA	NA	NA	NA	NA
preserved ~ CumPres	5216.342	89.33175	0e+00	0.0000000	0.0013468	-0.0745915	-0.0531458	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:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       1.269       -0.173
##
## Degrees of Freedom: 3272 Total (i.e. Null);  3271 Residual
## Null Deviance:       4004
## Residual Deviance: 3935  AIC: 4451
## log likelihood:  -1967.63
## Nagelkerke R2:  0.02946294
## % pres/err predicted correctly:  -1448.572
## % of predictable range [ (model-null)/(1-null) ]:  0.01682888
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      0.1222      -0.1612
##
## Degrees of Freedom: 3272 Total (i.e. Null);  3271 Residual
## Null Deviance:       4004
## Residual Deviance: 3969  AIC: 4476
## log likelihood:  -1984.69
## Nagelkerke R2:  0.01492366
## % pres/err predicted correctly:  -1460.455
## % of predictable range [ (model-null)/(1-null) ]:  0.008769305
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      0.5208       0.0178      -0.2435
##
## Degrees of Freedom: 3272 Total (i.e. Null);  3270 Residual
## Null Deviance:       4004
## Residual Deviance: 3965  AIC: 4480
## log likelihood:  -1982.481
## Nagelkerke R2:  0.01681467
## % pres/err predicted correctly:  -1459.101
## % of predictable range [ (model-null)/(1-null) ]:  0.00968762
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     0.28231     -0.09045
##
## Degrees of Freedom: 3272 Total (i.e. Null);  3271 Residual
## Null Deviance:       4004
## Residual Deviance: 3971  AIC: 4486
## log likelihood:  -1985.71
## Nagelkerke R2:  0.01404961
## % pres/err predicted correctly:  -1460.958
## % of predictable range [ (model-null)/(1-null) ]:  0.008427786
## ***************************
## model index:  1
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     0.06202     -0.12030
##
## Degrees of Freedom: 3272 Total (i.e. Null);  3271 Residual
## Null Deviance:       4004
## Residual Deviance: 3991  AIC: 4515
## log likelihood:  -1995.395
## Nagelkerke R2:  0.005722579
## % pres/err predicted correctly:  -1467.472
## % of predictable range [ (model-null)/(1-null) ]:  0.004009547
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##    -0.06705
##
## Degrees of Freedom: 3272 Total (i.e. Null);  3272 Residual
## Null Deviance:       4004
## Residual Deviance: 4004  AIC: 4522
## log likelihood:  -2002.018
## Nagelkerke R2:  -3.146185e-16
## % pres/err predicted correctly:  -1473.384
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

