AP - 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	556	35	130	NA	NA	721
2	67	NA	444	98	112	721
3	317	NA	174	214	16	721
4	308	NA	244	70	39	661
5	238	NA	216	73	39	566
6	211	1	142	71	22	447
7	180	NA	104	29	19	332
8	94	NA	55	24	4	177
9	74	NA	2	NA	7	83

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7711512	0.0485437	0.1803051	NA	NA	721
2	0.0929265	NA	0.6158114	0.1359223	0.1553398	721
3	0.4396671	NA	0.2413315	0.2968100	0.0221914	721
4	0.4659607	NA	0.3691377	0.1059002	0.0590015	661
5	0.4204947	NA	0.3816254	0.1289753	0.0689046	566
6	0.4720358	0.0022371	0.3176734	0.1588367	0.0492170	447

pos_factor	O	P	V	1	S	total
7	0.5421687	NA	0.3132530	0.0873494	0.0572289	332
8	0.5310734	NA	0.3107345	0.1355932	0.0225989	177
9	0.8915663	NA	0.0240964	NA	0.0843373	83

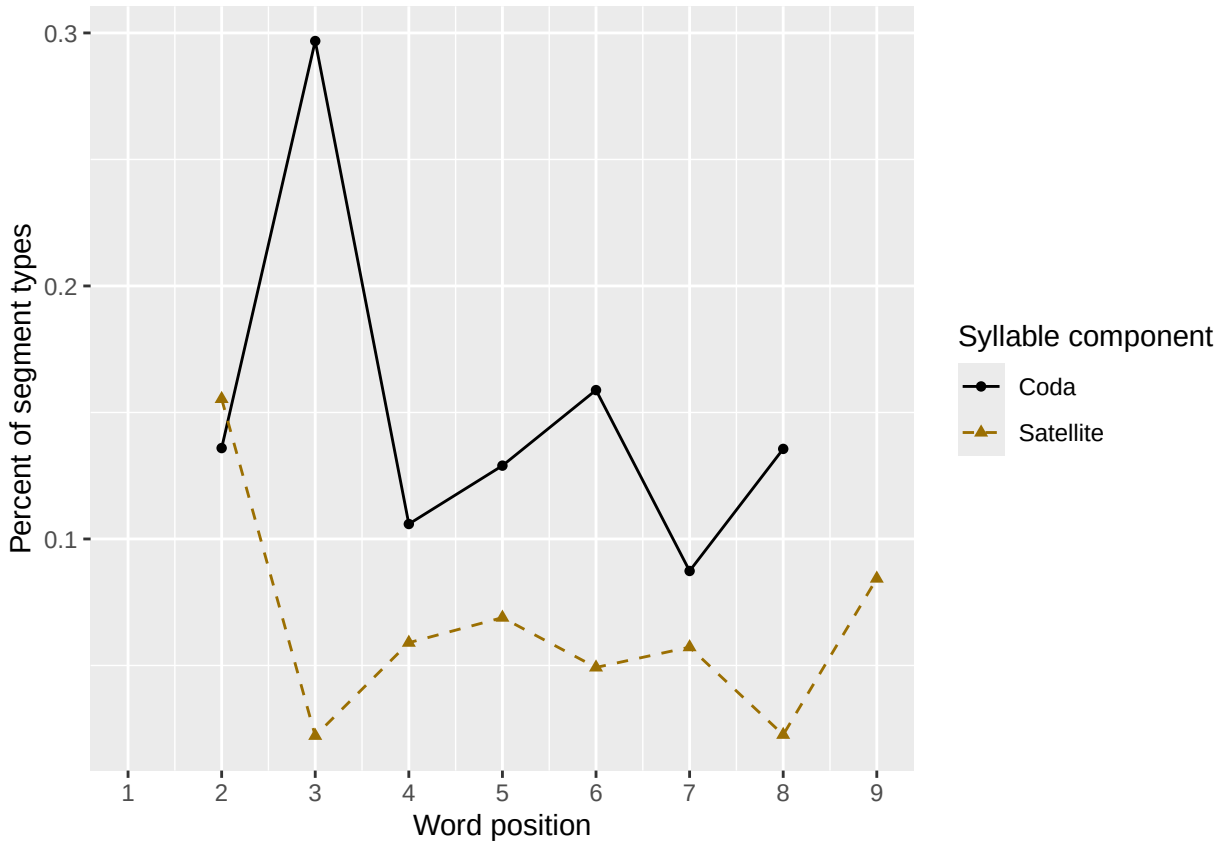
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.917 0.983 0.983 NA     NA     NA     NA     NA     NA
## 2       5 0.926 0.947 0.968  0.958 NA     NA     NA     NA     NA
## 3       6 0.916 0.964 0.952  0.950  0.961 NA     NA     NA     NA
## 4       7 0.913 0.974 0.948  0.961  0.961  0.991 NA     NA     NA
## 5       8 0.887 0.955 0.955  0.945  0.948  0.952  0.945 NA     NA
## 6       9 0.957 0.959 0.920  0.933  0.918  0.947  0.952  0.938 NA
## 7      10 0.928 0.972 0.952  0.940  0.968  0.928  0.970  0.950  0.928
```

```
# 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.
```

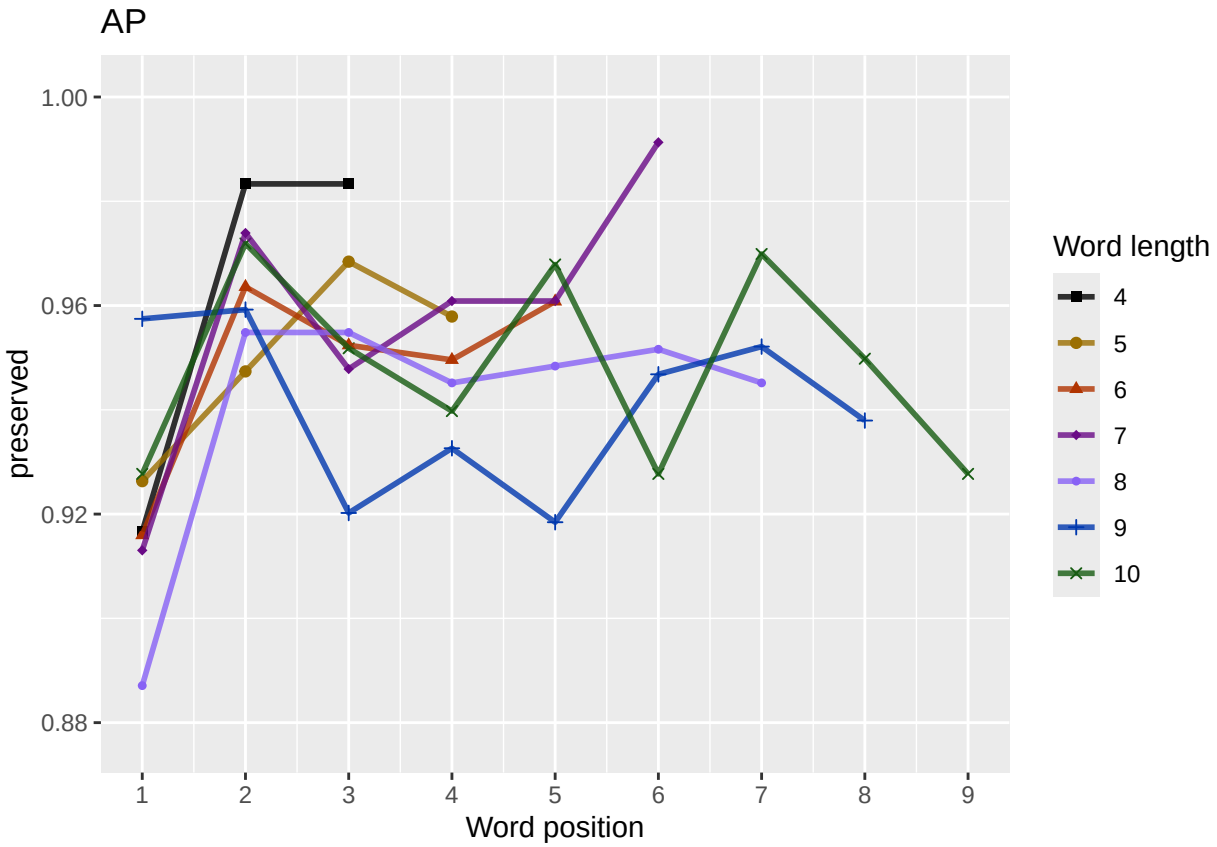
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    60    60    60    NA    NA    NA    NA    NA    NA
## 2       5    95    95    95    95    NA    NA    NA    NA    NA
## 3       6   119   119   119   119   119    NA    NA    NA    NA
## 4       7   115   115   115   115   115   115    NA    NA    NA
## 5       8   155   155   155   155   155   155   155    NA    NA
## 6       9    94    94    94    94    94    94    94    94    NA
## 7      10    83    83    83    83    83    83    83    83    83
```

```
obs_linetypes <- c("solid","solid","solid","solid",
                   "solid","solid","solid","solid","solid")
pred_linetypes <- "dashed"
pos_len_summary$stimlen<-factor(pos_len_summary$stimlen)
pos_len_summary$pos<-factor(pos_len_summary$pos)
# len_pos_plot <- ggplot(pos_len_summary,aes(x=pos,y=preserved,group=stimlen,shape=stimlen,color=stimle
len_pos_plot <- plot_len_pos_obs_predicted(PosDat,
                                           paste0(CurPat),
                                           NULL,
                                           c(min_preserved,max_preserved),
                                           palette_values,
                                           shape_values,
                                           obs_linetypes,
                                           pred_linetypes)
```

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

```
ggsave(paste0(FigDir,CurPat,"_",CurTask,"_percent_preserved_by_length_pos.png"),
       plot=len_pos_plot,device="png",unit="cm",width=15,height=11)
len_pos_plot
```



Length and position

```
# length and position

LPModelEquations<-c("preserved ~ 1",
            "preserved ~ stimlen",
            "preserved ~ pos",
            "preserved ~ stimlen + pos",
            "preserved ~ stimlen*pos",
            "preserved ~ I(pos^2)+pos",
            "preserved ~ stimlen + I(pos^2) + pos",
            "preserved ~ stimlen * (I(pos^2) + pos)"
)

LPRes<-TestModels(LPModelEquations,PosDat)
```

