VS - 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	511	33	123	NA	NA	667
2	62	NA	404	94	107	667
3	295	NA	163	193	16	667
4	280	NA	227	67	36	610
5	225	NA	195	66	35	521
6	190	1	130	65	22	408
7	165	NA	95	27	18	305
8	84	NA	48	23	4	159
9	66	NA	2	NA	7	75

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7661169	0.0494753	0.1844078	NA	NA	667
2	0.0929535	NA	0.6056972	0.1409295	0.1604198	667
3	0.4422789	NA	0.2443778	0.2893553	0.0239880	667
4	0.4590164	NA	0.3721311	0.1098361	0.0590164	610
5	0.4318618	NA	0.3742802	0.1266795	0.0671785	521
6	0.4656863	0.0024510	0.3186275	0.1593137	0.0539216	408

pos_factor	O	P	V	1	S	total
7	0.5409836	NA	0.3114754	0.0885246	0.0590164	305
8	0.5283019	NA	0.3018868	0.1446541	0.0251572	159
9	0.8800000	NA	0.0266667	NA	0.0933333	75

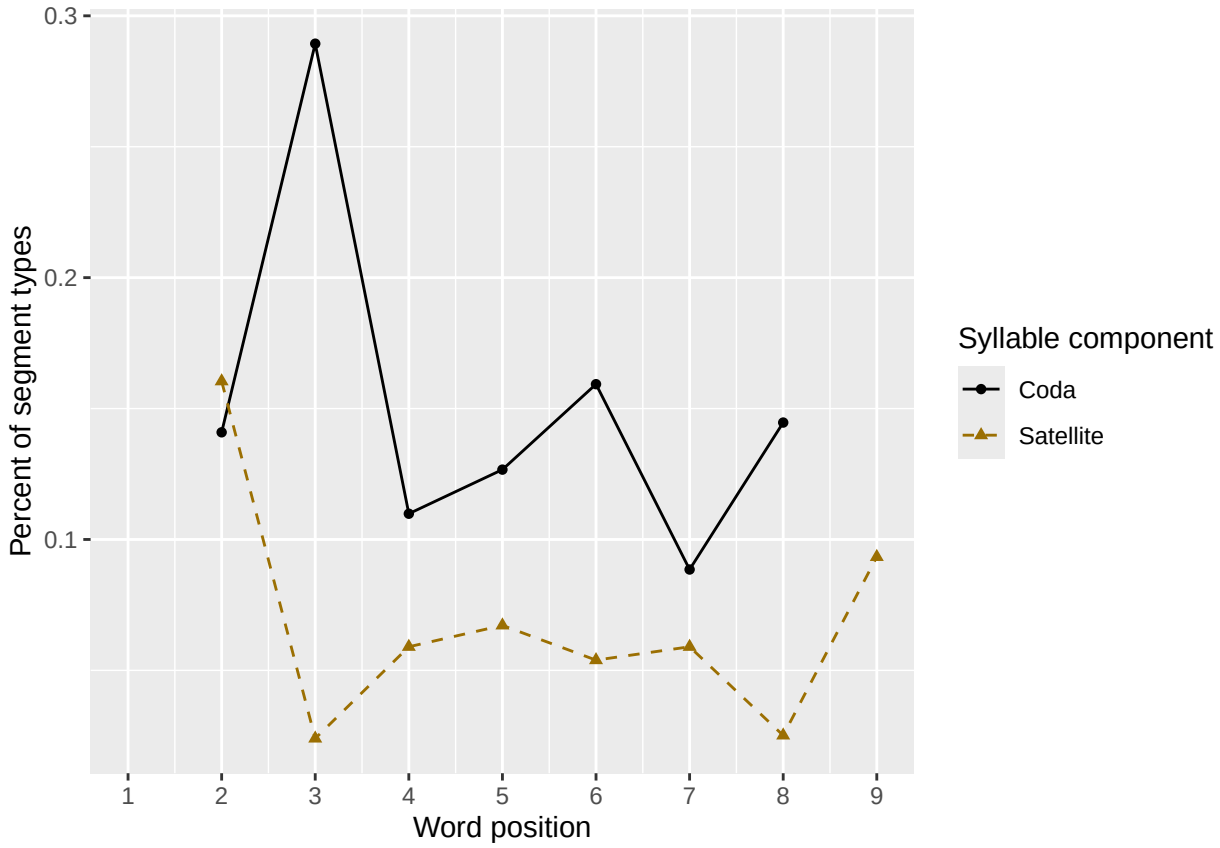
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 0.918 0.953 0.813 NA     NA     NA     NA     NA     NA
## 2       5 0.942 0.845 0.794  0.807 NA     NA     NA     NA     NA
## 3       6 0.938 0.827 0.768  0.664  0.652 NA     NA     NA     NA
## 4       7 0.908 0.862 0.825  0.741  0.680  0.582 NA     NA     NA
## 5       8 0.919 0.844 0.796  0.684  0.622  0.583  0.524 NA     NA
## 6       9 0.885 0.849 0.765  0.652  0.728  0.552  0.420  0.406 NA
## 7      10 0.929 0.913 0.764  0.600  0.496  0.450  0.375  0.451  0.405
```

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

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

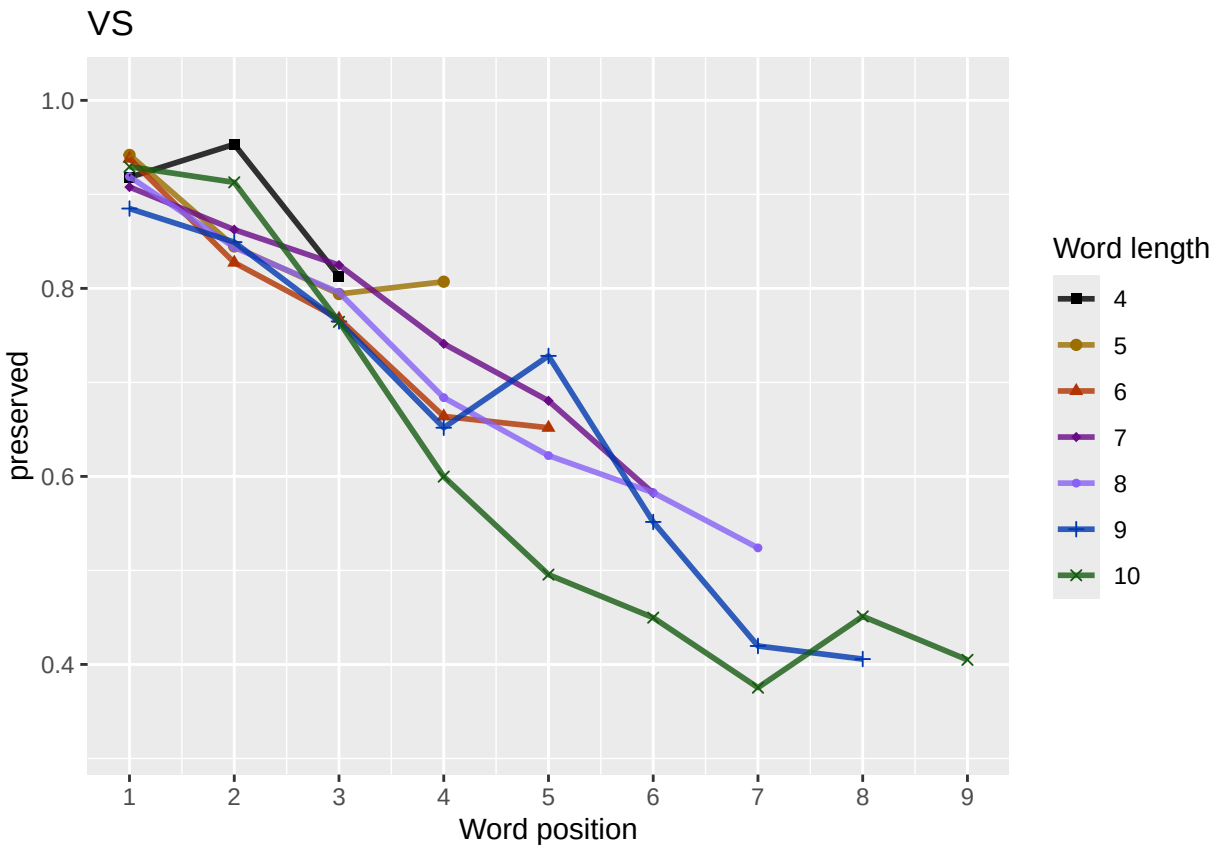
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1       4    57    57    57    NA    NA    NA    NA    NA    NA
## 2       5    89    89    89    89    NA    NA    NA    NA    NA
## 3       6   113   113   113   113   113    NA    NA    NA    NA
## 4       7   103   103   103   103   103   103    NA    NA    NA
## 5       8   146   146   146   146   146   146   146    NA    NA
## 6       9    84    84    84    84    84    84    84    84    NA
## 7      10    75    75    75    75    75    75    75    75    75
```

```
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:  8
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.9979762         0.0201385         0.0501313        -0.5508944        -0.0002682
##      stimlen:pos
##       -0.0281893
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3942  AIC: 4476
## log likelihood:  -1970.913
## Nagelkerke R2:  0.1819909
## % pres/err predicted correctly:  -1358.553
## % of predictable range [ (model-null)/(1-null) ]:  0.1359885
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.65303     -0.08292      0.03579     -0.68134
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3945  AIC: 4477
## log likelihood:  -1972.693
## Nagelkerke R2:  0.1808382
## % pres/err predicted correctly:  -1359.71
## % of predictable range [ (model-null)/(1-null) ]:  0.135253
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.04950      0.03132     -0.66787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3955  AIC: 4490
## log likelihood:  -1977.724
## Nagelkerke R2:  0.1775739
## % pres/err predicted correctly:  -1364.781
## % of predictable range [ (model-null)/(1-null) ]:  0.1320303
## **************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.91215     -0.06262     -0.35791
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3965  AIC: 4499
## log likelihood:  -1982.333
## Nagelkerke R2:  0.1745771
## % pres/err predicted correctly:  -1365.421
## % of predictable range [ (model-null)/(1-null) ]:  0.131624
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.34194     -0.11464     -0.47927      0.01428
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3963  AIC: 4500
## log likelihood:  -1981.683
## Nagelkerke R2:  0.1750003
## % pres/err predicted correctly:  -1365.432
## % of predictable range [ (model-null)/(1-null) ]:  0.1316167
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.5136      -0.3787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3971  AIC: 4506
## log likelihood:  -1985.342
## Nagelkerke R2:  0.1726169
## % pres/err predicted correctly:  -1368.724
## % of predictable range [ (model-null)/(1-null) ]:  0.1295248
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen  
##      2.8679      -0.2472  
## 
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469 
## Residual Deviance: 4336  AIC: 4880
## log likelihood:  -2168.217
## Nagelkerke R2:  0.0478757
## % pres/err predicted correctly:  -1515.996
## % of predictable range [ (model-null)/(1-null) ]:  0.03593143
## ***************************
## model index:  1
## 
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial", 
##     data = PosDat)
## 
## Coefficients:
## (Intercept)  
##      0.9411  
## 
## Degrees of Freedom: 4078 Total (i.e. Null);  4078 Residual
## Null Deviance:       4469 
## Residual Deviance: 4469  AIC: 5024
## log likelihood:  -2234.269
## Nagelkerke R2:  -3.335871e-16
## % pres/err predicted correctly:  -1572.536
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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)	4475.787	0.000000	1.0000000	0.6903329	0.0518199	2.9979762	0.0201385	-0.5508944	-0.0281893	0.0501313	-0.0002682
preserved ~ stimlen + I(pos^2) + pos	4477.395	1.608030	0.4475286	0.3089437	0.0518083	2.6530285	-0.0829239	-0.6813407	NA	0.0357902	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	4489.541	13.75387	0.001031	0.000711	0.1775739	0.90495029	NA	-0.6678680	NA	0.0313213	NA
preserved ~ stimlen + pos	4498.748	22.95904	0.000010	0.000007	0.1745772	1.19121493	-0.0626199	-0.3579116	NA	NA	NA
preserved ~ stimlen * pos	4499.872	24.08567	0.000005	0.000004	0.1750033	1.33419449	-0.1146356	-0.4792721	0.0142837	NA	NA
preserved ~ pos	4505.562	29.78120	0.000000	0.000000	0.1726169	0.95136215	NA	-0.3787469	NA	NA	NA
preserved ~ stimlen	4879.791	404.00370	0.000000	0.000000	0.0047872	0.78678648	-0.2472446	NA	NA	NA	NA
preserved ~ 1	5023.545	547.75788	0.000000	0.000000	0.0000000	0.09410632	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  stimlen:I(pos^2)
##        2.9979762         0.0201385         0.0501313        -0.5508944        -0.0002682
##      stimlen:pos
##       -0.0281893
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3942  AIC: 4476
```

```
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
```