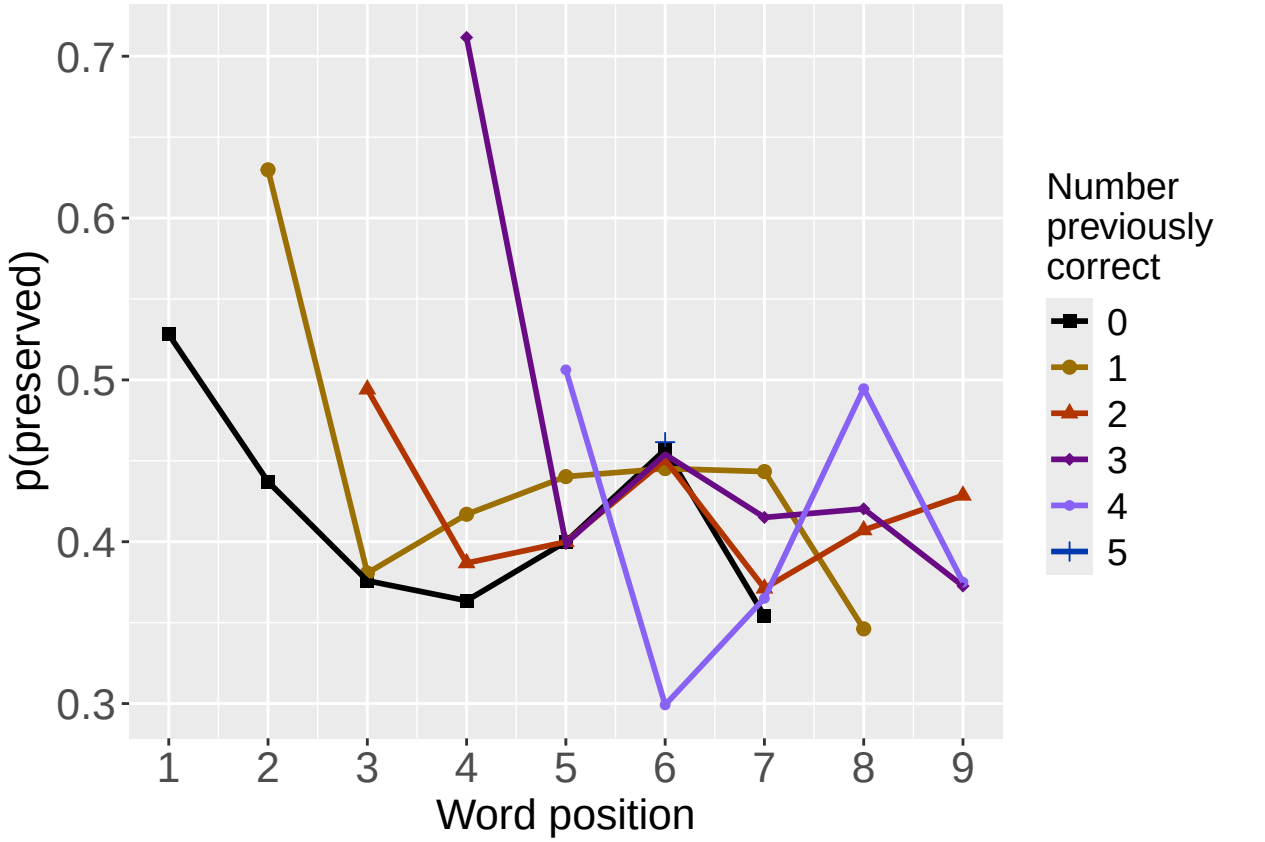
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ stimlen	4451.020	0.00000	1.0e+00	0.9999953	0.0294629	-0.2685605	NA	NA	NA	NA	-0.1730245
preserved ~ CumErr	4475.806	24.78576	4.1e-06	0.0000040	0.0149237	-0.1222491	NA	-0.1611862	NA	NA	NA
preserved ~ (I(pos^2) + pos)	4480.054	29.03410	5.0e-07	0.0000005	0.0168147	-0.5207502	NA	NA	0.0178045	-0.2435286	NA
preserved ~ pos	4486.396	35.37054	0.0e+00	0.0000000	0.0140496	-0.2823054	NA	NA	NA	-0.0904528	NA
preserved ~ CumPres	4515.325	64.30484	0.0e+00	0.0000000	0.0057226	-0.0620198	-0.1202971	NA	NA	NA	NA
preserved ~ 1	4522.436	71.41565	0.0e+00	0.0000000	0.0000000	-0.0670473	NA	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.
```

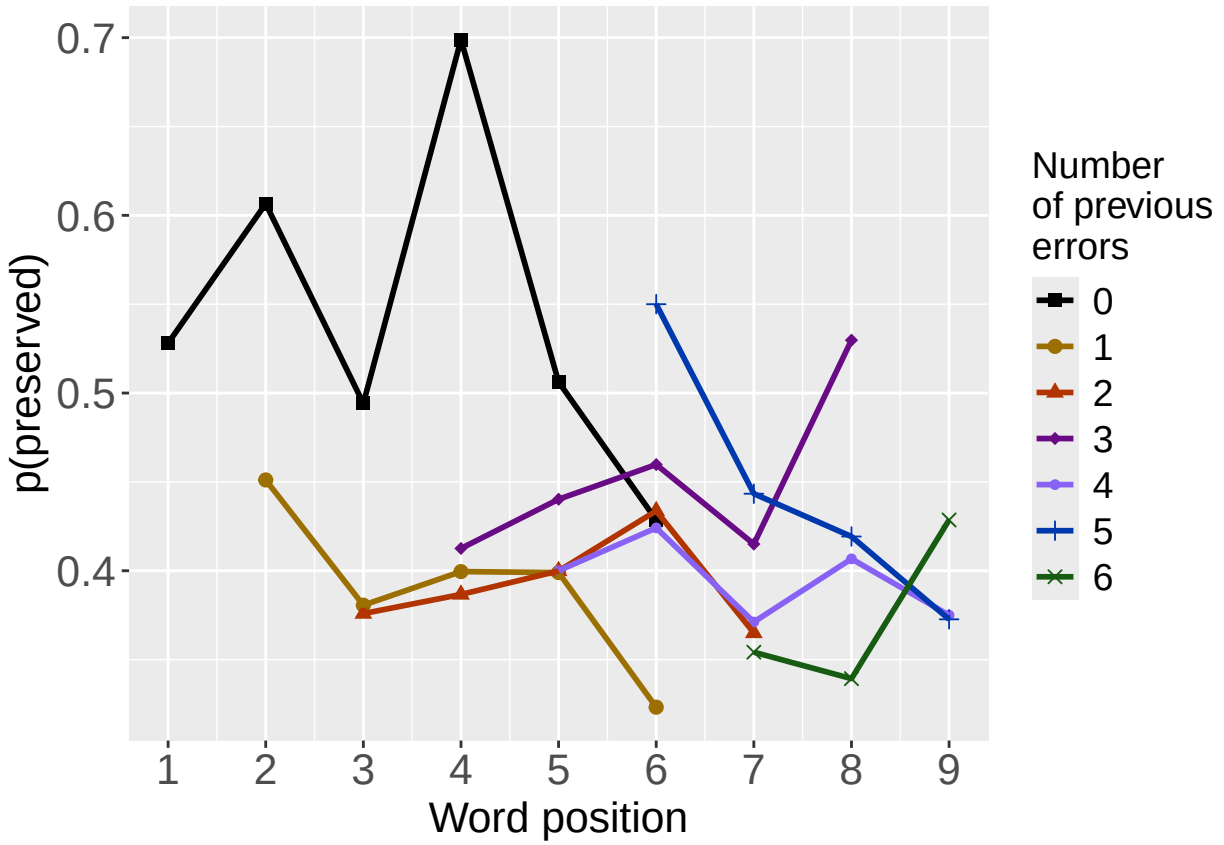
```
print(PrevCorPlot)
```