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

```
## **************************
## model index:  6
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.27136     -0.03485      0.34100
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1765  AIC: 1883
## log likelihood:  -882.592
## Nagelkerke R2:  0.005762917
## % pres/err predicted correctly:  -437.4149
## % of predictable range [ (model-null)/(1-null) ]:  0.001958543
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.04452      0.06847      0.46804     -0.04716
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1764  AIC: 1883
## log likelihood:  -882.1167
## Nagelkerke R2:  0.006412045
## % pres/err predicted correctly:  -437.3415
## % of predictable range [ (model-null)/(1-null) ]:  0.002125627
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.67025     -0.05465     -0.03197      0.33256
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1764  AIC: 1883
## log likelihood:  -881.8111
## Nagelkerke R2:  0.006829321
## % pres/err predicted correctly:  -437.238
## % of predictable range [ (model-null)/(1-null) ]:  0.002361305
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.09205           0.14617          -0.14894           1.35133           0.01412
##      stimlen:pos
##         -0.12602
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1761  AIC: 1884
## log likelihood:  -880.3555
## Nagelkerke R2:  0.008816083
## % pres/err predicted correctly:  -436.9552
## % of predictable range [ (model-null)/(1-null) ]:  0.003005029
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.19685     -0.07088      0.06428
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1769  AIC: 1886
## log likelihood:  -884.4989
## Nagelkerke R2:  0.00315719
## % pres/err predicted correctly:  -437.7649
## % of predictable range [ (model-null)/(1-null) ]:  0.001161809
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.891
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4428 Residual
## Null Deviance:       1774
## Residual Deviance: 1774  AIC: 1886
## log likelihood:  -886.8073
## Nagelkerke R2:  -6.728935e-16
## % pres/err predicted correctly:  -438.2752
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##     3.20465     -0.04068
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1773  AIC: 1886
## log likelihood:  -886.2955
## Nagelkerke R2:  0.0007002624
## % pres/err predicted correctly:  -438.1319
## % of predictable range [ (model-null)/(1-null) ]:  0.0003262443
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.7265       0.0439
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1772  AIC: 1886
## log likelihood:  -885.8618
## Nagelkerke R2:  0.00129358
## % pres/err predicted correctly:  -438.0882
## % of predictable range [ (model-null)/(1-null) ]:  0.0004258722
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	1883.001	0.0000000	1.0000000	0.2342293	0.0057629	2.9271361	NA	0.3410029	NA	-0.0348462	NA
preserved ~ stimlen * pos	1883.097	0.0962260	0.9530261	0.2232267	0.0064120	2.0044523	0.0684741	1.4680428	-0.0471590	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	1883.15	0.156172	0.924888	0.472166	0.100682	2.3670253	-0.0546476	0.3325629	NA	-0.0319727	NA
preserved ~ stimlen * (I(pos^2) + pos)	1884.49	1.497750	0.472898	0.241076	0.100881	1.0920530	0.1461657	0.3513261	-0.1260168	-0.1489375	0.0141222
preserved ~ stimlen + pos	1885.50	2.502200	0.286189	0.146034	0.100315	2.2196854	-0.0708790	0.0642846	NA	NA	NA
preserved ~ 1	1885.65	2.657067	0.264865	0.135203	0.000000	2.0890849	NA	NA	NA	NA	NA
preserved ~ stimlen	1886.29	3.292456	0.192775	0.098451	0.000706	2.3204652	-0.0406791	NA	NA	NA	NA
preserved ~ pos	1886.49	3.489591	0.174680	0.089153	0.001293	2.0726530	NA	0.0439037	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.27136     -0.03485      0.34100
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1765  AIC: 1883
```

```
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
```

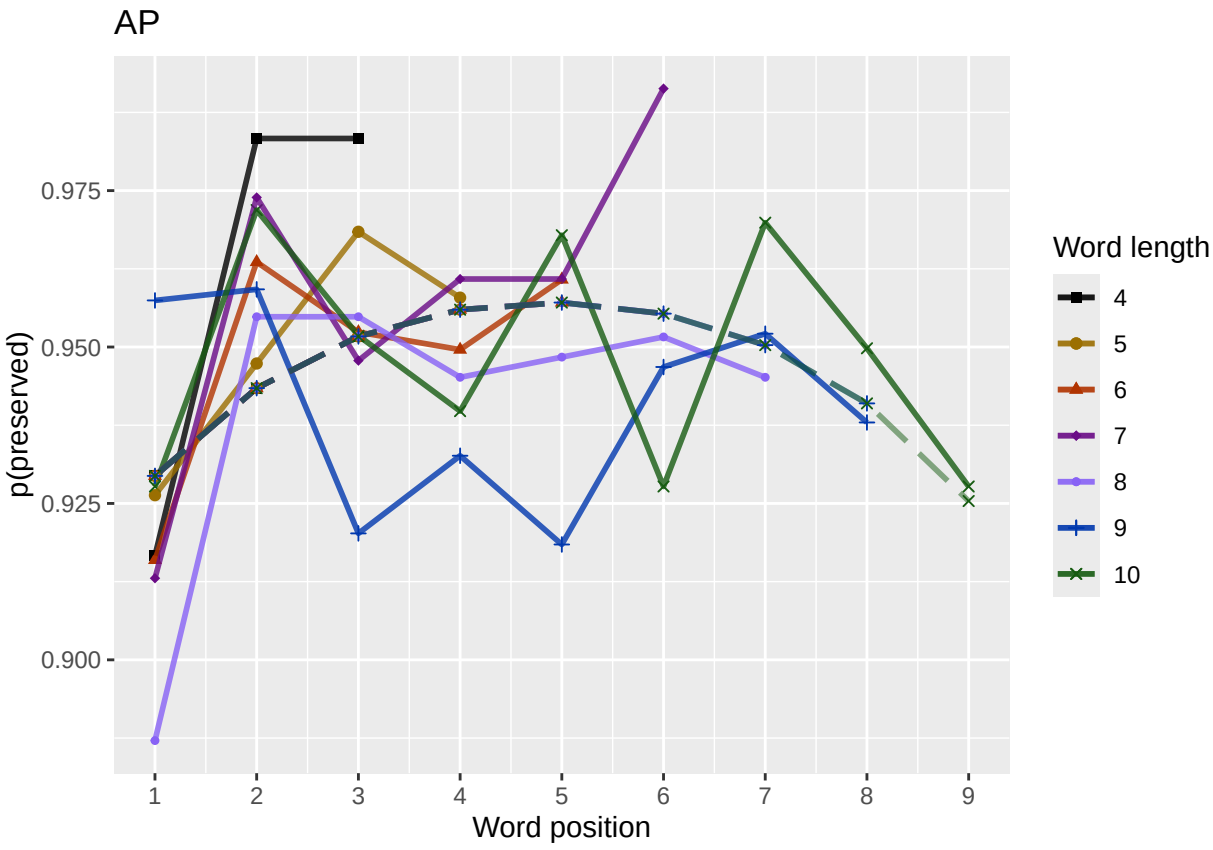
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.929 0.943 0.952 NA     NA     NA     NA     NA     NA
## 2       5 0.929 0.943 0.952  0.956 NA     NA     NA     NA     NA
## 3       6 0.929 0.943 0.952  0.956  0.957 NA     NA     NA     NA
## 4       7 0.929 0.943 0.952  0.956  0.957  0.955 NA     NA     NA
## 5       8 0.929 0.943 0.952  0.956  0.957  0.955  0.950 NA     NA
```

```
## 6       9 0.929 0.943 0.952  0.956  0.957  0.955  0.950  0.941 NA
## 7      10 0.929 0.943 0.952  0.956  0.957  0.955  0.950  0.941  0.925
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        8   721
```

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

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

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

```
## [1] "The number of responses with fragments was 8 / 721 = 1.11 percent"
```

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

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

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

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

```
## ****************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.79189      0.08625      0.62824     -0.06023
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4408 Residual
## Null Deviance:       1679
## Residual Deviance: 1661  AIC: 1776
## log likelihood:  -830.5382
## Nagelkerke R2:  0.01265424
## % pres/err predicted correctly:  -407.4542
## % of predictable range [ (model-null)/(1-null) ]:  0.003906182
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.70928     -0.06905     -0.03142      0.37074
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4408 Residual
## Null Deviance:       1679
## Residual Deviance: 1663  AIC: 1779
## log likelihood:  -831.7335
## Nagelkerke R2:  0.0109485
## % pres/err predicted correctly:  -407.564
## % of predictable range [ (model-null)/(1-null) ]:  0.003638268
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.20346     -0.03505      0.38162
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4409 Residual
## Null Deviance:       1679
## Residual Deviance: 1666  AIC: 1779
## log likelihood:  -832.9321
## Nagelkerke R2:  0.009237201
## % pres/err predicted correctly:  -407.794
## % of predictable range [ (model-null)/(1-null) ]:  0.003077398
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
```

```
##         1.242237          0.131263         -0.090569          1.152146          0.008605
##      stimlen:pos
##        -0.107151
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4406 Residual
## Null Deviance:       1679
## Residual Deviance: 1660  AIC: 1779
## log likelihood:  -829.9814
## Nagelkerke R2:  0.01344851
## % pres/err predicted correctly:  -407.2841
## % of predictable range [ (model-null)/(1-null) ]:  0.004321002
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.1970      -0.0836       0.1125
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4409 Residual
## Null Deviance:       1679
## Residual Deviance: 1668  AIC: 1780
## log likelihood:  -834.0236
## Nagelkerke R2:  0.007678074
## % pres/err predicted correctly:  -407.9949
## % of predictable range [ (model-null)/(1-null) ]:  0.002587537
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.6370       0.0896
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4410 Residual
## Null Deviance:       1679
## Residual Deviance: 1672  AIC: 1782
## log likelihood:  -835.8429
## Nagelkerke R2:  0.005077525
## % pres/err predicted correctly:  -408.3718
## % of predictable range [ (model-null)/(1-null) ]:  0.001668432
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.963
```