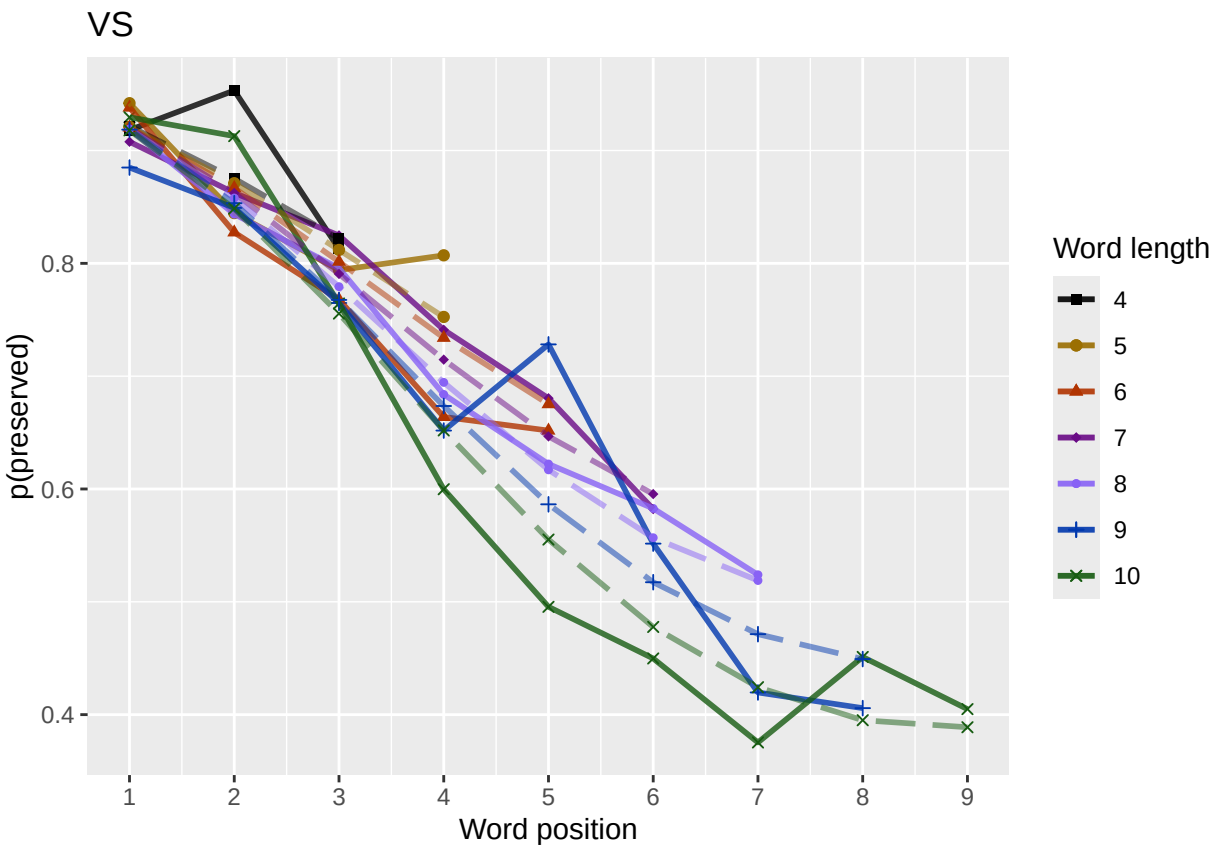
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.922 0.875 0.822 NA     NA     NA     NA     NA     NA
## 2       5 0.921 0.871 0.812  0.752 NA     NA     NA     NA     NA
## 3       6 0.920 0.867 0.801  0.734  0.675 NA     NA     NA     NA
## 4       7 0.920 0.862 0.791  0.715  0.647  0.596 NA     NA     NA
```

```
## 5       8 0.919 0.858 0.779  0.694  0.617  0.557  0.519 NA     NA
## 6       9 0.919 0.853 0.768  0.674  0.586  0.517  0.471  0.449 NA
## 7      10 0.918 0.849 0.755  0.652  0.555  0.478  0.424  0.395  0.389
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       75   667
```

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

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

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

```
## [1] "The number of responses with fragments was 75 / 667 = 11.24 percent"
```

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

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

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

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

```
## ****************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.8812180         0.0351241         0.0604225        -0.4934166        -0.0003778
##      stimlen:pos
##       -0.0377970
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3900 Residual
## Null Deviance:       4040
## Residual Deviance: 3666  AIC: 4204
## log likelihood:  -1833.067
## Nagelkerke R2:  0.1415276
## % pres/err predicted correctly:  -1254.829
## % of predictable range [ (model-null)/(1-null) ]:  0.1052432
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.75132     -0.10054      0.04137     -0.67297
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3902 Residual
## Null Deviance:       4040
## Residual Deviance: 3672  AIC: 4208
## log likelihood:  -1836.044
## Nagelkerke R2:  0.1393771
## % pres/err predicted correctly:  -1257.66
## % of predictable range [ (model-null)/(1-null) ]:  0.1032265
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.01762      0.03603     -0.65709
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3903 Residual
## Null Deviance:       4040
## Residual Deviance: 3686  AIC: 4225
## log likelihood:  -1843.069
## Nagelkerke R2:  0.1342877
## % pres/err predicted correctly:  -1264.576
## % of predictable range [ (model-null)/(1-null) ]:  0.09829839
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos
##     2.92448     -0.07814     -0.30366
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3903 Residual
## Null Deviance:       4040
## Residual Deviance: 3696  AIC: 4235
## log likelihood:  -1847.874
## Nagelkerke R2:  0.130797
## % pres/err predicted correctly:  -1265.069
## % of predictable range [ (model-null)/(1-null) ]:  0.09794749
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.27529     -0.12057     -0.40545      0.01197
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3902 Residual
## Null Deviance:       4040
## Residual Deviance: 3695  AIC: 4237
## log likelihood:  -1847.453
## Nagelkerke R2:  0.131103
## % pres/err predicted correctly:  -1265.196
## % of predictable range [ (model-null)/(1-null) ]:  0.09785667
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##       2.422       -0.329
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3904 Residual
## Null Deviance:       4040
## Residual Deviance: 3705  AIC: 4245
## log likelihood:  -1852.309
## Nagelkerke R2:  0.1275669
## % pres/err predicted correctly:  -1269.676
## % of predictable range [ (model-null)/(1-null) ]:  0.09466486
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8562      -0.2266
```