```
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
```

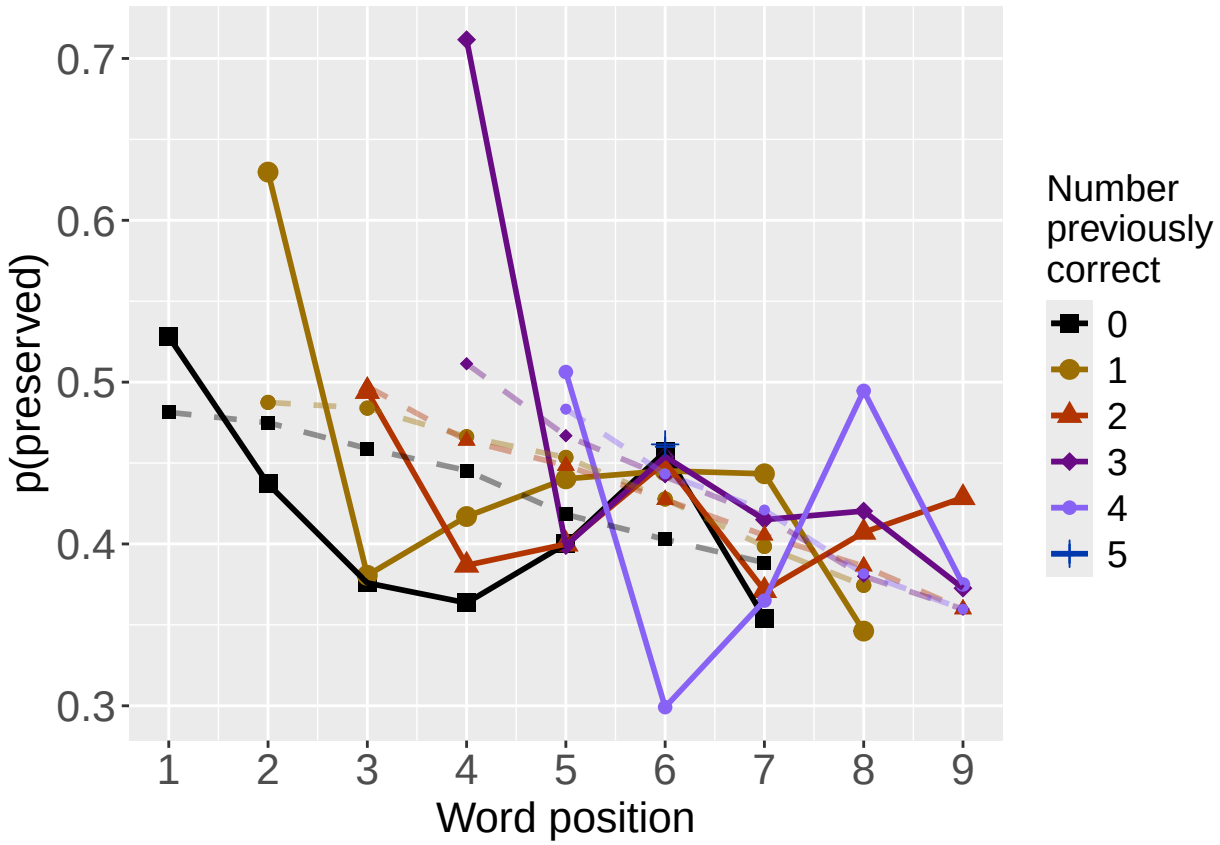
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

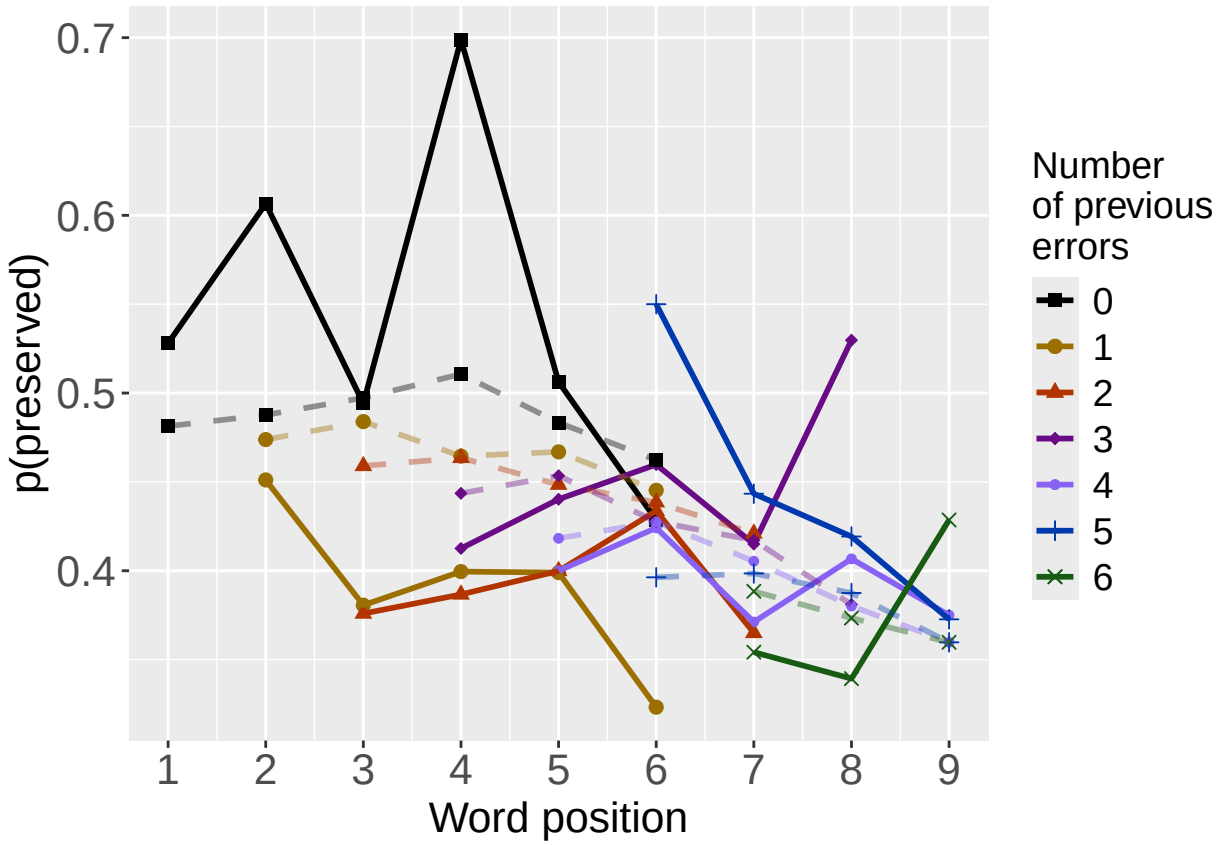
```
print(PrevCorPlot)
```