```
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4411 Residual
## Null Deviance:       1679
## Residual Deviance: 1679  AIC: 1786
## log likelihood:  -839.3906
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -409.0559
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     3.22137     -0.03349
##
## Degrees of Freedom: 4411 Total (i.e. Null);  4410 Residual
## Null Deviance:       1679
## Residual Deviance: 1678  AIC: 1787
## log likelihood:  -839.0657
## Nagelkerke R2:  0.000465333
## % pres/err predicted correctly:  -408.9671
## % of predictable range [ (model-null)/(1-null) ]:  0.0002165214
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	1776.08	0.000000	1.000000	0.5330038	0.0126542	7.918880	-0.0862451	1.6282356	-0.0602328	NA	NA
preserved ~ stimlen + I(pos^2) + pos	1778.61	2.530110	0.2822238	0.1504264	0.0109485	2.709283	-0.0690487	0.3707373	NA	-0.0314169	NA
preserved ~ I(pos^2) + pos	1779.22	3.139414	0.2081062	0.1109210	0.0092372	2.203459	NA	0.3816241	NA	-0.0350549	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	1779.344	4.263596	0.1955776	0.1042486	0.0134485	2.2422370	0.1312625	1.1521461	-0.1071510	-0.0905692	0.0086053
preserved ~ stimlen + pos	1780.224	5.141202	0.1261100	0.0672171	0.0076781	1.1196955	-0.0836049	0.1125330	NA	NA	NA
preserved ~ pos	1781.995	6.915945	0.0519240	0.0276757	0.0050772	0.5636993	NA	0.0895953	NA	NA	NA
preserved ~ 1	1785.778	8.697298	0.0078390	0.0041782	0.0000000	0.0963368	NA	NA	NA	NA	NA
preserved ~ stimlen	1786.948	10.862259	0.0043781	0.0023336	0.0004653	0.5221367	-0.0334903	NA	NA	NA	NA

```
# plot no fragment data

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

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

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

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

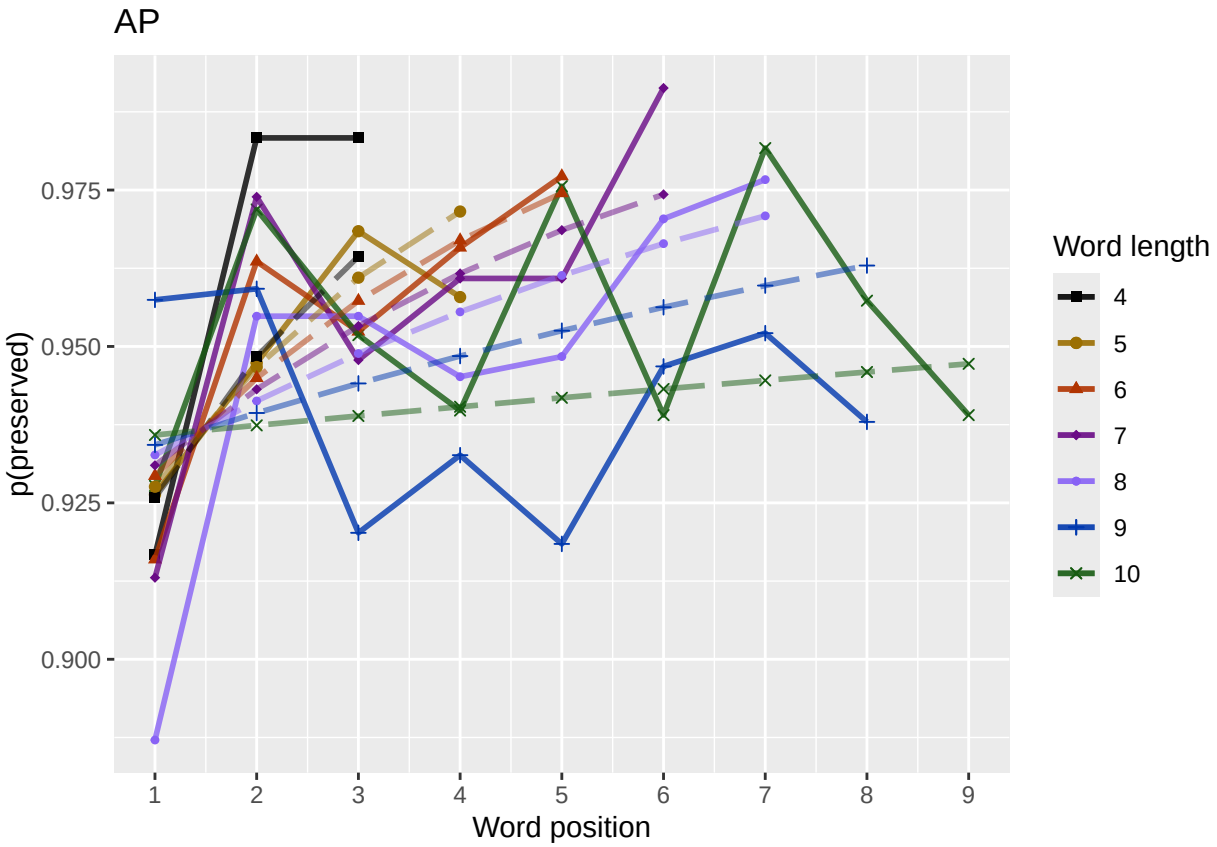
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_fi
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.926 0.948 0.964 NA     NA     NA     NA     NA     NA
## 2       5 0.928 0.947 0.961  0.972 NA     NA     NA     NA     NA
## 3       6 0.929 0.945 0.957  0.967  0.975 NA     NA     NA     NA
## 4       7 0.931 0.943 0.953  0.962  0.969  0.974 NA     NA     NA
## 5       8 0.933 0.941 0.949  0.956  0.961  0.966  0.971 NA     NA
## 6       9 0.934 0.939 0.944  0.948  0.953  0.956  0.960  0.963 NA
## 7      10 0.936 0.937 0.939  0.940  0.942  0.943  0.945  0.946  0.947
```

```
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: 0.88 - 1.00"
```

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

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

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

```
# in case min is close to the end or we are not using a min (for non-U-shaped)
# functions, we need to get differences _first_ and then average those (e.g.
# if there is an effect of length and we just average position data,
# later positions) will have a lower average (because of the length effect)
# even if all position functions go upward

table_pos_diffs <- t(diff(t(as.matrix(table_to_use))))
ncol_pos_diff_matrix <- ncol(table_pos_diffs)
first_col_mean <- mean(as.numeric(unlist(table_pos_diffs[,1])))
potential_u_shape <- 0
if(first_col_mean < 0){ # downward, so potential U-shape
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs >= 0] <- NA
  potential_u_shape <- 1
}else{
  # upward - use the positive values
  # take only negative values from the table
  filtered_table_pos_diffs<-table_pos_diffs
  filtered_table_pos_diffs[table_pos_diffs <= 0] <- NA
}
average_pos_diffs <- apply(filtered_table_pos_diffs,2,mean,na.rm=TRUE)
OA_mean_pos_diff <- mean(average_pos_diffs,na.rm=TRUE)

table_len_diffs <- diff(as.matrix(table_to_use))
average_len_diffs <- apply(table_len_diffs,1,mean,na.rm=TRUE)
OA_mean_len_diff <- mean(average_len_diffs,na.rm=TRUE)
CurrentLabel<-"mean change in probability for each additional length: "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional length: "
```

```
print(OA_mean_len_diff)
```

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

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

CurrentLabel<-"mean change in probability for each additional position (excluding  U-shape): "
print(CurrentLabel)
```

```
## [1] "mean change in probability for each additional position (excluding  U-shape): "
```

```
print(OA_mean_pos_diff)
```

```
## [1] 0.006926761
```

```
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:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##         2.217905          -0.037576           0.381802           0.003075          -0.009111
##     pos:log_freq
##         0.088439
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1744  AIC: 1862
## log likelihood:  -871.7941
## Nagelkerke R2:  0.02047529
## % pres/err predicted correctly:  -435.0271
## % of predictable range [ (model-null)/(1-null) ]:  0.007394369
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    2.311903    -0.004462    -0.032765     0.339269     0.156293
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4424 Residual
## Null Deviance:       1774
```

```
## Residual Deviance: 1746  AIC: 1863
## log likelihood:  -872.8267
## Nagelkerke R2:  0.01907143
## % pres/err predicted correctly:  -435.2375
## % of predictable range [ (model-null)/(1-null) ]:  0.006915347
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.70921      0.05915      0.15994
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1751  AIC: 1864
## log likelihood:  -875.7344
## Nagelkerke R2:  0.01511488
## % pres/err predicted correctly:  -435.7635
## % of predictable range [ (model-null)/(1-null) ]:  0.005717879
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          2.264070          -0.006309          -0.037227           0.380661           0.001178
## I(pos^2):log_freq       pos:log_freq
##         -0.009083           0.088452
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4422 Residual
## Null Deviance:       1774
## Residual Deviance: 1744  AIC: 1864
## log likelihood:  -871.7848
## Nagelkerke R2:  0.02048804
## % pres/err predicted correctly:  -435.0268
## % of predictable range [ (model-null)/(1-null) ]:  0.007394874
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.67588       0.07083       0.10047       0.01658
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1751  AIC: 1865
```

```
## log likelihood:  -875.2605
## Nagelkerke R2:  0.01576006
## % pres/err predicted correctly:  -435.7203
## % of predictable range [ (model-null)/(1-null) ]:  0.005816287
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         2.311926         -0.005775          0.219534         -0.033042          0.341506
## stimlen:log_freq
##        -0.008149
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1746  AIC: 1865
## log likelihood:  -872.7539
## Nagelkerke R2:  0.01917053
## % pres/err predicted correctly:  -435.2292
## % of predictable range [ (model-null)/(1-null) ]:  0.006934251
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.85245     -0.02137      0.06460      0.15527
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1751  AIC: 1866
## log likelihood:  -875.6214
## Nagelkerke R2:  0.01526875
## % pres/err predicted correctly:  -435.7381
## % of predictable range [ (model-null)/(1-null) ]:  0.005775651
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          2.272757          -0.009300           0.087262          -0.036945           0.380911
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.011966          -0.008578           0.087759
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4421 Residual
## Null Deviance:       1774
```

```
## Residual Deviance: 1743  AIC: 1866
## log likelihood:  -871.6538
## Nagelkerke R2:  0.02066599
## % pres/err predicted correctly:  -435.0097
## % of predictable range [ (model-null)/(1-null) ]:  0.00743383
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##    2.860544     0.008938     0.155124
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1755  AIC: 1866
## log likelihood:  -877.4268
## Nagelkerke R2:  0.01280957
## % pres/err predicted correctly:  -436.0994
## % of predictable range [ (model-null)/(1-null) ]:  0.004953243
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.84426      -0.02531       0.07774       0.09215       0.01736
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4424 Residual
## Null Deviance:       1774
## Residual Deviance: 1750  AIC: 1866
## log likelihood:  -875.1031
## Nagelkerke R2:  0.0159743
## % pres/err predicted correctly:  -435.6892
## % of predictable range [ (model-null)/(1-null) ]:  0.005886965
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.855893         -0.022450          0.199595          0.064603         -0.005713
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4424 Residual
## Null Deviance:       1774
## Residual Deviance: 1751  AIC: 1868
## log likelihood:  -875.5864
## Nagelkerke R2:  0.01531643
```

```
## % pres/err predicted correctly:  -435.7257
## % of predictable range [ (model-null)/(1-null) ]:  0.005803904
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.84943          -0.02867           0.19595           0.08046          -0.01545
##     log_freq:pos
##          0.02176
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1750  AIC: 1868
## log likelihood:  -874.8788
## Nagelkerke R2:  0.01627961
## % pres/err predicted correctly:  -435.6458
## % of predictable range [ (model-null)/(1-null) ]:  0.005985846
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         2.864004          0.007859          0.199435         -0.005711
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1755  AIC: 1868
## log likelihood:  -877.3918
## Nagelkerke R2:  0.01285726
## % pres/err predicted correctly:  -436.0867
## % of predictable range [ (model-null)/(1-null) ]:  0.004982058
## **************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.27136     -0.03485      0.34100
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1765  AIC: 1883
## log likelihood:  -882.592
## Nagelkerke R2:  0.005762917
## % pres/err predicted correctly:  -437.4149
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.001958543
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     2.04452      0.06847      0.46804     -0.04716
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1764  AIC: 1883
## log likelihood:  -882.1167
## Nagelkerke R2:  0.006412045
## % pres/err predicted correctly:  -437.3415
## % of predictable range [ (model-null)/(1-null) ]:  0.002125627
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.67025     -0.05465     -0.03197      0.33256
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1764  AIC: 1883
## log likelihood:  -881.8111
## Nagelkerke R2:  0.006829321
## % pres/err predicted correctly:  -437.238
## % of predictable range [ (model-null)/(1-null) ]:  0.002361305
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          1.09205           0.14617          -0.14894           1.35133           0.01412
##      stimlen:pos
##         -0.12602
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1761  AIC: 1884
## log likelihood:  -880.3555
## Nagelkerke R2:  0.008816083
## % pres/err predicted correctly:  -436.9552
## % of predictable range [ (model-null)/(1-null) ]:  0.003005029
```