```
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3904 Residual
## Null Deviance:       4040
## Residual Deviance: 3940  AIC: 4488
## log likelihood:  -1969.848
## Nagelkerke R2:  0.03923219
## % pres/err predicted correctly:  -1361.35
## % of predictable range [ (model-null)/(1-null) ]:  0.0293486
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.096
##
## Degrees of Freedom: 3905 Total (i.e. Null);  3905 Residual
## Null Deviance:       4040
## Residual Deviance: 4040  AIC: 4598
## log likelihood:  -2019.865
## Nagelkerke R2:  -3.445209e-16
## % pres/err predicted correctly:  -1402.542
## % 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)	4203.952	0.000000	1.000000	0.9903965	0.0391527	0.6881218	0.0351241	-0.4934166	-0.6037797	0.0604225	-0.0003778
preserved ~ stimlen + I(pos^2) + pos	4208.437	4.484678	0.106209	0.0096010	0.0139377	0.7751319	-0.1005406	-0.6672969	NA	0.0413729	NA
preserved ~ I(pos^2) + pos	4225.037	21.085418	0.000026	0.0000023	0.0000000	0.9134287	NA	-0.6570876	NA	0.0360327	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	4235.486	31.53397	7.000000	1.000000	0.1307970	2.924477	-0.0781419	-0.3036638	NA	NA	NA
preserved ~ stimlen * pos	4236.867	32.91471	1.000000	1.000000	0.1311030	3.275288	-0.1205667	-0.4054540	0.0119739	NA	NA
preserved ~ pos	4245.459	41.50697	0.000000	0.000000	0.1275669	2.422432	NA	-0.3290013	NA	NA	NA
preserved ~ stimlen	4487.832	283.87922	0.000000	0.000000	0.0392322	2.856247	-0.2266477	NA	NA	NA	NA
preserved ~ 1	4597.809	393.85708	0.000000	0.000000	0.0000000	1.096451	NA	NA	NA	NA	NA

```
# plot no fragment data

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

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

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

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

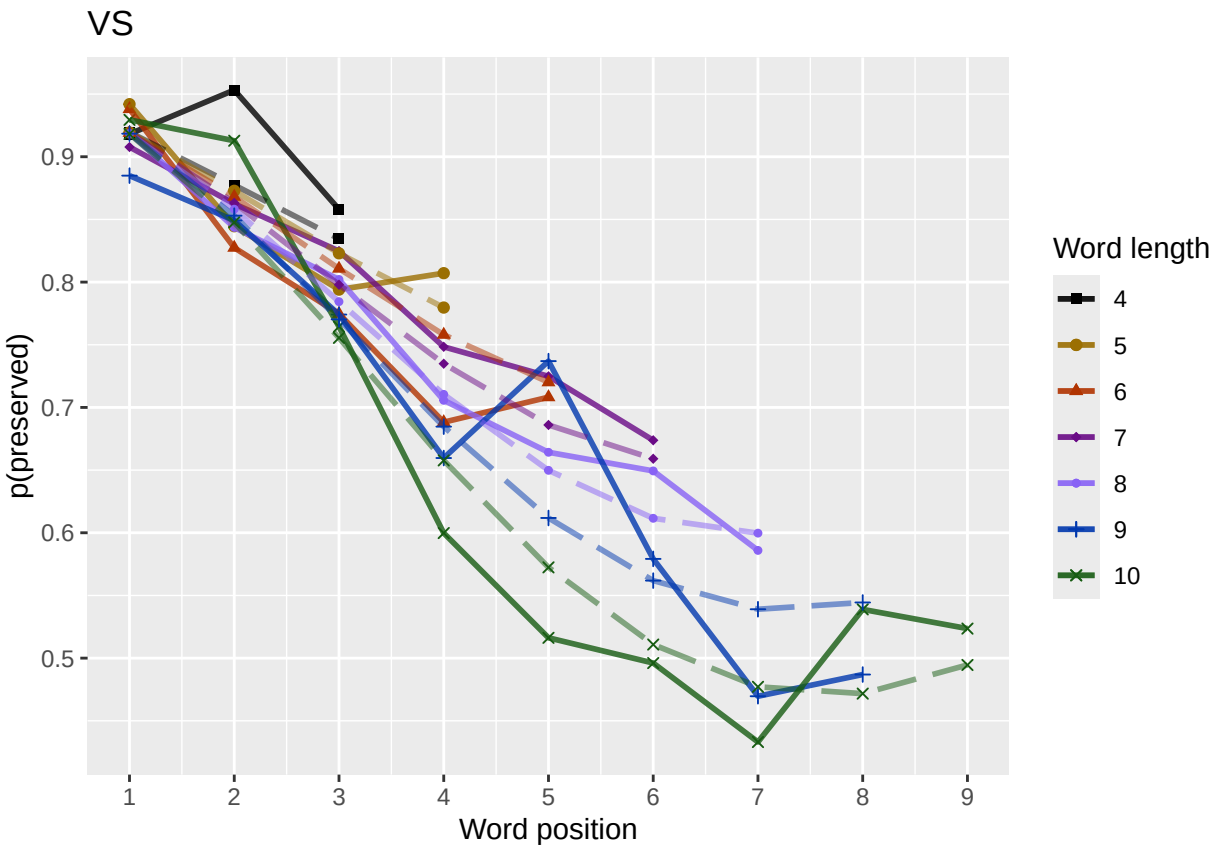
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.920 0.877 0.835 NA     NA     NA     NA     NA     NA
## 2       5 0.919 0.873 0.823  0.780 NA     NA     NA     NA     NA
## 3       6 0.919 0.868 0.811  0.758  0.720 NA     NA     NA     NA
## 4       7 0.919 0.863 0.798  0.735  0.686  0.659 NA     NA     NA
## 5       8 0.919 0.858 0.784  0.710  0.650  0.612  0.600 NA     NA
## 6       9 0.918 0.853 0.770  0.685  0.612  0.562  0.539  0.544 NA
## 7      10 0.918 0.848 0.755  0.658  0.572  0.511  0.477  0.472  0.495
```

```
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: 0.32 - 1.01"
```

```
# 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] -0.01536603
```

```
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] -0.05383219
```

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

```
if(potential_u_shape){ # downward, so potential U-shape
  # take only positive values from the table
  filtered_pos_upward_u<-table_pos_diffs
  filtered_pos_upward_u[table_pos_diffs <= 0] <- NA
  average_pos_u_diffs <- apply(filtered_pos_upward_u,
                               2,mean,na.rm=TRUE)
  OA_mean_pos_u_diff <- mean(average_pos_u_diffs,na.rm=TRUE)
```

```
  if(is.nan(OA_mean_pos_u_diff) | (OA_mean_pos_u_diff <= 0)){
    print("No U-shape in this participant")
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 0)
    potential_u_shape <- FALSE
  }else{
    results_report_DF <- AddReportLine(results_report_DF, "U-shape", 1)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         log_freq         I(pos^2)              pos
##         3.53918         -0.07311          0.24634          0.03477         -0.67208
## stimlen:log_freq
##        -0.02969
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3939  AIC: 4475
## log likelihood:  -1969.747
## Nagelkerke R2:  0.1827454
## % pres/err predicted correctly:  -1357.705
## % of predictable range [ (model-null)/(1-null) ]:  0.1365274
## ***************************
## 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)
##       2.9979762        0.0201385        0.0501313       -0.5508944        -0.0002682
##     stimlen:pos
##      -0.0281893
##
```

```
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3942  AIC: 4476
## log likelihood:  -1970.913
## Nagelkerke R2:  0.1819909
## % pres/err predicted correctly:  -1358.553
## % of predictable range [ (model-null)/(1-null) ]:  0.1359885
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.65303     -0.08292      0.03579     -0.68134
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3945  AIC: 4477
## log likelihood:  -1972.693
## Nagelkerke R2:  0.1808382
## % pres/err predicted correctly:  -1359.71
## % of predictable range [ (model-null)/(1-null) ]:  0.135253
## ****************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          3.560747          -0.074347           0.363330           0.036259          -0.682709
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.034469           0.005496          -0.047371
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4071 Residual
## Null Deviance:       4469
## Residual Deviance: 3938  AIC: 4478
## log likelihood:  -1968.918
## Nagelkerke R2:  0.1832822
## % pres/err predicted correctly:  -1357.469
## % of predictable range [ (model-null)/(1-null) ]:  0.1366774
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     3.62771     -0.07970      0.03574     -0.68087      0.01071
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4074 Residual
```

```
## Null Deviance:       4469
## Residual Deviance: 3945  AIC: 4479
## log likelihood:  -1972.553
## Nagelkerke R2:  0.1809286
## % pres/err predicted correctly:  -1359.616
## % of predictable range [ (model-null)/(1-null) ]:  0.1353131
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)               pos           log_freq
##          3.617404          -0.077660           0.035886          -0.683263           0.104733
## I(pos^2):log_freq       pos:log_freq
##          0.003487          -0.039720
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4072 Residual
## Null Deviance:       4469
## Residual Deviance: 3944  AIC: 4482
## log likelihood:  -1971.943
## Nagelkerke R2:  0.1813239
## % pres/err predicted correctly:  -1359.587
## % of predictable range [ (model-null)/(1-null) ]:  0.1353314
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.04950      0.03132     -0.66787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3955  AIC: 4490
## log likelihood:  -1977.724
## Nagelkerke R2:  0.1775739
## % pres/err predicted correctly:  -1364.781
## % of predictable range [ (model-null)/(1-null) ]:  0.1320303
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)               pos           log_freq  I(pos^2):log_freq
##          3.046266           0.031529          -0.668504           0.129395           0.003201
##      pos:log_freq
##         -0.040701
##
```

```
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3952  AIC: 4492
## log likelihood:  -1976.047
## Nagelkerke R2:  0.1786632
## % pres/err predicted correctly:  -1363.791
## % of predictable range [ (model-null)/(1-null) ]:  0.1326594
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.81224          -0.05263           0.26526          -0.35790          -0.03209
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4074 Residual
## Null Deviance:       4469
## Residual Deviance: 3958  AIC: 4495
## log likelihood:  -1978.83
## Nagelkerke R2:  0.1768556
## % pres/err predicted correctly:  -1362.838
## % of predictable range [ (model-null)/(1-null) ]:  0.1332655
## ****************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.812650         -0.052437          0.265564         -0.358354         -0.031208
##     log_freq:pos
##        -0.001647
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3958  AIC: 4497
## log likelihood:  -1978.818
## Nagelkerke R2:  0.1768634
## % pres/err predicted correctly:  -1362.862
## % of predictable range [ (model-null)/(1-null) ]:  0.1332499
## ****************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.91215     -0.06262     -0.35791
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
```

```
## Null Deviance:       4469
## Residual Deviance: 3965  AIC: 4499
## log likelihood:  -1982.333
## Nagelkerke R2:  0.1745771
## % pres/err predicted correctly:  -1365.421
## % of predictable range [ (model-null)/(1-null) ]:  0.131624
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.34194     -0.11464     -0.47927      0.01428
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3963  AIC: 4500
## log likelihood:  -1981.683
## Nagelkerke R2:  0.1750003
## % pres/err predicted correctly:  -1365.432
## % of predictable range [ (model-null)/(1-null) ]:  0.1316167
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.88545     -0.05910     -0.35791      0.01178
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3964  AIC: 4500
## log likelihood:  -1982.164
## Nagelkerke R2:  0.1746868
## % pres/err predicted correctly:  -1365.278
## % of predictable range [ (model-null)/(1-null) ]:  0.1317147
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.86862      -0.05558      -0.36135       0.06333      -0.01163
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4074 Residual
## Null Deviance:       4469
## Residual Deviance: 3963  AIC: 4501
## log likelihood:  -1981.429
```

```
## Nagelkerke R2:  0.1751655
## % pres/err predicted correctly:  -1365.026
## % of predictable range [ (model-null)/(1-null) ]:  0.1318745
## **************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.5136      -0.3787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3971  AIC: 4506
## log likelihood:  -1985.342
## Nagelkerke R2:  0.1726169
## % pres/err predicted correctly:  -1368.724
## % of predictable range [ (model-null)/(1-null) ]:  0.1295248
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.51231      -0.37928       0.08329      -0.01389
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3967  AIC: 4506
## log likelihood:  -1983.639
## Nagelkerke R2:  0.1737268
## % pres/err predicted correctly:  -1367.564
## % of predictable range [ (model-null)/(1-null) ]:  0.1302616
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.50522     -0.37650      0.02235
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3969  AIC: 4506
## log likelihood:  -1984.699
## Nagelkerke R2:  0.1730357
## % pres/err predicted correctly:  -1368.086
## % of predictable range [ (model-null)/(1-null) ]:  0.1299301
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.77285          -0.23775           0.25535          -0.03063
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 4329  AIC: 4876
## log likelihood:  -2164.711
## Nagelkerke R2:  0.05037391
## % pres/err predicted correctly:  -1513.631
## % of predictable range [ (model-null)/(1-null) ]:  0.03743494
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8679      -0.2472
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 4336  AIC: 4880
## log likelihood:  -2168.217
## Nagelkerke R2:  0.0478757
## % pres/err predicted correctly:  -1515.996
## % of predictable range [ (model-null)/(1-null) ]:  0.03593143
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.84297     -0.24397      0.01113
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 4336  AIC: 4881
## log likelihood:  -2168.05
## Nagelkerke R2:  0.04799477
## % pres/err predicted correctly:  -1516.039
## % of predictable range [ (model-null)/(1-null) ]:  0.03590447
## ***************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9411
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4078 Residual
## Null Deviance:       4469
## Residual Deviance: 4469  AIC: 5024
## log likelihood:  -2234.269
## Nagelkerke R2:  -3.335871e-16
## % pres/err predicted correctly:  -1572.536
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0731122	0.2463404	-0.0296866	-0.6720759	NA	NA	0.0347729	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.5508944	NA	NA	0.0501313	NA	NA	-0.0281893	-0.0302682
preserved ~ stimlen + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0829239	NA	NA	-0.6813407	NA	NA	0.0357902	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0743467	0.3633299	-0.0344686	-0.6827089	NA	-0.047371	0.0362588	NA	0.0054964	NA	NA

Model	AIC	Delta_AIC	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 + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0797048	[illegible]	0.0107064	NA	-0.6808660	NA	NA	0.0357374	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0776603	[illegible]	0.1047312	NA	-0.6832633	-0.0297199	NA	0.0358859	0.0034870	NA	NA	NA
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.6678680	NA	NA	0.0313213	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1293047	NA	-0.6685034	-0.0307010	NA	0.0315293	0.0032011	NA	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0526348	[illegible]	0.2652616	-0.0320943	-0.3579009	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0524365	[illegible]	0.2655643	-0.0312073	-0.3583538	NA	-0.001647	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0626199	[illegible]	NA	NA	-0.3579116	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1146356	[illegible]	NA	NA	-0.4792721	NA	NA	NA	NA	NA	0.0142837	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0591028	[illegible]	0.0117797	NA	-0.3579146	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0555783	[illegible]	0.0633312	NA	-0.3610471	-0.0116308	NA	NA	NA	NA	NA	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.3787469	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0832886	NA	-0.3792821	-0.0138926	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0223540	NA	-0.3765030	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2377455	[illegible]	0.2553497	-0.0306252	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2472446	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2439658	[illegible]	0.0111313	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ 1	5023.545	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          3.53918          -0.07311           0.24634           0.03477          -0.67208
## stimlen:log_freq
##         -0.02969
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3939  AIC: 4475
```