```
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
```

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.70199     -0.18321      0.02725     -0.25536
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4853  AIC: 5444
## log likelihood:  -2426.53
## Nagelkerke R2:  0.03777839
## % pres/err predicted correctly:  -1782.233
## % of predictable range [ (model-null)/(1-null) ]:  0.02398138
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     0.35743      0.01711     -0.22252
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4930  AIC: 5522
## log likelihood:  -2465.006
## Nagelkerke R2:  0.01136824
## % pres/err predicted correctly:  -1811.734
## % of predictable range [ (model-null)/(1-null) ]:  0.007834223
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	I(pos^2)	pos
preserved ~ stimlen + I(pos^2) + pos	5444.036	0.00000	1.00e+00	0.9999408	0.0377784	1.7019856	-0.1832095	0.0272494	-0.2553557

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	I(posˆ2)	pos
preserved ~ stimlen	5463.507	19.47057	5.92e-05	0.0000592	0.0313710	1.2246165	-0.1801101	NA	NA
preserved ~ I(posˆ2) + pos	5521.569	77.53283	0.00e+00	0.0000000	0.0113682	0.3574333	NA	0.0171109	-0.2225215

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

```
########
# 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)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)      stimlen      CumPres
##     1.22166     -0.18094      0.00731
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5465
## log likelihood:  -2435.895
## Nagelkerke R2:  0.03139596
## % pres/err predicted correctly:  -1789.352
## % of predictable range [ (model-null)/(1-null) ]:  0.02008465
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##    -0.12513     -0.03281
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4961  AIC: 5559
## log likelihood:  -2480.574
## Nagelkerke R2:  0.0005388407
## % pres/err predicted correctly:  -1824.831
## % of predictable range [ (model-null)/(1-null) ]:  0.0006658744
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	CumPres
preserved ~ stimlen	5463.507	0.000000	1.0000000	0.6812127	0.0313710	1.2246165	-0.1801101	NA
preserved ~ stimlen + CumPres	5465.025	1.518701	0.4679702	0.3187873	0.0313960	1.2216570	-0.1809350	0.0073105
preserved ~ CumPres	5559.274	95.767504	0.0000000	0.0000000	0.0005388	-0.1251316	NA	-0.0328123

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"pos"