```
## **************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.19685     -0.07088      0.06428
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1769  AIC: 1886
## log likelihood:  -884.4989
## Nagelkerke R2:  0.00315719
## % pres/err predicted correctly:  -437.7649
## % of predictable range [ (model-null)/(1-null) ]:  0.001161809
## **************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.891
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4428 Residual
## Null Deviance:       1774
## Residual Deviance: 1774  AIC: 1886
## log likelihood:  -886.8073
## Nagelkerke R2:  -6.728935e-16
## % pres/err predicted correctly:  -438.2752
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     3.20465     -0.04068
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1773  AIC: 1886
## log likelihood:  -886.2955
## Nagelkerke R2:  0.0007002624
## % pres/err predicted correctly:  -438.1319
## % of predictable range [ (model-null)/(1-null) ]:  0.0003262443
## **************************
## model index:  16
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.7265       0.0439
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1772  AIC: 1886
## log likelihood:  -885.8618
## Nagelkerke R2:  0.00129358
## % pres/err predicted correctly:  -438.0882
## % of predictable range [ (model-null)/(1-null) ]:  0.0004258722
## ***************************
```

```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	[illegible]	pos	[illegible]	[illegible]	I(pos^2)	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ (I(pos^2) + pos) * log_freq	1862.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0030[illegible]	NA	0.3818[illegible]	0.0284[illegible]	[illegible]	-0.0375[illegible]	-[illegible]91112	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	1863.0[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0044622	0.156[illegible]	NA	0.339[illegible]	NA	NA	-0.0327655	NA	NA	NA	NA
preserved ~ pos + log_freq	1863.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1599[illegible]	NA	0.059[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	1864.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0063086	0.0011[illegible]	NA	0.3806[illegible]	[illegible]	[illegible]	-0.0372[illegible]	-[illegible]90826	NA	NA	NA
preserved ~ pos * log_freq	1864.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.1004[illegible]	NA	0.0708[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA

Model	AIC	Delta_AIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	[illegible]	log_freq:pos	I(pos^2)	[illegible]	I(pos^2):log_freq	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen * log_freq + I(pos^2) + pos	1864.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0057752	0.2195336	-0.0081488	[illegible]	NA	NA	-0.0330419	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	1865.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0213724	[illegible]	NA	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	1866.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0093003	0.0872621	-0.0119662	[illegible]	NA	0.0877593	-0.0369451	NA	-0.0085782	NA	NA
preserved ~ stimlen + log_freq	1866.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	1866.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0253111	[illegible]	NA	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	1867.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0224500	0.1995953	-0.0057130	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	1867.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0286718	0.1959539	-0.0154451	[illegible]	NA	[illegible]	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	1868.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0057107	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1883.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	[illegible]	NA	NA	-0.0348462	NA	NA	NA	NA
preserved ~ stimlen * pos	1883.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	NA	NA	NA	NA	NA	-0.0471590	NA
preserved ~ stimlen + I(pos^2) + pos	1883.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0546476	NA	NA	[illegible]	NA	NA	-0.0319727	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1884.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	[illegible]	NA	NA	-0.1489375	NA	NA	-0.1260168	0.0141222
preserved ~ stimlen + pos	1885.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0708790	NA	NA	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	1885.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	1886.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0406791	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	1886.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	[illegible]	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)          I(pos^2)               pos          log_freq  I(pos^2):log_freq
##          2.217905         -0.037576          0.381802          0.003075          -0.009111
##      pos:log_freq
##          0.088439
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1744  AIC: 1862
```

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

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

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

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

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

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

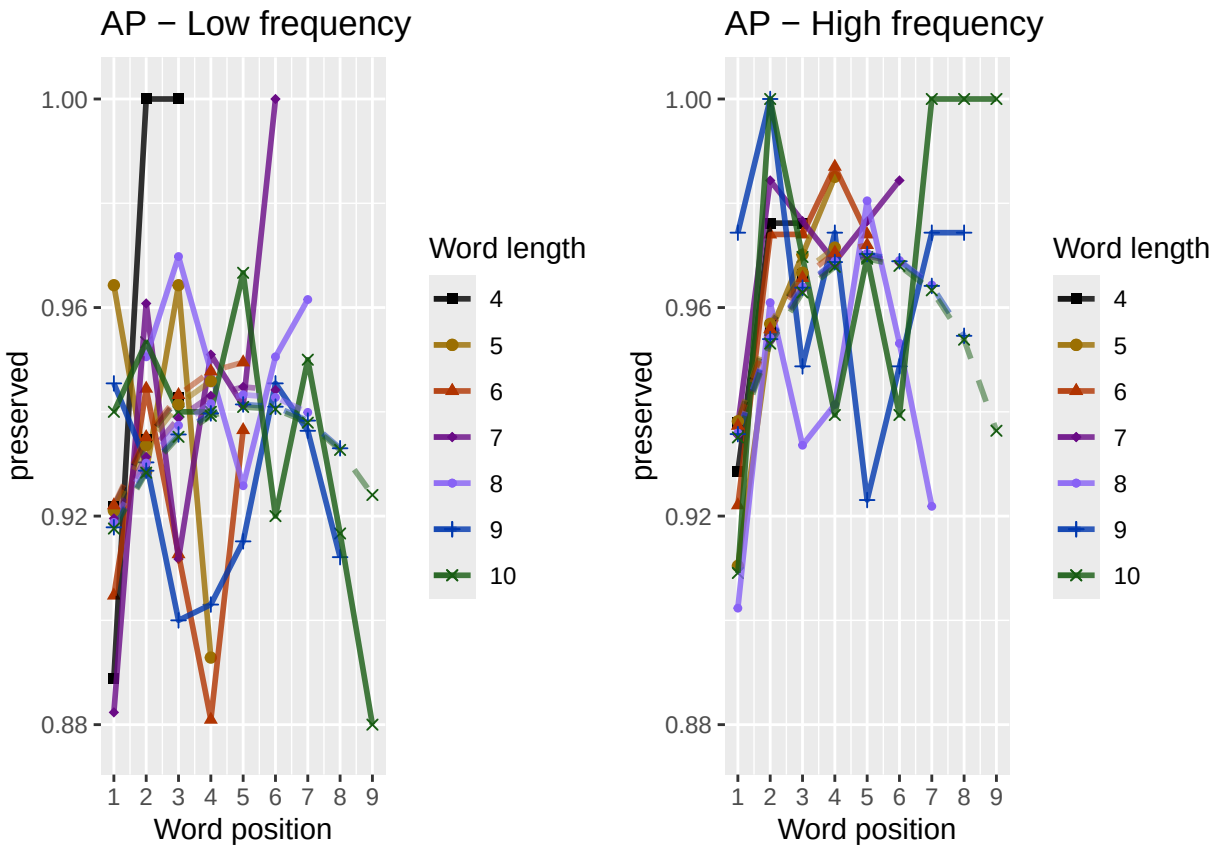
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

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

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

```r
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
##       3.196       -1.027
```

```
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1757
## log likelihood:  -831.2166
## Nagelkerke R2:  0.07512643
## % pres/err predicted correctly:  -413.9801
## % of predictable range [ (model-null)/(1-null) ]:  0.05530734
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.5131       0.1589
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1752  AIC: 1866
## log likelihood:  -876.1936
## Nagelkerke R2:  0.0144895
## % pres/err predicted correctly:  -435.94
## % of predictable range [ (model-null)/(1-null) ]:  0.005316175
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.27136     -0.03485      0.34100
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1765  AIC: 1883
## log likelihood:  -882.592
## Nagelkerke R2:  0.005762917
## % pres/err predicted correctly:  -437.4149
## % of predictable range [ (model-null)/(1-null) ]:  0.001958543
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.891
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4428 Residual
## Null Deviance:       1774
```

```
## Residual Deviance: 1774  AIC: 1886
## log likelihood:  -886.8073
## Nagelkerke R2:  -6.728935e-16
## % pres/err predicted correctly:  -438.2752
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     3.20465     -0.04068
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1773  AIC: 1886
## log likelihood:  -886.2955
## Nagelkerke R2:  0.0007002624
## % pres/err predicted correctly:  -438.1319
## % of predictable range [ (model-null)/(1-null) ]:  0.0003262443
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.7265       0.0439
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1772  AIC: 1886
## log likelihood:  -885.8618
## Nagelkerke R2:  0.00129358
## % pres/err predicted correctly:  -438.0882
## % of predictable range [ (model-null)/(1-null) ]:  0.0004258722
## ***************************
```