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

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

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

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

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

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

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

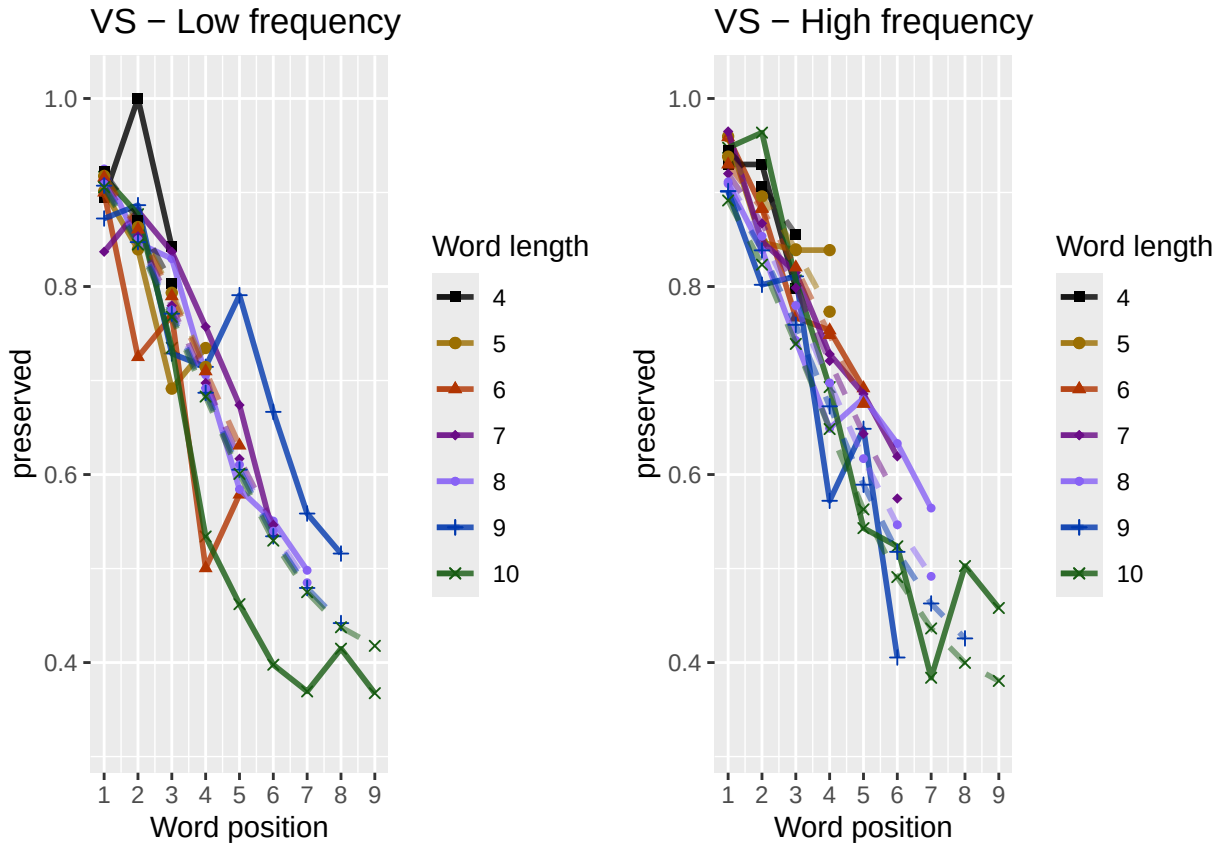
```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
```

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

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
```

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

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




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

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

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

```
##
## Coefficients:
## (Intercept)       CumErr
##      1.4689      -0.7698
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3954  AIC: 4434
## log likelihood:  -1976.819
## Nagelkerke R2:  0.178162
## % pres/err predicted correctly:  -1344.689
## % of predictable range [ (model-null)/(1-null) ]:  0.144799
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.04950      0.03132     -0.66787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3955  AIC: 4490
## log likelihood:  -1977.724
## Nagelkerke R2:  0.1775739
## % pres/err predicted correctly:  -1364.781
## % of predictable range [ (model-null)/(1-null) ]:  0.1320303
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.5136      -0.3787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3971  AIC: 4506
## log likelihood:  -1985.342
## Nagelkerke R2:  0.1726169
## % pres/err predicted correctly:  -1368.724
## % of predictable range [ (model-null)/(1-null) ]:  0.1295248
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
```

```
##      1.6068      -0.2785
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 4285  AIC: 4850
## log likelihood:  -2142.584
## Nagelkerke R2:  0.06604162
## % pres/err predicted correctly:  -1499.852
## % of predictable range [ (model-null)/(1-null) ]:  0.04619154
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8679      -0.2472
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 4336  AIC: 4880
## log likelihood:  -2168.217
## Nagelkerke R2:  0.0478757
## % pres/err predicted correctly:  -1515.996
## % of predictable range [ (model-null)/(1-null) ]:  0.03593143
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9411
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4078 Residual
## Null Deviance:       4469
## Residual Deviance: 4469  AIC: 5024
## log likelihood:  -2234.269
## Nagelkerke R2:  -3.335871e-16
## % pres/err predicted correctly:  -1572.536
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	4434.268	0.00000	1	1	0.1781620	0.4689128	NA	-0.7697735	NA	NA	NA
preserved ~ (I(pos^2) + pos)	4489.541	55.27362	0	0	0.1775739	0.0495029	NA	NA	0.0313213	-0.6678680	NA
preserved ~ pos	4505.569	71.30095	0	0	0.1726162	2.5136215	NA	NA	NA	-0.3787469	NA
preserved ~ CumPres	4849.529	415.26155	0	0	0.0660416	6.6068225	-0.2784994	NA	NA	NA	NA
preserved ~ stimlen	4879.791	445.52350	0	0	0.0478757	7.8678648	NA	NA	NA	NA	-0.2472446
preserved ~ 1	5023.545	589.27763	0	0	0.0000000	0.9410632	NA	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	0.7643302	535
O	0.7212948	1878
P	0.9264706	34
S	0.7717007	245
V	0.6849375	1387

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4331      -0.8001
##
## Degrees of Freedom: 3799 Total (i.e. Null);  3798 Residual
## Null Deviance:       4190
## Residual Deviance: 3716  AIC: 4173
## log likelihood:  -1858
## Nagelkerke R2:  0.1756682
## % pres/err predicted correctly:  -1267.866
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.1426339
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.98557      0.03003     -0.65082
##
## Degrees of Freedom: 3799 Total (i.e. Null);  3797 Residual
## Null Deviance:       4190
## Residual Deviance: 3720  AIC: 4230
## log likelihood:  -1860.242
## Nagelkerke R2:  0.174108
## % pres/err predicted correctly:  -1287.637
## % of predictable range [ (model-null)/(1-null) ]:  0.1292753
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.4727      -0.3739
##
## Degrees of Freedom: 3799 Total (i.e. Null);  3798 Residual
## Null Deviance:       4190
## Residual Deviance: 3734  AIC: 4244
## log likelihood:  -1866.869
## Nagelkerke R2:  0.1694861
## % pres/err predicted correctly:  -1291.002
## % of predictable range [ (model-null)/(1-null) ]:  0.1270012
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.5332      -0.2798
##
## Degrees of Freedom: 3799 Total (i.e. Null);  3798 Residual
## Null Deviance:       4190
## Residual Deviance: 4031  AIC: 4567
## log likelihood:  -2015.319
## Nagelkerke R2:  0.06161174
## % pres/err predicted correctly:  -1415.789
## % of predictable range [ (model-null)/(1-null) ]:  0.04268352
## ***************************
## model index:  5
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8259      -0.2452
##
## Degrees of Freedom: 3799 Total (i.e. Null);  3798 Residual
## Null Deviance:       4190
## Residual Deviance: 4068  AIC: 4583
## log likelihood:  -2033.754
## Nagelkerke R2:  0.04761716
## % pres/err predicted correctly:  -1426.055
## % of predictable range [ (model-null)/(1-null) ]:  0.03574688
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.9153
##
## Degrees of Freedom: 3799 Total (i.e. Null);  3799 Residual
## Null Deviance:       4190
## Residual Deviance: 4190  AIC: 4716
## log likelihood:  -2095.175
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -1478.959
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(posˆ2)	pos	stimlen
preserved ~ CumErr	4172.576	0.00000	1	1	0.1756682	2.4330982	NA	-0.8001232	NA	NA	NA
preserved ~ (I(posˆ2) + pos)	4230.180	57.60344	0	0	0.1741082	0.9855653	NA	NA	0.0300269	-0.6508191	NA
preserved ~ pos	4244.339	71.76211	0	0	0.1694862	2.4727166	NA	NA	NA	-0.3739140	NA
preserved ~ CumPres	4566.983	394.40671	0	0	0.0616117	7.5331767	-0.2798136	NA	NA	NA	NA
preserved ~ stimlen	4583.214	410.63714	0	0	0.0476172	2.8258730	NA	NA	NA	NA	-0.2451885
preserved ~ 1	4715.957	543.38067	0	0	0.0000000	0.9152993	NA	NA	NA	NA	NA