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

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     1.22534     -0.16811     -0.02424
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4870  AIC: 5462
## log likelihood:  -2434.803
## Nagelkerke R2:  0.03214204
## % pres/err predicted correctly:  -1788.188
## % of predictable range [ (model-null)/(1-null) ]:  0.02072183
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      0.1210      -0.0747
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4937  AIC: 5528
## log likelihood:  -2468.424
## Nagelkerke R2:  0.008997405
## % pres/err predicted correctly:  -1813.461
## % of predictable range [ (model-null)/(1-null) ]:  0.006888917
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos
preserved ~ stimlen + pos	5462.025	0.000000	1.0000000	0.6771413	0.0321420	1.2253391	-0.1681075	-0.0242365
preserved ~ stimlen	5463.507	1.481331	0.4767965	0.3228587	0.0313710	1.2246165	-0.1801101	NA
preserved ~ pos	5528.487	66.461695	0.0000000	0.0000000	0.0089974	0.1209975	NA	-0.0746982

```
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)	stimlen	I(pos^2)	pos	CumPres
preserved ~ stimlen + I(pos^2) + pos	5444.036	0.000000	1.0000000	0.9999408	0.0377784	1.7019856	-0.1832095	0.0272494	-0.2553557	NA
preserved ~ stimlen + pos	5462.025	0.000000	1.0000000	0.6771413	0.0321420	1.2253391	-0.1681075	NA	-0.0242365	NA
preserved ~ stimlen	5463.507	19.470570	0.0000592	0.0000592	0.0313710	0.2246165	-0.1801101	NA	NA	NA
preserved ~ stimlen	5463.507	0.000000	1.0000000	0.6812127	0.0313710	0.2246165	-0.1801101	NA	NA	NA
preserved ~ stimlen	5463.507	1.481331	0.4767965	0.3228587	0.0313710	0.2246165	-0.1801101	NA	NA	NA
preserved ~ stimlen + CumPres	5465.025	1.518701	0.4679702	0.3187870	0.0313960	0.2216570	-0.1809350	NA	NA	0.0073105
preserved ~ I(pos^2) + pos	5521.569	77.532827	0.0000000	0.0000000	0.0113682	2.3574333	NA	0.0171109	-0.2225215	NA
preserved ~ pos	5528.487	66.461695	0.0000000	0.0000000	0.0089974	0.1209975	NA	NA	-0.0746982	NA
preserved ~ CumPres	5559.274	95.767504	0.0000000	0.0000000	0.0005388	-0.1251316	NA	NA	NA	-0.0328123

```
# 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!
```

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     1.58846     -0.16872      0.02715     -0.25458      0.04583
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4052 Residual
## Null Deviance:       4963
## Residual Deviance: 4846  AIC: 5439
## log likelihood:  -2423.085
## Nagelkerke R2:  0.04011817
## % pres/err predicted correctly:  -1778.961
## % of predictable range [ (model-null)/(1-null) ]:  0.02577204
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     1.70199     -0.18321      0.02725     -0.25536
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4853  AIC: 5444
## log likelihood:  -2426.53
## Nagelkerke R2:  0.03777839
## % pres/err predicted correctly:  -1782.233
## % of predictable range [ (model-null)/(1-null) ]:  0.02398138
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##     -0.1669
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4056 Residual
## Null Deviance:       4963
## Residual Deviance: 4963  AIC: 5557
## log likelihood:  -2481.346
## Nagelkerke R2:  -3.146332e-16
## % pres/err predicted correctly:  -1826.048
## % 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)	stimlen	I(pos^2)	pos	log_freq
preserved ~ stimlen + I(pos^2) + pos + log_freq	5438.949	0.000000	1.000000	0.9271299	0.0401182	5.2588464	-0.1687182	0.0271509	-0.2545782	0.0458332
preserved ~ stimlen + I(pos^2) + pos	5444.036	5.086816	0.078598	0.0728706	0.0377784	7.4701985	-0.1832095	0.0272494	-0.2553557	NA
preserved ~ 1	5557.103	118.153152	0.000000	0.0000000	0.0000000	-0.1669442	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
```