```
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	1757.478	0.0000	1	1	0.0751263	3.195941	NA	-1.026839	NA	NA	NA
preserved ~ CumPres	1866.328	108.8504	0	0	0.0144892	2.513149	0.1589193	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1883.001	125.5230	0	0	0.0057629	2.271361	NA	NA	-0.0348462	0.3410029	NA
preserved ~ 1	1885.658	128.1800	0	0	0.0000000	2.890849	NA	NA	NA	NA	NA
preserved ~ stimlen	1886.293	128.8154	0	0	0.0007003	3.204652	NA	NA	NA	NA	-0.0406791
preserved ~ pos	1886.490	129.0126	0	0	0.0012930	2.726530	NA	NA	NA	0.0439037	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	0.9689119	579
O	0.9336593	2045
P	1.0000000	36
S	0.8733850	258
V	0.9691154	1511

```
#  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
##       3.302       -1.192
##
## Degrees of Freedom: 4134 Total (i.e. Null);  4133 Residual
## Null Deviance:       1560
## Residual Deviance: 1442  AIC: 1522
## log likelihood:  -721.2182
## Nagelkerke R2:  0.08890546
## % pres/err predicted correctly:  -353.2898
## % of predictable range [ (model-null)/(1-null) ]:  0.06653638
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      CumPres
##      2.5484       0.1974
##
## Degrees of Freedom: 4134 Total (i.e. Null);  4133 Residual
## Null Deviance:       1560
## Residual Deviance: 1534  AIC: 1633
## log likelihood:  -767.1557
## Nagelkerke R2:  0.01939561
## % pres/err predicted correctly:  -375.9757
## % of predictable range [ (model-null)/(1-null) ]:  0.006764906
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##      2.1620      -0.0475       0.4589
##
## Degrees of Freedom: 4134 Total (i.e. Null);  4132 Residual
## Null Deviance:       1560
## Residual Deviance: 1546  AIC: 1650
## log likelihood:  -773.1127
## Nagelkerke R2:  0.01026818
## % pres/err predicted correctly:  -377.261
## % of predictable range [ (model-null)/(1-null) ]:  0.003378357
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.978
##
## Degrees of Freedom: 4134 Total (i.e. Null);  4134 Residual
## Null Deviance:       1560
## Residual Deviance: 1560  AIC: 1657
## log likelihood:  -779.7937
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -378.5433
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
```

```
##     2.77543      0.05397
##
## Degrees of Freedom: 4134 Total (i.e. Null);  4133 Residual
## Null Deviance:       1560
## Residual Deviance: 1557  AIC: 1658
## log likelihood:  -778.5626
## Nagelkerke R2:  0.001894671
## % pres/err predicted correctly:  -378.3066
## % of predictable range [ (model-null)/(1-null) ]:  0.0006236181
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     3.13563     -0.02054
##
## Degrees of Freedom: 4134 Total (i.e. Null);  4133 Residual
## Null Deviance:       1560
## Residual Deviance: 1559  AIC: 1659
## log likelihood:  -779.6798
## Nagelkerke R2:  0.0001753863
## % pres/err predicted correctly:  -378.5055
## % of predictable range [ (model-null)/(1-null) ]:  9.939848e-05
## ***************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1522.102	0.0000	1	1	0.0889053	3.301816	NA	-1.192296	NA	NA	NA
preserved ~ CumPres	1633.483	111.3809	0	0	0.0193950	0.548435	0.1973672	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1650.437	128.3344	0	0	0.0102682	2.161996	NA	NA	-0.0475043	0.4589168	NA
preserved ~ 1	1657.334	135.2318	0	0	0.0000000	2.977599	NA	NA	NA	NA	NA
preserved ~ pos	1657.955	135.8526	0	0	0.0018947	2.775426	NA	NA	NA	0.0539696	NA
preserved ~ stimlen	1659.018	136.9155	0	0	0.0001753	3.135634	NA	NA	NA	NA	-0.0205405

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

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

```
                            preserved,syll_component)
OVData$CumPres <- CalcCumPres(OVData)
OVData$CumErr <- CalcCumErrFromPreserved(OVData)

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.195       -1.203
##
## Degrees of Freedom: 3555 Total (i.e. Null);  3554 Residual
## Null Deviance:       1400
## Residual Deviance: 1319  AIC: 1390
## log likelihood:  -659.7486
## Nagelkerke R2:  0.06864086
## % pres/err predicted correctly:  -326.2426
## % of predictable range [ (model-null)/(1-null) ]:  0.04971291
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.5769       0.1829
##
## Degrees of Freedom: 3555 Total (i.e. Null);  3554 Residual
## Null Deviance:       1400
## Residual Deviance: 1384  AIC: 1466
## log likelihood:  -692.2273
## Nagelkerke R2:  0.01325527
## % pres/err predicted correctly:  -341.7047
## % of predictable range [ (model-null)/(1-null) ]:  0.004812123
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.09680     -0.04736      0.46468
##
## Degrees of Freedom: 3555 Total (i.e. Null);  3553 Residual
## Null Deviance:       1400
## Residual Deviance: 1386  AIC: 1472
## log likelihood:  -693.108
## Nagelkerke R2:  0.01173934
## % pres/err predicted correctly:  -342.0133
## % of predictable range [ (model-null)/(1-null) ]:  0.003915911
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.67879      0.06468
##
## Degrees of Freedom: 3555 Total (i.e. Null);  3554 Residual
## Null Deviance:       1400
## Residual Deviance: 1396  AIC: 1479
## log likelihood:  -698.2185
## Nagelkerke R2:  0.002927467
## % pres/err predicted correctly:  -343.0178
## % of predictable range [ (model-null)/(1-null) ]:  0.0009990628
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.918
##
## Degrees of Freedom: 3555 Total (i.e. Null);  3555 Residual
## Null Deviance:       1400
## Residual Deviance: 1400  AIC: 1479
## log likelihood:  -699.9131
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -343.3618
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     3.05096     -0.01734
```