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

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

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

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

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.9718       0.0339      -0.6806
##
## Degrees of Freedom: 3264 Total (i.e. Null);  3262 Residual
## Null Deviance:       3640
## Residual Deviance: 3212  AIC: 3643
## log likelihood:  -1606.077
## Nagelkerke R2:  0.1827332
## % pres/err predicted correctly:  -1112.834
## % of predictable range [ (model-null)/(1-null) ]:  0.1371313
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.4106      -0.3704
##
## Degrees of Freedom: 3264 Total (i.e. Null);  3263 Residual
## Null Deviance:       3640
## Residual Deviance: 3228  AIC: 3661
## log likelihood:  -1614.083
## Nagelkerke R2:  0.1763156
```

```
## % pres/err predicted correctly:  -1117.491
## % of predictable range [ (model-null)/(1-null) ]:  0.1335243
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.3397      -0.8304
##
## Degrees of Freedom: 3264 Total (i.e. Null);  3263 Residual
## Null Deviance:       3640
## Residual Deviance: 3282  AIC: 3677
## log likelihood:  -1641.144
## Nagelkerke R2:  0.1543911
## % pres/err predicted correctly:  -1126.767
## % of predictable range [ (model-null)/(1-null) ]:  0.1263377
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.6152      -0.3983
##
## Degrees of Freedom: 3264 Total (i.e. Null);  3263 Residual
## Null Deviance:       3640
## Residual Deviance: 3439  AIC: 3892
## log likelihood:  -1719.385
## Nagelkerke R2:  0.08891703
## % pres/err predicted correctly:  -1206.029
## % of predictable range [ (model-null)/(1-null) ]:  0.06493507
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       2.883       -0.258
##
## Degrees of Freedom: 3264 Total (i.e. Null);  3263 Residual
## Null Deviance:       3640
## Residual Deviance: 3520  AIC: 3966
## log likelihood:  -1760.224
## Nagelkerke R2:  0.05347555
## % pres/err predicted correctly:  -1238.003
## % of predictable range [ (model-null)/(1-null) ]:  0.04016564
## **************************
```

```
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##      0.8753
##
## Degrees of Freedom: 3264 Total (i.e. Null);  3264 Residual
## Null Deviance:       3640
## Residual Deviance: 3640  AIC: 4096
## log likelihood:  -1819.972
## Nagelkerke R2:  1.652041e-16
## % pres/err predicted correctly:  -1289.85
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

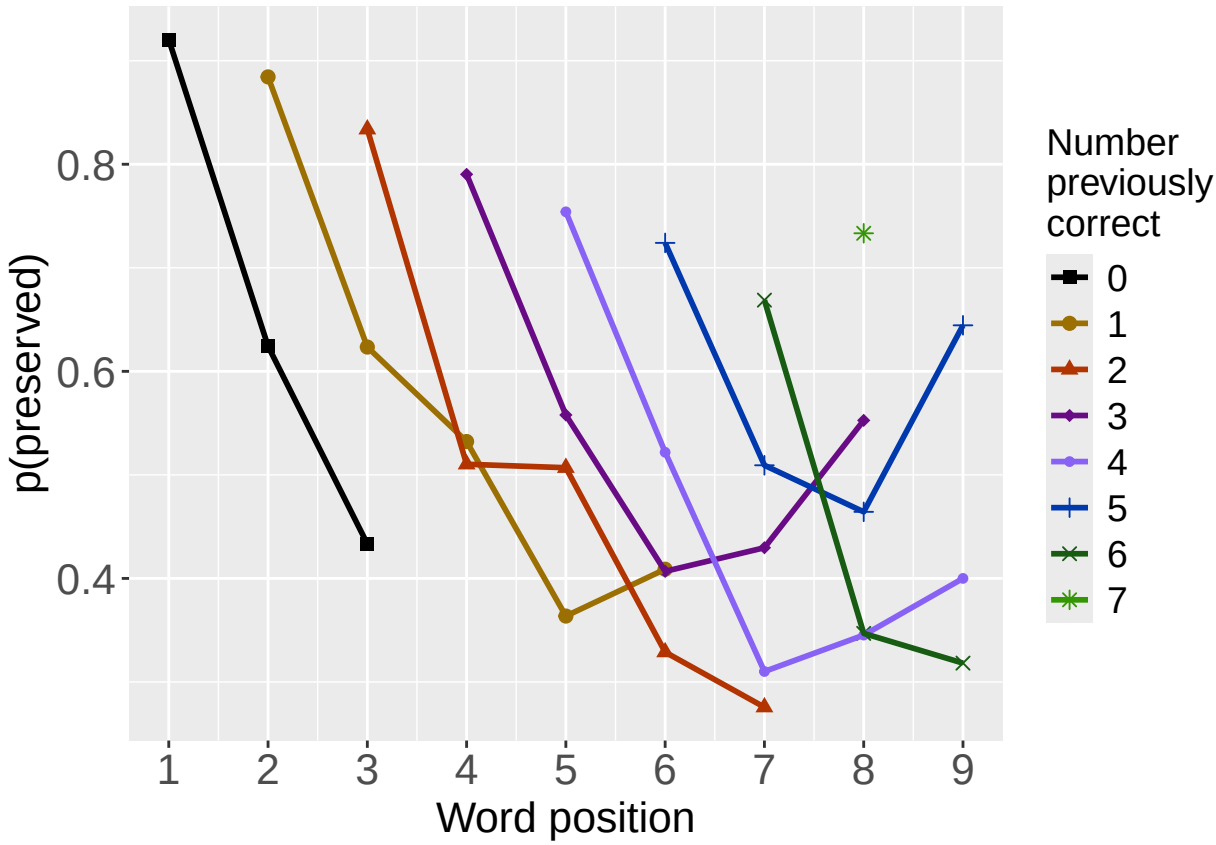
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ (I(pos^2) + pos)	3643.397	0.00000	1.000000	0.9998437	7.182733e-02	2.9717958	NA	NA	0.0339047	-0.6806019	NA
preserved ~ pos	3660.925	17.52811	0.000156	2.000156e-04	2.176315e-02	3.4106159	NA	NA	NA	-0.3703533	NA
preserved ~ CumErr	3676.761	33.36388	0.000000	1.000000e-07	1.154391e-03	1.3396810	NA	-0.8303676	NA	NA	NA
preserved ~ CumPres	3892.277	248.88034	0.000000	0.000000e+00	8.889170e-05	0.6152139	-0.3982809	NA	NA	NA	NA
preserved ~ stimlen	3966.353	322.95616	0.000000	0.000000e+00	5.347528e-06	0.8829860	NA	NA	NA	NA	-0.2579917
preserved ~ 1	4095.642	452.24533	0.000000	0.000000e+00	0.000000e+00	0.8753125	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.
```

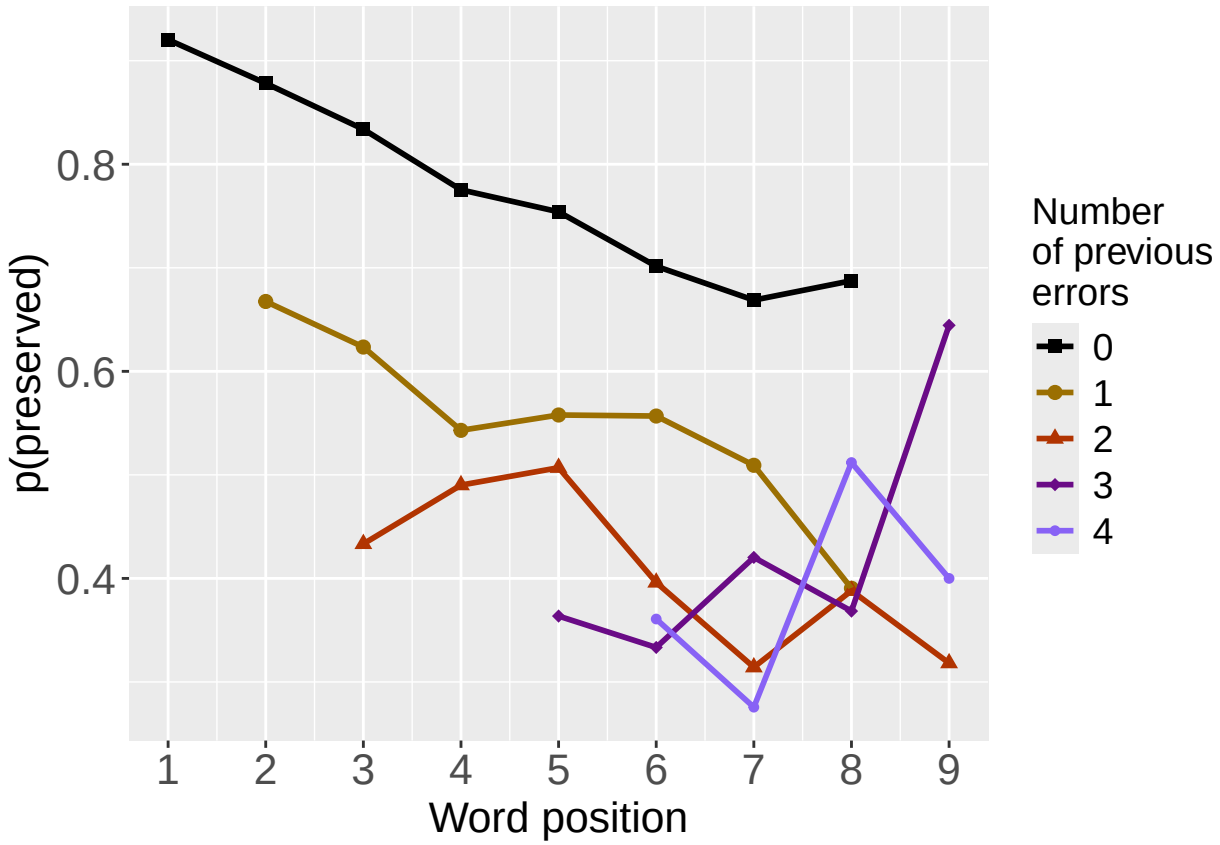
```
print(PrevCorPlot)
```