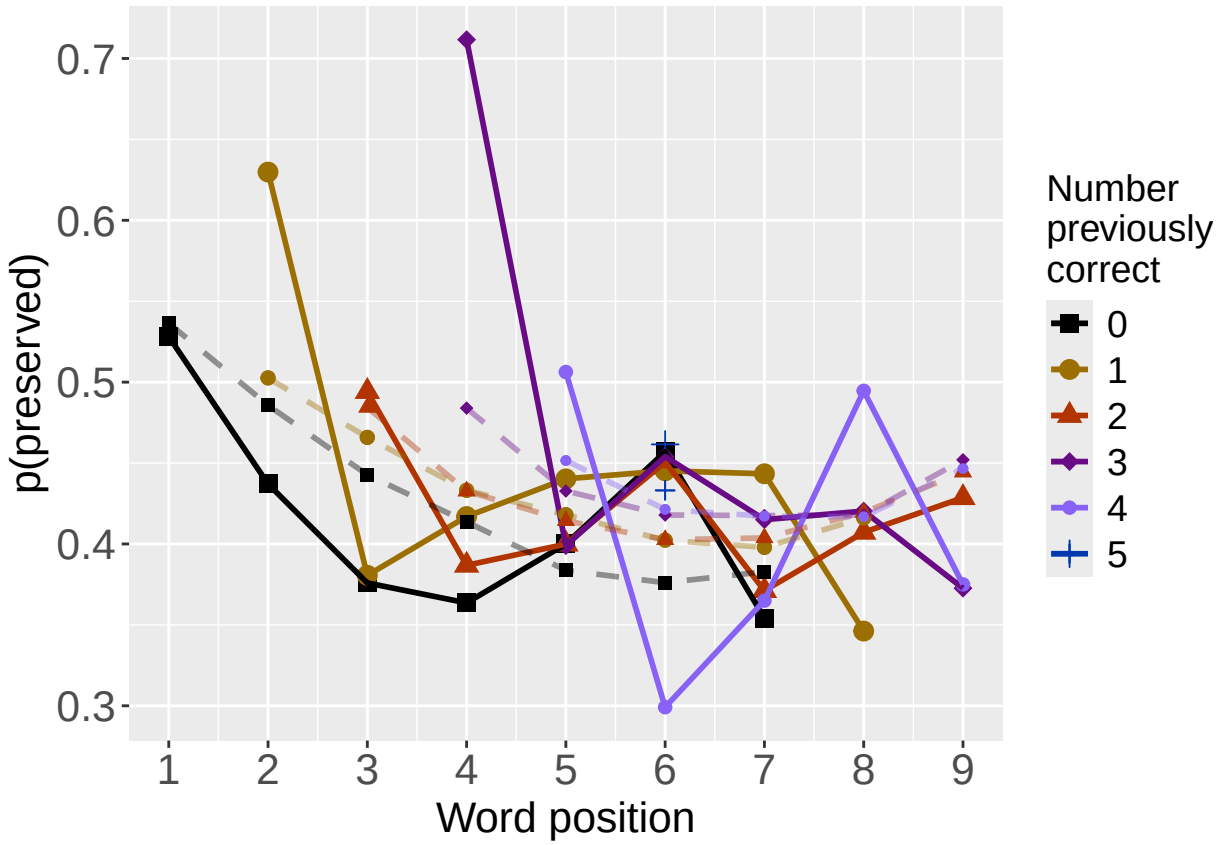
```
## Single term deletions
##
## Model:
## preserved ~ stimlen + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## stimlen   1   4907.1 5497.9
## pos       1   4864.7 5455.5
## I(pos^2)  1   4862.6 5453.4
## log_freq  1   4853.1 5443.8
## <none>        4846.2 5438.9
```

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

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

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

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

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

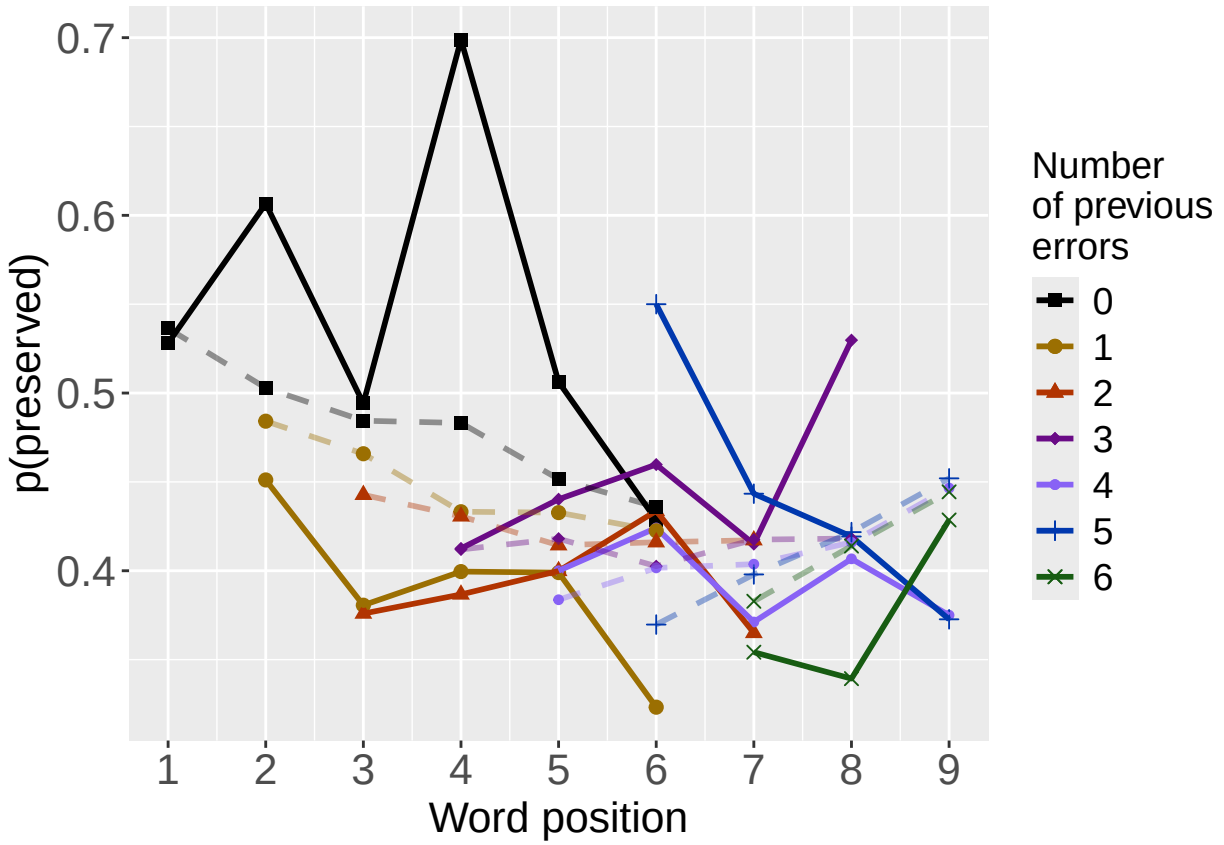
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