```
##
## Degrees of Freedom: 3555 Total (i.e. Null);  3554 Residual
## Null Deviance:       1400
## Residual Deviance: 1400  AIC: 1481
## log likelihood:  -699.8384
## Nagelkerke R2:  0.0001291128
## % pres/err predicted correctly:  -343.3321
## % of predictable range [ (model-null)/(1-null) ]:  8.62362e-05
## ***************************
```

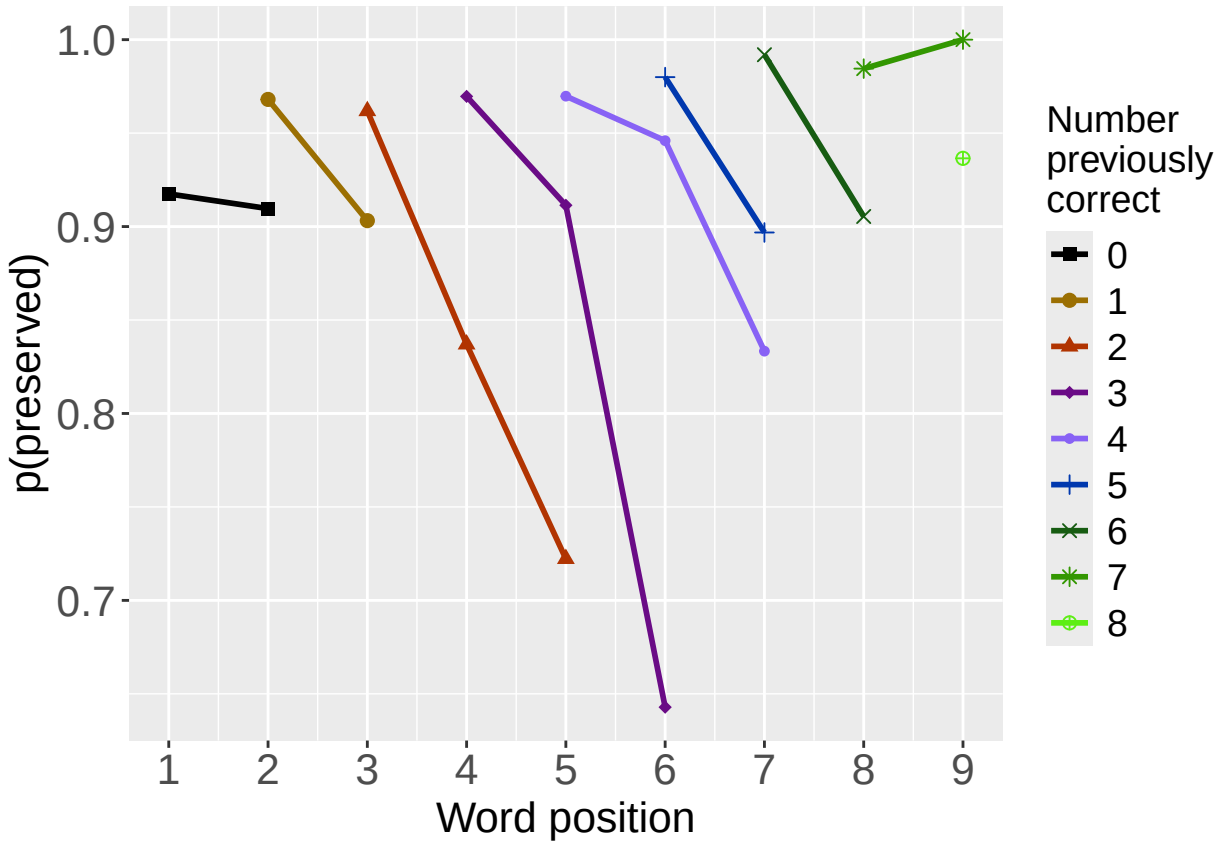
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(posˆ2)	pos	stimlen
preserved ~ CumErr	1390.380	0.00000	1	1	0.0686409	3.194552	NA	-1.202841	NA	NA	NA
preserved ~ CumPres	1466.300	75.91991	0	0	0.0132553	2.576900	0.1828897	NA	NA	NA	NA
preserved ~ (I(posˆ2) + pos)	1471.804	81.42400	0	0	0.0117393	3.096801	NA	NA	-0.0473558	0.4646766	NA
preserved ~ pos	1479.207	88.82654	0	0	0.0029273	2.678794	NA	NA	NA	0.0646800	NA
preserved ~ 1	1479.322	88.94112	0	0	0.0000000	2.917919	NA	NA	NA	NA	NA
preserved ~ stimlen	1481.097	90.71622	0	0	0.0001293	3.050958	NA	NA	NA	NA	-0.0173376

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

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

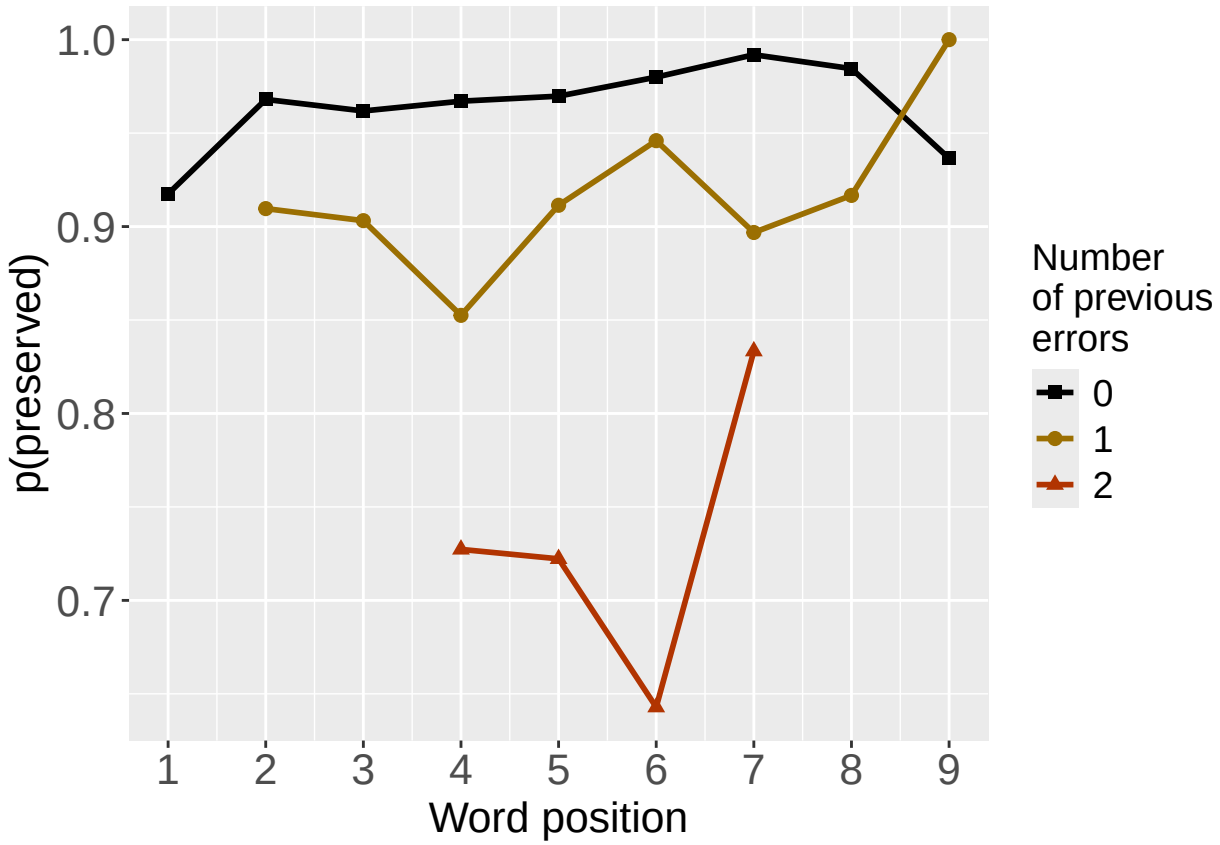
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

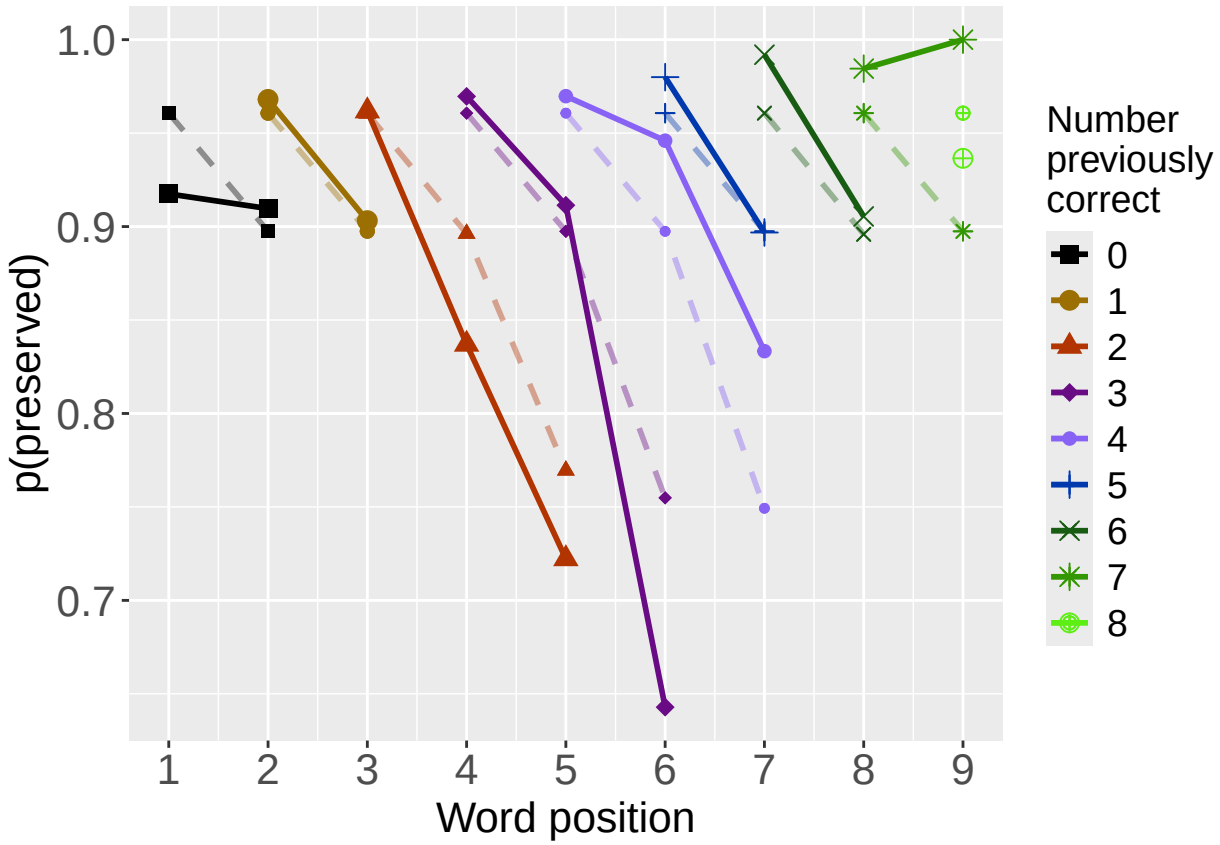
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

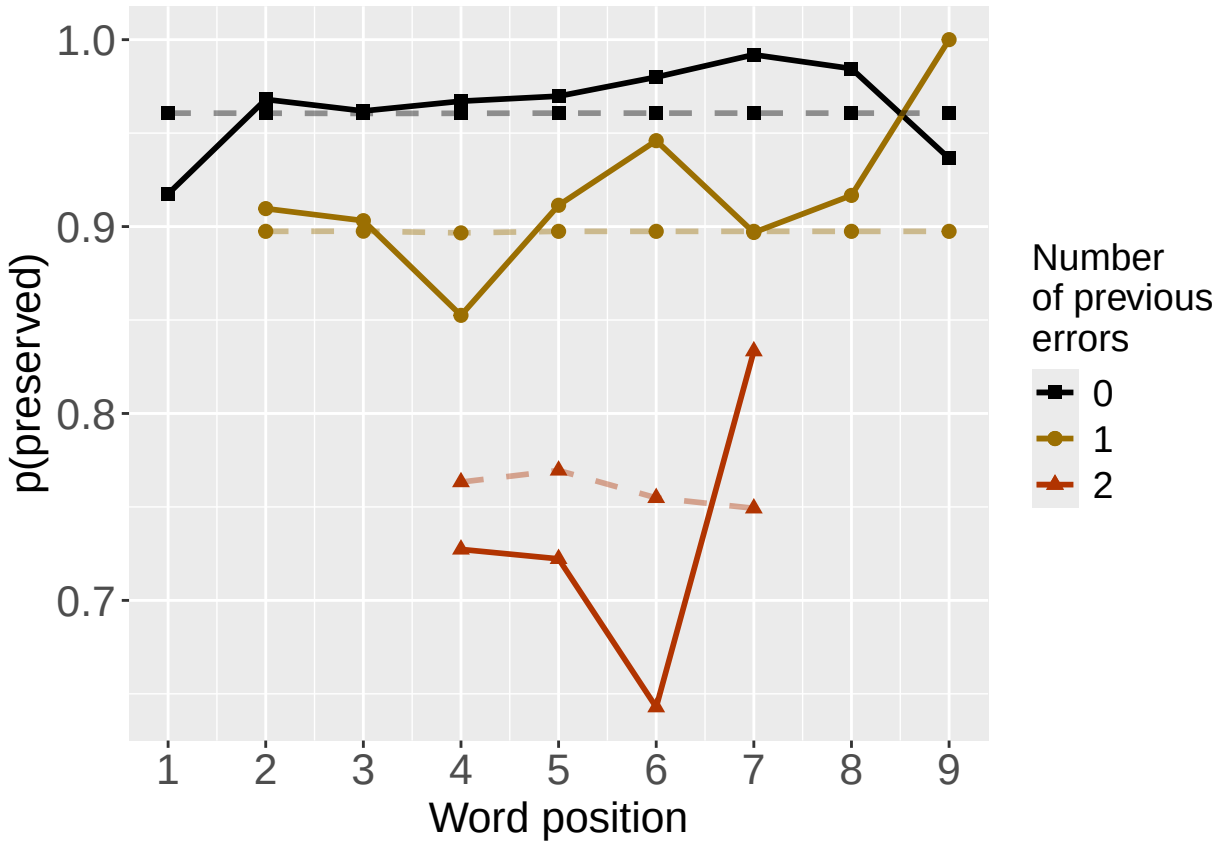
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.13401     -1.25484     -0.03513      0.48213
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1630  AIC: 1729
## log likelihood:  -815.0914
## Nagelkerke R2:  0.0965678
## % pres/err predicted correctly:  -410.5956
## % of predictable range [ (model-null)/(1-null) ]:  0.06301197
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.196       -1.027
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1757
## log likelihood:  -831.2166
## Nagelkerke R2:  0.07512643
## % pres/err predicted correctly:  -413.9801
## % of predictable range [ (model-null)/(1-null) ]:  0.05530734
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.27136     -0.03485      0.34100
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1765  AIC: 1883
## log likelihood:  -882.592
## Nagelkerke R2:  0.005762917
## % pres/err predicted correctly:  -437.4149
## % of predictable range [ (model-null)/(1-null) ]:  0.001958543
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1729.330	0.00000	1e+00	0.9999992	0.0965678	2.134007	-1.254841	-0.0351303	0.4821333

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1757.478	28.14771	8e-07	0.0000008	0.0751264	3.195941	-1.026839	NA	NA
preserved ~ I(posˆ2) + pos	1883.001	153.67068	0e+00	0.0000000	0.0057629	2.271361	NA	-0.0348462	0.3410029

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.196       -1.027
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1757
## log likelihood:  -831.2166
## Nagelkerke R2:  0.07512643
## % pres/err predicted correctly:  -413.9801
## % of predictable range [ (model-null)/(1-null) ]:  0.05530734
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##    3.130254    -1.029434     0.008636
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1759
## log likelihood:  -831.1955
## Nagelkerke R2:  0.07515459
## % pres/err predicted correctly:  -413.9545
## % of predictable range [ (model-null)/(1-null) ]:  0.0553655
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     3.20465     -0.04068
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1773  AIC: 1886
## log likelihood:  -886.2955
## Nagelkerke R2:  0.0007002624
## % pres/err predicted correctly:  -438.1319
## % of predictable range [ (model-null)/(1-null) ]:  0.0003262443
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	1757.478	0.00000	1.0000000	0.729618	0.0751264	3.195941	-1.026839	NA
preserved ~ CumErr + stimlen	1759.463	1.98537	0.3705802	0.270382	0.0751546	3.130254	-1.029434	0.0086356
preserved ~ stimlen	1886.293	128.81543	0.0000000	0.000000	0.0007003	3.204652	NA	-0.0406791

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      2.7595      -1.0633       0.1895
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
```

```
## Null Deviance:       1774
## Residual Deviance: 1636  AIC: 1732
## log likelihood:  -817.9419
## Nagelkerke R2:  0.09278882
## % pres/err predicted correctly:  -410.9008
## % of predictable range [ (model-null)/(1-null) ]:  0.06231717
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.196       -1.027
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1757
## log likelihood:  -831.2166
## Nagelkerke R2:  0.07512643
## % pres/err predicted correctly:  -413.9801
## % of predictable range [ (model-null)/(1-null) ]:  0.05530734
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      2.5131       0.1589
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1752  AIC: 1866
## log likelihood:  -876.1936
## Nagelkerke R2:  0.0144895
## % pres/err predicted correctly:  -435.94
## % of predictable range [ (model-null)/(1-null) ]:  0.005316175
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	1732.196	0.00000	1.0e+00	0.9999968	0.0927888	2.759476	-1.063251	0.1895276
preserved ~ CumErr	1757.478	25.28188	3.2e-06	0.0000032	0.0751264	3.195941	-1.026839	NA
preserved ~ CumPres	1866.328	134.13224	0.0e+00	0.0000000	0.0144895	2.513149	NA	0.1589193