```
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
```

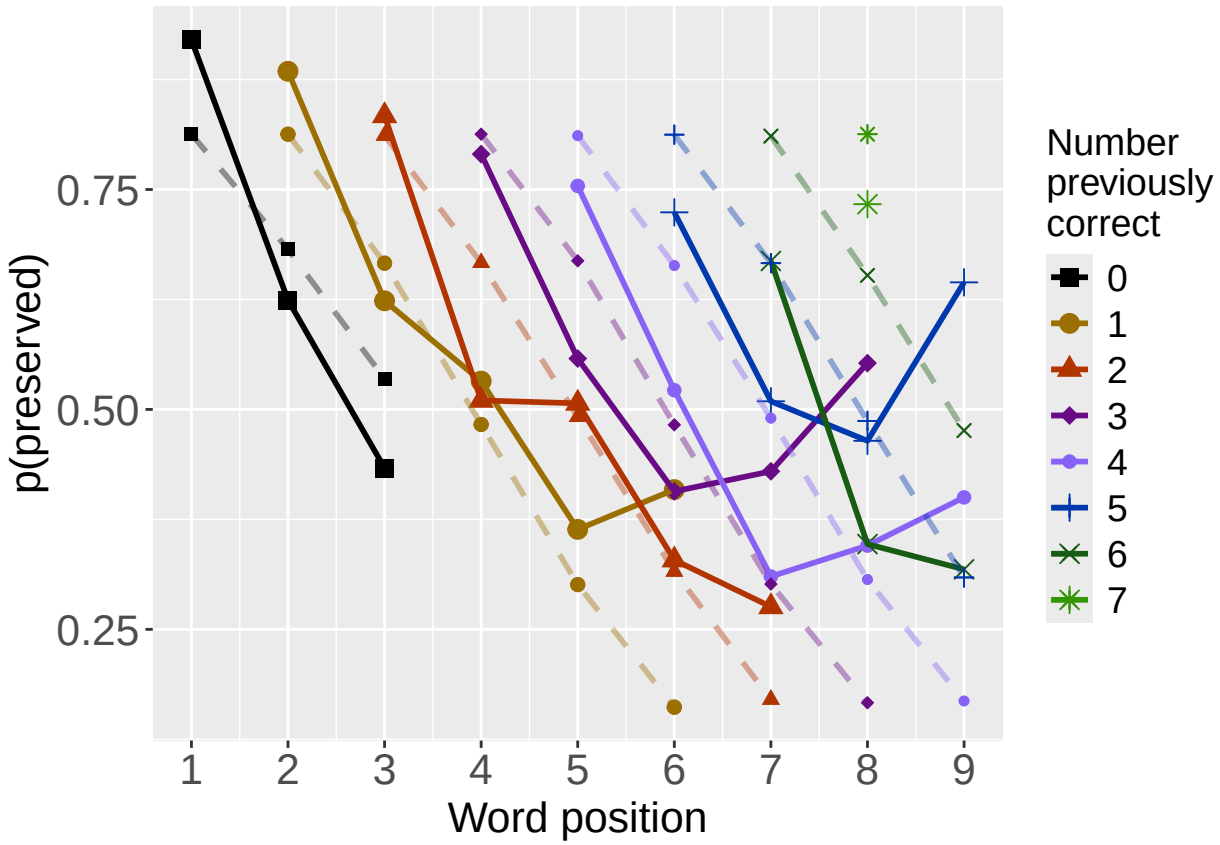
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

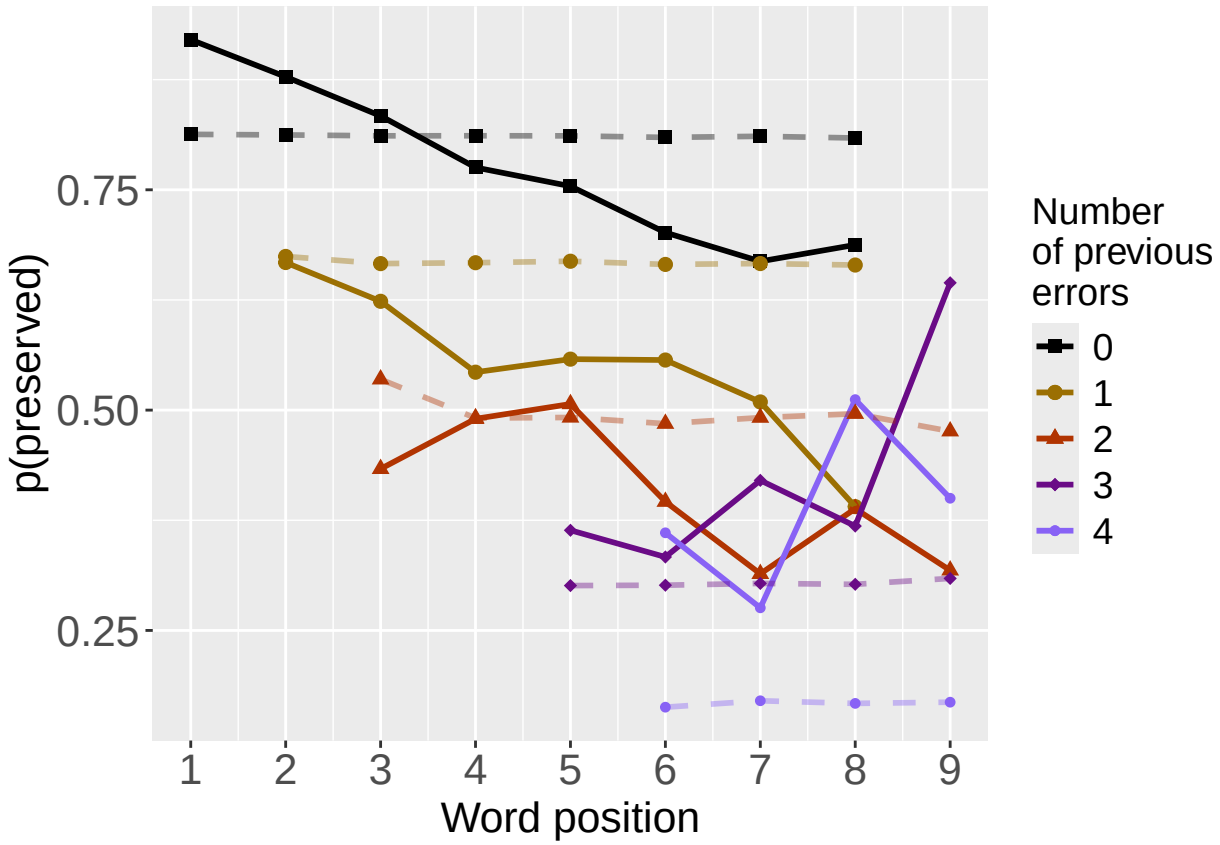
```
print(PrevCorPlot)
```




```
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)       CumErr     I(pos^2)          pos
##     3.06824     -0.52672      0.05233     -0.69472
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3806  AIC: 4297
## log likelihood:  -1903.2
## Nagelkerke R2:  0.2251079
## % pres/err predicted correctly:  -1296.17
## % of predictable range [ (model-null)/(1-null) ]:  0.1756337
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4689      -0.7698
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3954  AIC: 4434
## log likelihood:  -1976.819
## Nagelkerke R2:  0.178162
## % pres/err predicted correctly:  -1344.689
## % of predictable range [ (model-null)/(1-null) ]:  0.144799
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.04950      0.03132     -0.66787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3955  AIC: 4490
## log likelihood:  -1977.724
## Nagelkerke R2:  0.1775739
## % pres/err predicted correctly:  -1364.781
## % of predictable range [ (model-null)/(1-null) ]:  0.1320303
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	4296.566	0.0000	1	1	0.2251079	3.068236	-0.5267192	0.0523327	-0.6947248

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	4434.268	137.7013	0	0	0.1781620	1.468913	-0.7697735	NA	NA
preserved ~ I(posˆ2) + pos	4489.541	192.9749	0	0	0.1775739	3.049503	NA	0.0313213	-0.6678680

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##      2.3490      -0.7162      -0.1181
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3929  AIC: 4410
## log likelihood:  -1964.475
## Nagelkerke R2:  0.1861523
## % pres/err predicted correctly:  -1334.78
## % of predictable range [ (model-null)/(1-null) ]:  0.1510966
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4689      -0.7698
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3954  AIC: 4434
## log likelihood:  -1976.819
## Nagelkerke R2:  0.178162
## % pres/err predicted correctly:  -1344.689
## % of predictable range [ (model-null)/(1-null) ]:  0.144799
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.8679      -0.2472
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 4336  AIC: 4880
## log likelihood:  -2168.217
## Nagelkerke R2:  0.0478757
## % pres/err predicted correctly:  -1515.996
## % of predictable range [ (model-null)/(1-null) ]:  0.03593143
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	4409.743	0.00000	1.0e+00	0.9999953	0.1861523	2.348972	-0.7162471	-0.1180621
preserved ~ CumErr	4434.268	24.52423	4.7e-06	0.0000047	0.1781620	1.468913	-0.7697735	NA
preserved ~ stimlen	4879.791	470.04774	0.0e+00	0.0000000	0.0478757	2.867865	NA	-0.2472446

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      1.9972      -0.7135      -0.2319
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
```

```
## Null Deviance:       4469
## Residual Deviance: 3845  AIC: 4342
## log likelihood:  -1922.714
## Nagelkerke R2:  0.212829
## % pres/err predicted correctly:  -1307.729
## % of predictable range [ (model-null)/(1-null) ]:  0.1682879
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4689      -0.7698
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3954  AIC: 4434
## log likelihood:  -1976.819
## Nagelkerke R2:  0.178162
## % pres/err predicted correctly:  -1344.689
## % of predictable range [ (model-null)/(1-null) ]:  0.144799
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.6068      -0.2785
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 4285  AIC: 4850
## log likelihood:  -2142.584
## Nagelkerke R2:  0.06604162
## % pres/err predicted correctly:  -1499.852
## % of predictable range [ (model-null)/(1-null) ]:  0.04619154
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	4342.153	0.00000	1	1	0.2128290	1.997196	-0.7134854	-0.2318829
preserved ~ CumErr	4434.268	92.11464	0	0	0.1781620	1.468913	-0.7697735	NA
preserved ~ CumPres	4849.529	507.37619	0	0	0.0660416	1.606823	NA	-0.2784994