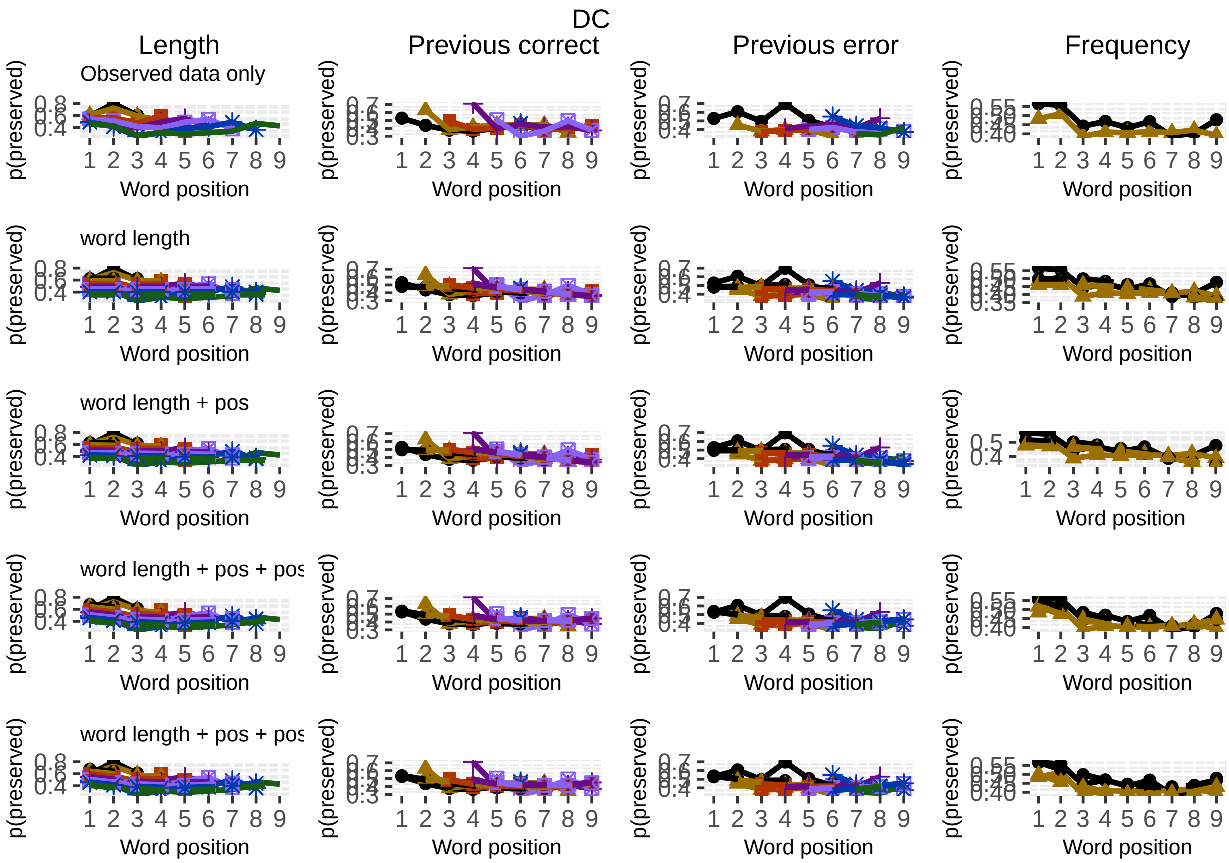
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      1.2246      -0.1801
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4055 Residual
## Null Deviance:       4963
## Residual Deviance: 4872  AIC: 5464
## log likelihood:  -2435.932
```