```
########
# 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
##      2.5699      -1.2528       0.1895
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1636  AIC: 1732
## log likelihood:  -817.9419
## Nagelkerke R2:  0.09278882
## % pres/err predicted correctly:  -410.9008
## % of predictable range [ (model-null)/(1-null) ]:  0.06231717
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.196       -1.027
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1757
## log likelihood:  -831.2166
## Nagelkerke R2:  0.07512643
## % pres/err predicted correctly:  -413.9801
## % of predictable range [ (model-null)/(1-null) ]:  0.05530734
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.7265       0.0439
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1772  AIC: 1886
## log likelihood:  -885.8618
## Nagelkerke R2:  0.00129358
## % pres/err predicted correctly:  -438.0882
## % of predictable range [ (model-null)/(1-null) ]:  0.0004258722
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	1732.196	0.00000	1.0e+00	0.9999968	0.0927888	2.569948	-1.252779	0.1895276
preserved ~ CumErr	1757.478	25.28188	3.2e-06	0.0000032	0.0751264	3.195941	-1.026839	NA
preserved ~ pos	1886.490	154.29444	0.0e+00	0.0000000	0.0012936	2.726530	NA	0.0439037

```
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	1729.330	0.00000	1.0000000	0.9999992	0.0965672	2.8134007	-1.254841	-0.0351303	0.4821333	NA	NA
preserved ~ CumErr + CumPres	1732.196	0.00000	1.0000000	0.9999968	0.0927888	2.8759476	-1.063251	NA	NA	NA	0.1895276
preserved ~ CumErr + pos	1732.196	0.00000	1.0000000	0.9999968	0.0927888	2.8569948	-1.252779	NA	0.1895276	NA	NA
preserved ~ CumErr	1757.478	28.14771	0.0000008	0.0000000	0.0751263	4.1195941	-1.026839	NA	NA	NA	NA
preserved ~ CumErr	1757.478	0.00000	1.0000000	0.0729618	0.0751263	4.1195941	-1.026839	NA	NA	NA	NA
preserved ~ CumErr	1757.478	25.28188	0.0000032	0.0000032	0.0751263	4.1195941	-1.026839	NA	NA	NA	NA
preserved ~ CumErr	1757.478	25.28188	0.0000032	0.0000032	0.0751263	4.1195941	-1.026839	NA	NA	NA	NA
preserved ~ CumErr + stimlen	1759.463	1.98537	0.3705802	0.2270382	0.0751546	4.6130254	-1.029434	NA	NA	0.0086356	NA
preserved ~ CumPres	1866.328	134.13224	0.0000000	0.0000000	0.0014489	2.5513149	NA	NA	NA	NA	0.1589193
preserved ~ I(pos^2) + pos	1883.001	153.67068	0.0000000	0.0000000	0.0005762	2.9271361	NA	-0.0348462	0.3410029	NA	NA
preserved ~ stimlen	1886.293	128.81543	0.0000000	0.0000000	0.0000700	3.3204652	NA	NA	NA	-0.0406791	NA
preserved ~ pos	1886.490	154.29444	0.0000000	0.0000000	0.0012936	2.6726530	NA	NA	0.0439037	NA	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.13205     -1.22847     -0.03373      0.48251      0.13015
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4424 Residual
## Null Deviance:       1774
## Residual Deviance: 1618  AIC: 1716
## log likelihood:  -808.8152
## Nagelkerke R2:  0.1048711
## % pres/err predicted correctly:  -410.0033
## % of predictable range [ (model-null)/(1-null) ]:  0.06436031
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     2.30403     -1.23119     -0.03255      0.47885     -0.02345      0.12490
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4423 Residual
## Null Deviance:       1774
## Residual Deviance: 1617  AIC: 1717
## log likelihood:  -808.6895
## Nagelkerke R2:  0.1050371
## % pres/err predicted correctly:  -409.9953
## % of predictable range [ (model-null)/(1-null) ]:  0.06437864
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     2.60819     -1.26008     -0.03166      0.47218     -0.06484
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4424 Residual
## Null Deviance:       1774
## Residual Deviance: 1628  AIC: 1729
## log likelihood:  -814.0398
## Nagelkerke R2:  0.09796062
## % pres/err predicted correctly:  -410.5211
## % of predictable range [ (model-null)/(1-null) ]:  0.06318161
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.13401     -1.25484     -0.03513      0.48213
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1630  AIC: 1729
## log likelihood:  -815.0914
## Nagelkerke R2:  0.0965678
## % pres/err predicted correctly:  -410.5956
## % of predictable range [ (model-null)/(1-null) ]:  0.06301197
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       2.891
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4428 Residual
## Null Deviance:       1774
## Residual Deviance: 1774  AIC: 1886
## log likelihood:  -886.8073
## Nagelkerke R2:  -6.728935e-16
## % pres/err predicted correctly:  -438.2752
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + log_freq	1715.669	0.0000000	1.0000000	0.7019202	0.2104872	2.1132055	-1.2284688	-0.0337298	0.4825105	0.1301494	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	1717.394	1.7246499	0.4221795	0.2963376	0.2105032	2.1304032	-1.2311870	-0.0325536	0.4788507	0.1248961	-0.0234514
preserved ~ CumErr + I(pos^2) + pos + stimlen	1728.815	13.1462890	0.0013974	0.0009809	0.0979626	2.0608192	-1.2600820	-0.0316618	0.4721751	NA	-0.0648436
preserved ~ CumErr + I(pos^2) + pos	1729.330	13.6609360	0.0010804	0.0007583	0.0965672	2.0134007	-1.2548410	-0.0351303	0.4821338	NA	NA
preserved ~ 1	1885.658	169.9886850	0.0000000	0.0000000	0.0000000	0.2890849	NA	NA	NA	NA	NA

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

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

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

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

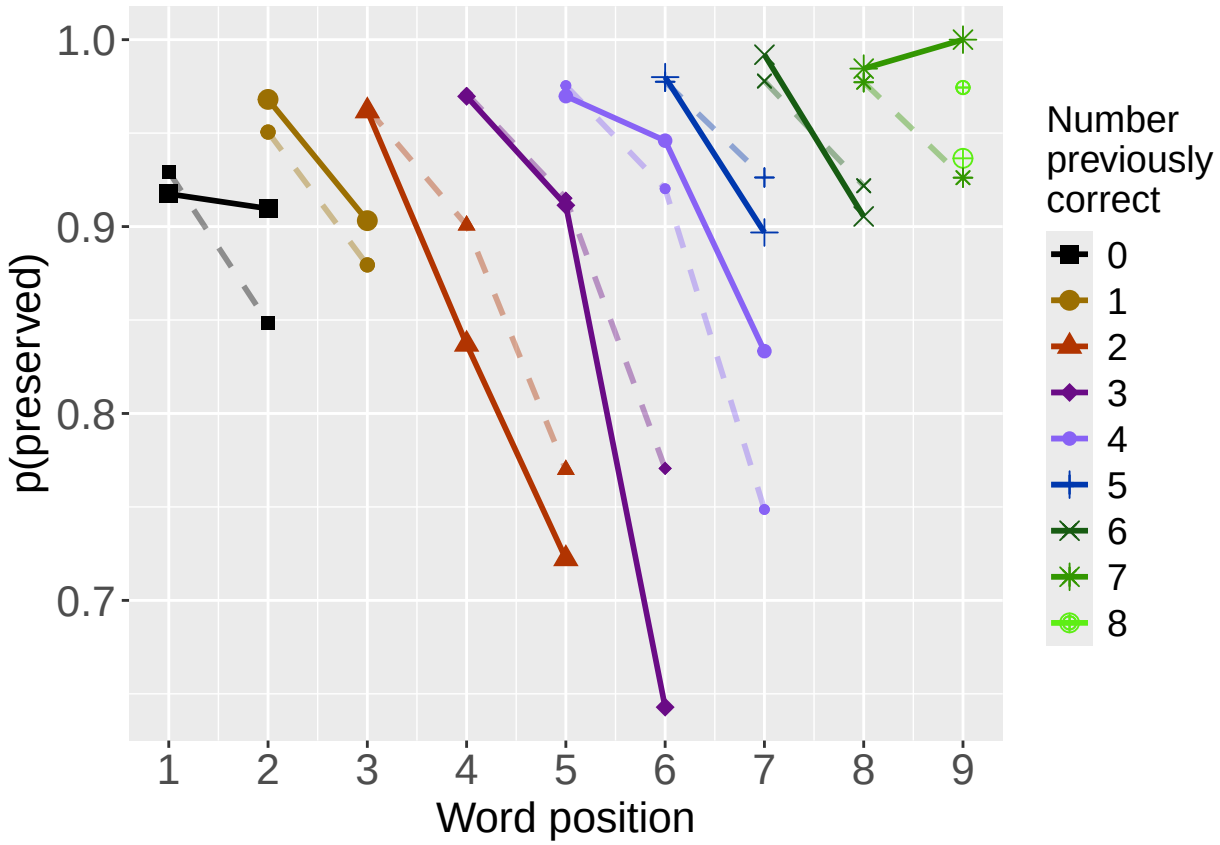
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## CumErr    1   1745.7 1841.7
## pos       1   1631.5 1727.5
## log_freq  1   1630.2 1726.2
## I(pos^2)  1   1622.8 1718.9
## <none>        1617.6 1715.7
```