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.2291      -0.4816      -0.2319
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3845  AIC: 4342
## log likelihood:  -1922.714
## Nagelkerke R2:  0.212829
## % pres/err predicted correctly:  -1307.729
## % of predictable range [ (model-null)/(1-null) ]:  0.1682879
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4689      -0.7698
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3954  AIC: 4434
## log likelihood:  -1976.819
## Nagelkerke R2:  0.178162
## % pres/err predicted correctly:  -1344.689
## % of predictable range [ (model-null)/(1-null) ]:  0.144799
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.5136      -0.3787
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3971  AIC: 4506
## log likelihood:  -1985.342
## Nagelkerke R2:  0.1726169
## % pres/err predicted correctly:  -1368.724
## % of predictable range [ (model-null)/(1-null) ]:  0.1295248
## **************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	4342.153	0.00000	1	1	0.2128290	2.229079	-0.4816026	-0.2318829
preserved ~ CumErr	4434.268	92.11464	0	0	0.1781620	1.468913	-0.7697735	NA
preserved ~ pos	4505.569	163.41560	0	0	0.1726169	2.513622	NA	-0.3787469

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	4296.566	0.00000	1.0e+00	1.000000	0.2251073	2.068236	-0.5267192	0.0523327	-0.6947248	NA	NA
preserved ~ CumErr + pos	4342.153	0.00000	1.0e+00	1.000000	0.2128290	2.229079	-0.4816026	NA	-0.2318829	NA	NA
preserved ~ CumErr + CumPres	4342.153	0.00000	1.0e+00	1.000000	0.2128290	1.997196	-0.7134854	NA	NA	NA	-0.2318829
preserved ~ CumErr + stimlen	4409.749	0.00000	1.0e+00	0.999995	3.186152e-01	2.348972	-0.7162471	NA	NA	-0.1180621	NA
preserved ~ CumErr	4434.268	37.70132	2.0e+00	0.000000	0.1781620	1.468913	-0.7697735	NA	NA	NA	NA
preserved ~ CumErr	4434.268	84.52423	4.7e-06	0.000004	7.1781620	1.468913	-0.7697735	NA	NA	NA	NA
preserved ~ CumErr	4434.268	92.11464	0.0e+00	0.000000	0.1781620	1.468913	-0.7697735	NA	NA	NA	NA
preserved ~ CumErr	4434.268	92.11464	0.0e+00	0.000000	0.1781620	1.468913	-0.7697735	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	4489.541	192.97499	4.0e+00	0.000000	0.1775739	2.049503	NA	0.0313213	-0.6678680	NA	NA
preserved ~ pos	4505.569	163.41560	0.0e+00	0.000000	0.1726169	2.513622	NA	NA	-0.3787469	NA	NA
preserved ~ CumPres	4849.525	507.37619	0.0e+00	0.000000	0.0066041	1.606823	NA	NA	NA	NA	-0.2784994
preserved ~ stimlen	4879.794	470.04770	4.0e+00	0.000000	0.0047857	2.867865	NA	NA	NA	-0.2472446	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.62005     -0.52382      0.05634     -0.70704     -0.07588
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4074 Residual
## Null Deviance:       4469
## Residual Deviance: 3798  AIC: 4287
## log likelihood:  -1899.107
## Nagelkerke R2:  0.2276686
## % pres/err predicted correctly:  -1292.053
## % of predictable range [ (model-null)/(1-null) ]:  0.1782497
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     3.58601     -0.52433      0.05629     -0.70645     -0.07152      0.01482
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4073 Residual
## Null Deviance:       4469
## Residual Deviance: 3798  AIC: 4288
## log likelihood:  -1898.852
## Nagelkerke R2:  0.2278282
## % pres/err predicted correctly:  -1291.936
## % of predictable range [ (model-null)/(1-null) ]:  0.1783245
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     3.06346     -0.52732      0.05266     -0.69483      0.02710
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4074 Residual
## Null Deviance:       4469
## Residual Deviance: 3805  AIC: 4296
## log likelihood:  -1902.299
## Nagelkerke R2:  0.225672
## % pres/err predicted correctly:  -1295.502
## % of predictable range [ (model-null)/(1-null) ]:  0.1760583
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     3.06824     -0.52672      0.05233     -0.69472
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3806  AIC: 4297
## log likelihood:  -1903.2
## Nagelkerke R2:  0.2251079
## % pres/err predicted correctly:  -1296.17
## % of predictable range [ (model-null)/(1-null) ]:  0.1756337
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##      0.9411
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4078 Residual
## Null Deviance:       4469
## Residual Deviance: 4469  AIC: 5024
## log likelihood:  -2234.269
## Nagelkerke R2:  -3.335871e-16
## % pres/err predicted correctly:  -1572.536
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen	4286.871	0.000000	1.000000	0.6726667	0.2276638	6.6200482	- 0.5238206	0.0563395	- 0.7070359	NA	- 0.0758790
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	4288.381	1.508862	0.4702782	0.3163409	0.2278283	7.5860090	- 0.5243310	0.0562907	- 0.7064529	0.0148208	- 0.0715192
preserved ~ CumErr + I(pos^2) + pos + log_freq	4296.408	9.536928	0.008496	0.005716	0.2256732	0.0634585	- 0.5273223	0.0526563	- 0.6948286	0.0270966	NA
preserved ~ CumErr + I(pos^2) + pos	4296.566	9.695282	0.007846	0.005278	0.2251037	0.0682361	- 0.5267192	0.0523327	- 0.6947248	NA	NA
preserved ~ 1	5023.547	736.674234	0.000000	0.000000	0.0000000	0.9410632	NA	NA	NA	NA	NA