```
## Nagelkerke R2:  0.03137098
## % pres/err predicted correctly:  -1789.353
## % of predictable range [ (model-null)/(1-null) ]:  0.0200839
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     1.22534     -0.16811     -0.02424
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4054 Residual
## Null Deviance:       4963
## Residual Deviance: 4870  AIC: 5462
## log likelihood:  -2434.803
## Nagelkerke R2:  0.03214204
## % pres/err predicted correctly:  -1788.188
## % of predictable range [ (model-null)/(1-null) ]:  0.02072183
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     I(pos^2)
##     1.70199     -0.18321     -0.25536      0.02725
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4053 Residual
## Null Deviance:       4963
## Residual Deviance: 4853  AIC: 5444
## log likelihood:  -2426.53
## Nagelkerke R2:  0.03777839
## % pres/err predicted correctly:  -1782.233
## % of predictable range [ (model-null)/(1-null) ]:  0.02398138
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     I(pos^2)     log_freq
##     1.58846     -0.16872     -0.25458      0.02715      0.04583
##
## Degrees of Freedom: 4056 Total (i.e. Null);  4052 Residual
## Null Deviance:       4963
## Residual Deviance: 4846  AIC: 5439
## log likelihood:  -2423.085
## Nagelkerke R2:  0.04011817
## % pres/err predicted correctly:  -1778.961
## % of predictable range [ (model-null)/(1-null) ]:  0.02577204
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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




```r
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)
```

	stimlen	I(pos^2)	pos	log_freq
McFadden	0.0132701	0.0026833	0.0037965	0.0029596
SquaredCorrelation	0.0178687	0.0036175	0.0051228	0.0040077
Nagelkerke	0.0178687	0.0036175	0.0051228	0.0040077
Estrella	0.0180878	0.0036585	0.0051776	0.0040402

```
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
## stimlen + pos + I(pos^2) + log_freq stimlen + pos + I(pos^2) + log_freq 4846.171
## stimlen + pos + I(pos^2)                      stimlen + pos + I(pos^2) 4853.059
## stimlen + pos                                          stimlen + pos 4869.605
## stimlen                                                      stimlen 4871.863
## null                                                            null 4962.692
##                                    deviance_explained percent_explained
## stimlen + pos + I(pos^2) + log_freq          116.52082          2.347936
## stimlen + pos + I(pos^2)                     109.63241          2.209132
## stimlen + pos                                 93.08656          1.875727
## stimlen                                       90.82830          1.830223
## null                                           0.00000          0.000000
##                                    percent_of_explained_deviance increment_in_explained
## stimlen + pos + I(pos^2) + log_freq                     100.00000               5.911742
## stimlen + pos + I(pos^2)                                 94.08826              14.199904
## stimlen + pos                                            79.88835               1.938079
## stimlen                                                  77.95028              77.950276
## null                                                           NA               0.000000
```

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

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

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

	deviance	deviance_explained
stimlen + pos + I(pos^2) + log_freq	4846.171	116.52082
stimlen + pos + I(pos^2)	4853.059	109.63241
stimlen + pos	4869.605	93.08656
stimlen	4871.863	90.82830
null	4962.692	0.00000

	percent_explained	percent_of_explained_deviance	increment_in_explained
stimlen + pos + I(pos^2) + log_freq	2.347936	100.00000	5.911741
stimlen + pos + I(pos^2)	2.209132	94.08826	14.199904
stimlen + pos	1.875727	79.88835	1.938079
stimlen	1.830223	77.95028	77.950276
null	0.000000	NA	0.000000

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

```
##           Nagelkerke
## stimlen    0.5836263
## I(pos^2)   0.1181539
## pos        0.1673195
## log_freq   0.1309003
```

```
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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## 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 (cumerr_residuals^2) * (N_values$cumerr_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

## Warning in (null_cumcor_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_stimlen
## 1                       preserved ~ stimlen       0.3982892       4871.863  0.000000000
## 2                   preserved ~ stimlen+pos       0.4037909       4869.605  0.005501782
## 3          preserved ~ stimlen+pos+I(pos^2)       0.4858061       4853.059  0.087516970
## 4 preserved ~ stimlen+pos+I(pos^2)+log_freq       0.4871227       4846.171  0.088833533
##   diff_stimlen+pos diff_stimlen+pos+I(pos^2) diff_stimlen+pos+I(pos^2)+log_freq
## 1     -0.005501782              -0.087516970                       -0.088833533
## 2      0.000000000              -0.082015188                       -0.083331751
## 3      0.082015188               0.000000000                       -0.001316563
## 4      0.083331751               0.001316563                        0.000000000
```

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

model	p_accounted_for	model_deviance
preserved ~ stimlen	0.3982892	4871.863
preserved ~ stimlen+pos	0.4037909	4869.605
preserved ~ stimlen+pos+I(pos^2)	0.4858061	4853.059
preserved ~ stimlen+pos+I(pos^2)+log_freq	0.4871227	4846.171

model	diff_stimlen	diff_stimlen+pos	diff_stimlen+pos+I(pos^2)
preserved ~ stimlen	0.0000000	-0.0055018	-0.0875170
preserved ~ stimlen+pos	0.0055018	0.0000000	-0.0820152
preserved ~ stimlen+pos+I(pos^2)	0.0875170	0.0820152	0.0000000
preserved ~ stimlen+pos+I(pos^2)+log_freq	0.0888335	0.0833318	0.0013166

```
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")
```