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

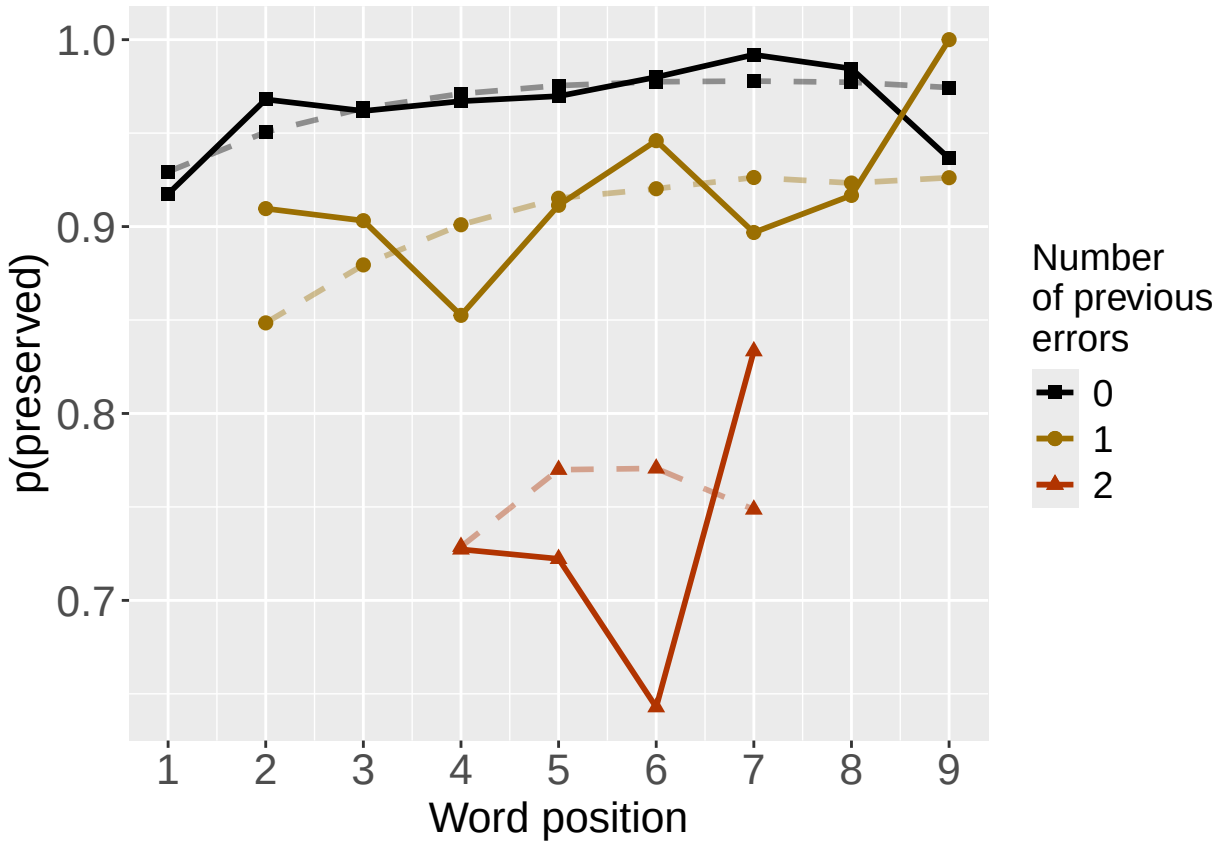
```
print(PrevCorPlot)
```




```
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!
```

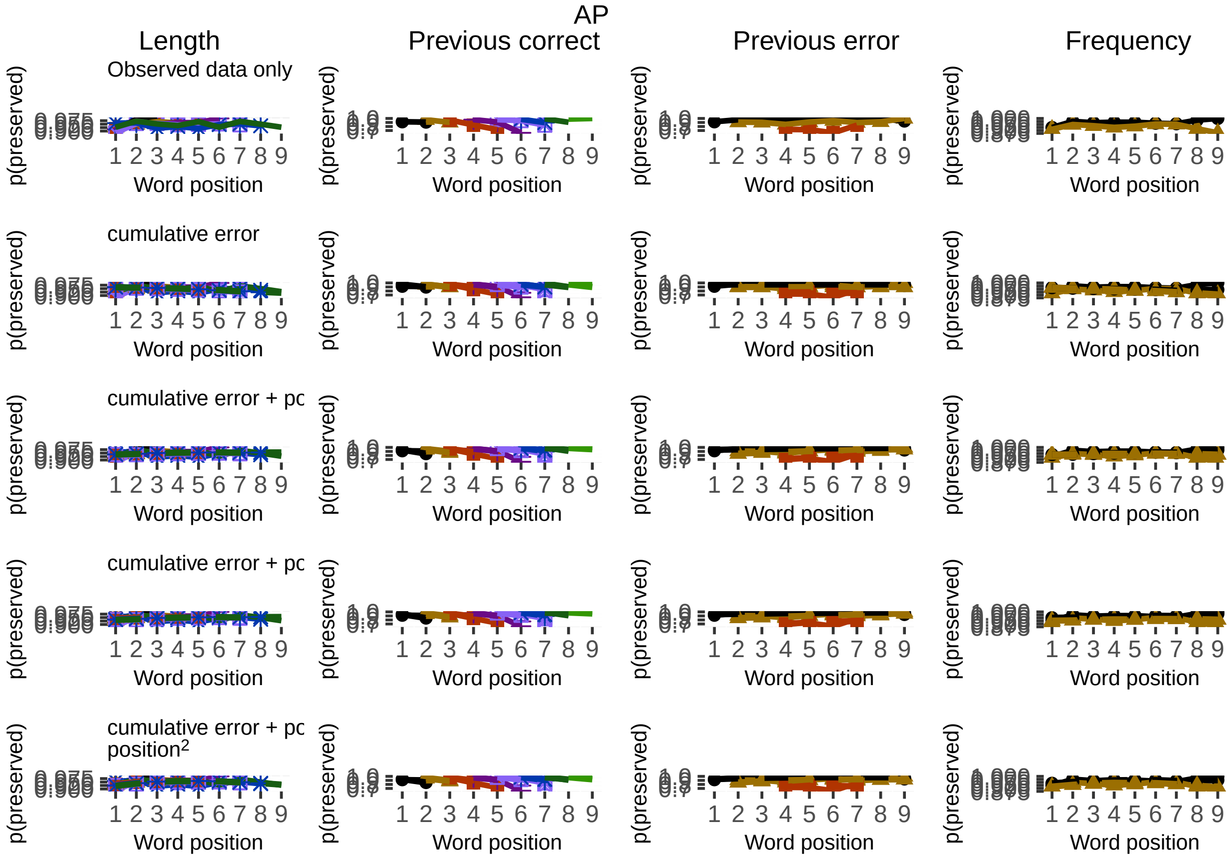
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       3.196       -1.027
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4427 Residual
## Null Deviance:       1774
## Residual Deviance: 1662  AIC: 1757
## log likelihood:  -831.2166
```

```
## Nagelkerke R2:  0.07512643
## % pres/err predicted correctly:  -413.9801
## % of predictable range [ (model-null)/(1-null) ]:  0.05530734
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      2.5699      -1.2528       0.1895
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4426 Residual
## Null Deviance:       1774
## Residual Deviance: 1636  AIC: 1732
## log likelihood:  -817.9419
## Nagelkerke R2:  0.09278882
## % pres/err predicted correctly:  -410.9008
## % of predictable range [ (model-null)/(1-null) ]:  0.06231717
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq
##      2.5495      -1.2258       0.2020       0.1324
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4425 Residual
## Null Deviance:       1774
## Residual Deviance: 1623  AIC: 1718
## log likelihood:  -811.4233
## Nagelkerke R2:  0.1014235
## % pres/err predicted correctly:  -410.227
## % of predictable range [ (model-null)/(1-null) ]:  0.06385111
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     log_freq     I(pos^2)
##     2.13205     -1.22847      0.48251      0.13015     -0.03373
##
## Degrees of Freedom: 4428 Total (i.e. Null);  4424 Residual
## Null Deviance:       1774
## Residual Deviance: 1618  AIC: 1716
## log likelihood:  -808.8152
## Nagelkerke R2:  0.1048711
## % pres/err predicted correctly:  -410.0033
## % of predictable range [ (model-null)/(1-null) ]:  0.06436031
```

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

## Warning: Removed 5 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 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 9 values. Consider specifying shapes manually if you need that many have
##   them.

## Warning: Removed 5 rows containing missing values or values outside the scale range (`geom_point()`)
## Removed 5 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 9 values. Consider specifying shapes manually if you need that many have
##   them.

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

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




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

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

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

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

	CumErr	I(pos^2)	pos	log_freq
McFadden	0.0753062	0.0031847	0.0056872	0.0103129
SquaredCorrelation	0.0314141	0.0013149	0.0023450	0.0043164
Nagelkerke	0.0314141	0.0013149	0.0023450	0.0043164
Estrella	0.0329076	0.0014127	0.0025275	0.0044882

```
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 + log_freq + I(pos^2) CumErr + pos + log_freq + I(pos^2) 1617.630
## CumErr + pos + log_freq                       CumErr + pos + log_freq 1622.847
## CumErr + pos                                             CumErr + pos 1635.884
## CumErr                                                         CumErr 1662.433
## null                                                             null 1773.615
##                                    deviance_explained percent_explained
## CumErr + pos + log_freq + I(pos^2)           155.9842          8.794705
## CumErr + pos + log_freq                      150.7680          8.500607
## CumErr + pos                                 137.7306          7.765533
## CumErr                                       111.1814          6.268632
## null                                           0.0000          0.000000
##                                    percent_of_explained_deviance increment_in_explained
## CumErr + pos + log_freq + I(pos^2)                     100.00000               3.344039
## CumErr + pos + log_freq                                 96.65596               8.358144
## CumErr + pos                                            88.29782              17.020476
## CumErr                                                  71.27734              71.277341
## 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 + log_freq + I(pos^2)	1617.630	155.9842
CumErr + pos + log_freq	1622.847	150.7680
CumErr + pos	1635.884	137.7306
CumErr	1662.433	111.1814
null	1773.615	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + log_freq + I(pos^2)	8.794705	100.00000	3.344039
CumErr + pos + log_freq	8.500607	96.65596	8.358144
CumErr + pos	7.765533	88.29782	17.020476
CumErr	6.268632	71.27734	71.277341
null	0.000000	NA	0.000000

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

```
##          Nagelkerke
## CumErr   0.79750786
## I(pos^2) 0.03338019
## pos      0.05953230
## log_freq 0.10957965
```

```
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	0.8177851	1662.433
preserved ~ CumErr+pos+log_freq	0.8579462	1622.847
preserved ~ CumErr+pos	0.8625122	1635.884
preserved ~ CumErr+pos+log_freq+I(pos^2)	0.8910590	1617.630

```
## 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       0.8177851       1662.433  0.00000000
## 2          preserved ~ CumErr+pos+log_freq       0.8579462       1622.847  0.04016117
## 3                   preserved ~ CumErr+pos       0.8625122       1635.884  0.04472716
## 4 preserved ~ CumErr+pos+log_freq+I(pos^2)       0.8910590       1617.630  0.07327393
##   diff_CumErr+pos+log_freq diff_CumErr+pos diff_CumErr+pos+log_freq+I(pos^2)
## 1              -0.04016117     -0.04472716                       -0.07327393
## 2               0.00000000     -0.00456599                       -0.03311276
## 3               0.00456599      0.00000000                       -0.02854677
## 4               0.03311276      0.02854677                        0.00000000
```

```
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+log_freq	diff_CumErr+pos
preserved ~ CumErr	0.0000000	-0.0401612	-0.0447272
preserved ~ CumErr+pos+log_freq	0.0401612	0.0000000	-0.0045660
preserved ~ CumErr+pos	0.0447272	0.0045660	0.0000000
preserved ~ CumErr+pos+log_freq+I(pos^2)	0.0732739	0.0331128	0.0285468