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

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

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

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

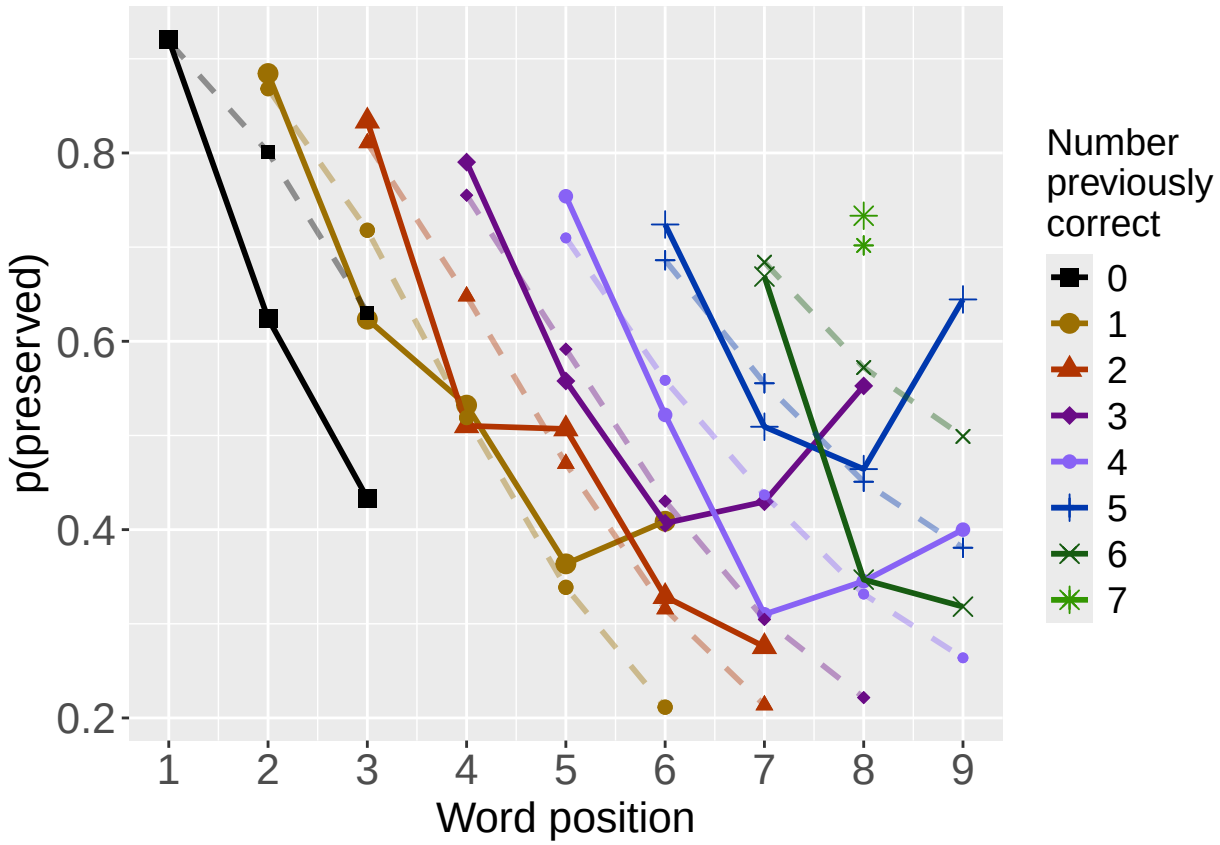
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen
##          Df Deviance    AIC
## CumErr    1   3945.4 4432.0
## pos       1   3882.7 4369.3
## I(pos^2)  1   3842.2 4328.8
## stimlen   1   3806.4 4293.1
## <none>        3798.2 4286.9
```

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

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

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

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

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

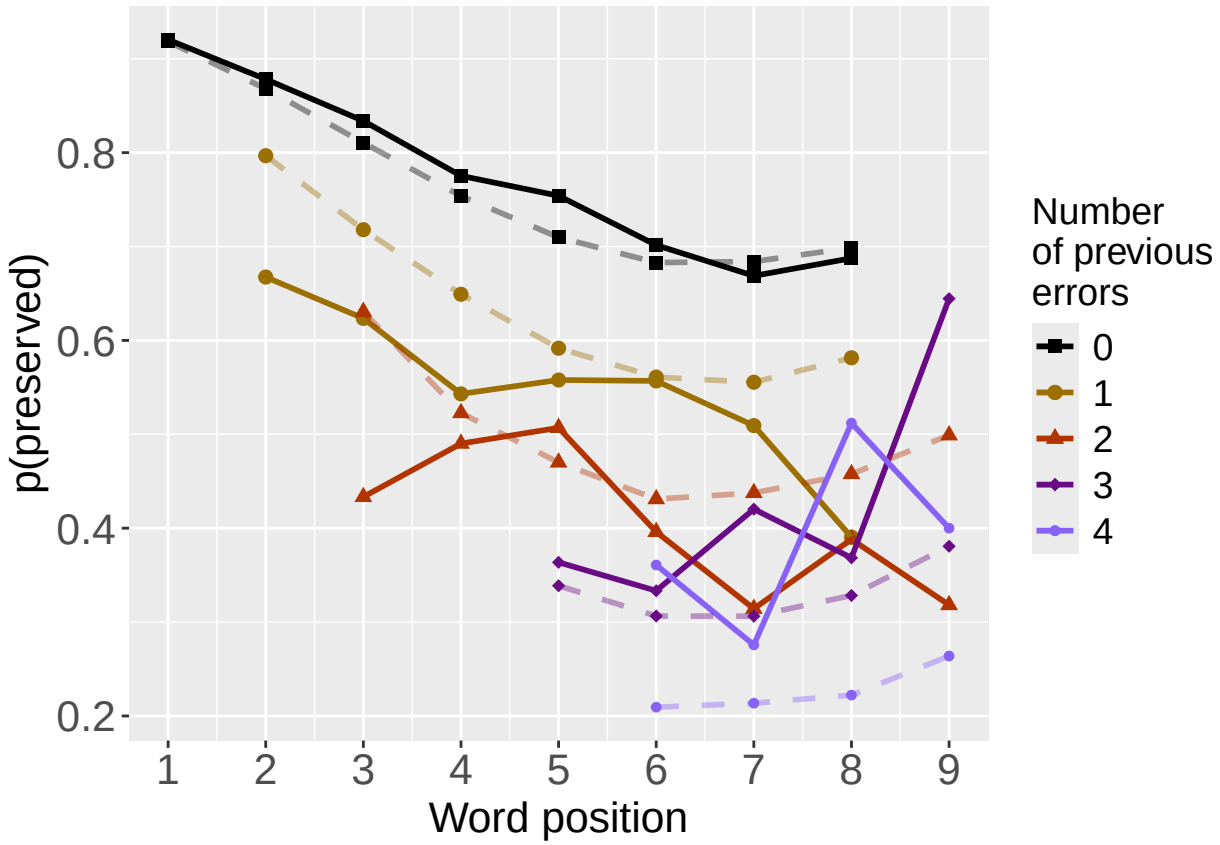
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

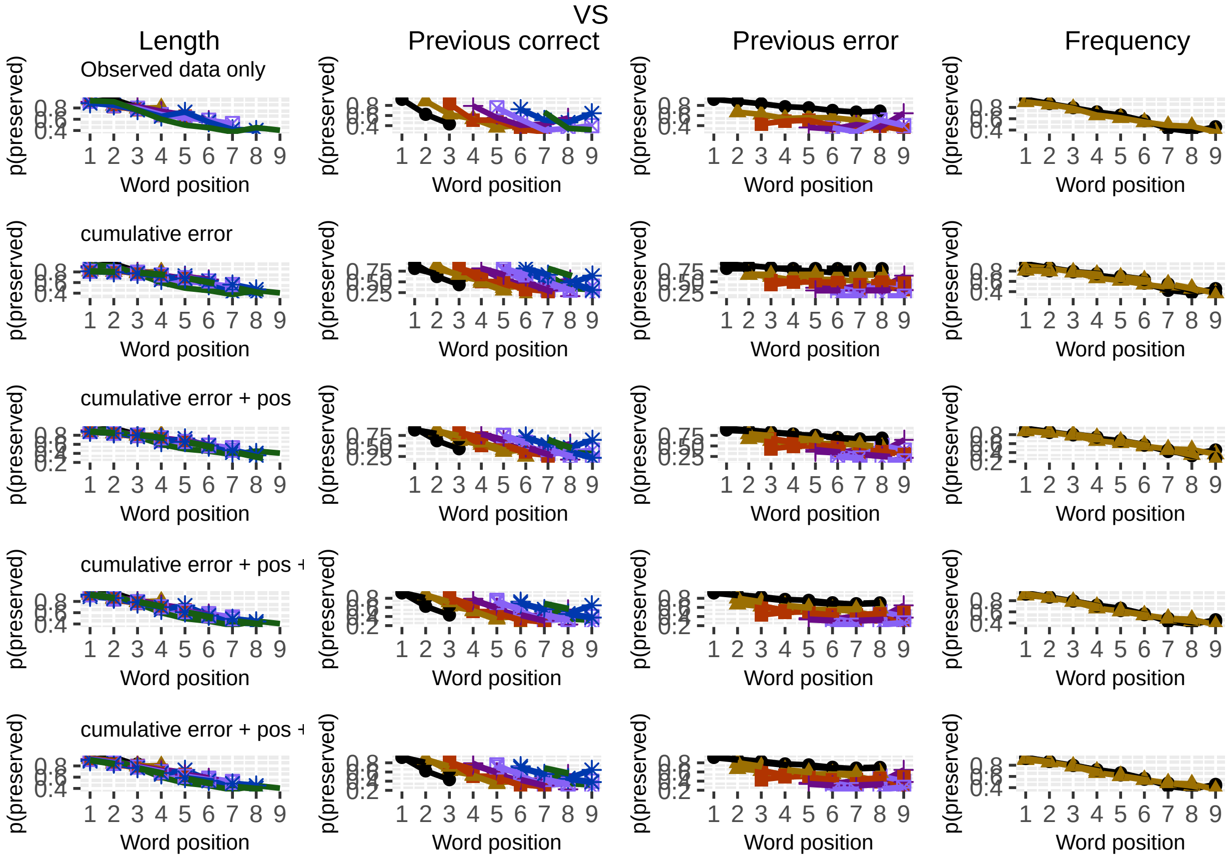
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4689      -0.7698
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4077 Residual
## Null Deviance:       4469
## Residual Deviance: 3954  AIC: 4434
## log likelihood:  -1976.819
```

```
## Nagelkerke R2:  0.178162
## % pres/err predicted correctly:  -1344.689
## % of predictable range [ (model-null)/(1-null) ]:  0.144799
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.2291      -0.4816      -0.2319
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4076 Residual
## Null Deviance:       4469
## Residual Deviance: 3845  AIC: 4342
## log likelihood:  -1922.714
## Nagelkerke R2:  0.212829
## % pres/err predicted correctly:  -1307.729
## % of predictable range [ (model-null)/(1-null) ]:  0.1682879
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)
##     3.06824     -0.52672     -0.69472      0.05233
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4075 Residual
## Null Deviance:       4469
## Residual Deviance: 3806  AIC: 4297
## log likelihood:  -1903.2
## Nagelkerke R2:  0.2251079
## % pres/err predicted correctly:  -1296.17
## % of predictable range [ (model-null)/(1-null) ]:  0.1756337
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)      stimlen
##     3.62005     -0.52382     -0.70704      0.05634     -0.07588
##
## Degrees of Freedom: 4078 Total (i.e. Null);  4074 Residual
## Null Deviance:       4469
## Residual Deviance: 3798  AIC: 4287
## log likelihood:  -1899.107
## Nagelkerke R2:  0.2276686
## % pres/err predicted correctly:  -1292.053
## % of predictable range [ (model-null)/(1-null) ]:  0.1782497
```

```
## ***************************
## `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 8 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need that many have
##   them.

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




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

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

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

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

	CumErr	I(pos^2)	pos	stimlen
McFadden	0.0618752	0.0332323	0.0441404	0.0090480
SquaredCorrelation	0.0685937	0.0379414	0.0496654	0.0106674
Nagelkerke	0.0685937	0.0379414	0.0496654	0.0106674
Estrella	0.0745917	0.0403023	0.0533721	0.0110415

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

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

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

```
##                                                        model deviance
## CumErr + pos + I(pos^2) + stimlen CumErr + pos + I(pos^2) + stimlen 3798.214
## CumErr + pos + I(pos^2)                     CumErr + pos + I(pos^2) 3806.400
## CumErr + pos                                          CumErr + pos 3845.427
## CumErr                                                      CumErr 3953.638
## null                                                          null 4468.537
##                                   deviance_explained percent_explained
## CumErr + pos + I(pos^2) + stimlen           670.3229          15.00095
## CumErr + pos + I(pos^2)                     662.1369          14.81776
## CumErr + pos                                623.1100          13.94438
## CumErr                                      514.8992          11.52277
## null                                          0.0000           0.00000
##                                   percent_of_explained_deviance increment_in_explained
## CumErr + pos + I(pos^2) + stimlen                     100.00000               1.221199
## CumErr + pos + I(pos^2)                                98.77880               5.822117
## CumErr + pos                                           92.95668              16.143082
## CumErr                                                 76.81360              76.813602
## null                                                         NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + I(pos^2) + stimlen	3798.214	670.3229
CumErr + pos + I(pos^2)	3806.400	662.1369
CumErr + pos	3845.427	623.1100
CumErr	3953.638	514.8992
null	4468.537	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + I(pos^2) + stimlen	15.00095	100.00000	1.221199
CumErr + pos + I(pos^2)	14.81776	98.77880	5.822117
CumErr + pos	13.94438	92.95668	16.143082
CumErr	11.52277	76.81360	76.813602
null	0.00000	NA	0.000000

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

```
##           Nagelkerke
## CumErr    0.41106601
## I(pos^2)  0.22737389
## pos       0.29763297
## stimlen   0.06392713
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.5696658	3953.638
preserved ~ CumErr+pos	0.6233833	3845.427
preserved ~ CumErr+pos+I(pos^2)	0.6597926	3806.400
preserved ~ CumErr+pos+I(pos^2)+stimlen	0.6644635	3798.214

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

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

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

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

```
##                                   model p_accounted_for model_deviance diff_CumErr
## 1                     preserved ~ CumErr       0.5696658       3953.638  0.00000000
## 2                  preserved ~ CumErr+pos       0.6233833       3845.427  0.05371750
## 3         preserved ~ CumErr+pos+I(pos^2)       0.6597926       3806.400  0.09012674
## 4 preserved ~ CumErr+pos+I(pos^2)+stimlen       0.6644635       3798.214  0.09479763
##   diff_CumErr+pos diff_CumErr+pos+I(pos^2) diff_CumErr+pos+I(pos^2)+stimlen
## 1     -0.05371750             -0.090126741                      -0.094797628
## 2      0.00000000             -0.036409238                      -0.041080126
## 3      0.03640924              0.000000000                      -0.004670887
## 4      0.04108013              0.004670887                       0.000000000
```

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

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

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

model	diff_CumErr	diff_CumErr+pos	diff_CumErr+pos+I(pos^2)
preserved ~ CumErr	0.0000000	-0.0537175	-0.0901267
preserved ~ CumErr+pos	0.0537175	0.0000000	-0.0364092
preserved ~ CumErr+pos+I(pos^2)	0.0901267	0.0364092	0.0000000
preserved ~ CumErr+pos+I(pos^2)+stimlen	0.0947976	0.0410801	0.0046709