AG - repetition - Serial position analysis (v5)

```r
# 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
```

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

```r
# 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
```

```r
# 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())
```

```r
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)
```

```r
# 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)
```

```r
# 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	509	29	120	NA	NA	658
2	58	NA	405	91	104	658
3	291	NA	157	197	13	658
4	284	NA	224	62	37	607
5	216	NA	203	68	37	524
6	201	1	131	64	22	419
7	166	NA	102	25	17	310
8	86	NA	48	26	4	164
9	69	NA	2	NA	6	77

```r
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7735562	0.0440729	0.1823708	NA	NA	658
2	0.0881459	NA	0.6155015	0.1382979	0.1580547	658
3	0.4422492	NA	0.2386018	0.2993921	0.0197568	658
4	0.4678748	NA	0.3690280	0.1021417	0.0609555	607
5	0.4122137	NA	0.3874046	0.1297710	0.0706107	524
6	0.4797136	0.0023866	0.3126492	0.1527446	0.0525060	419

pos_factor	O	P	V	1	S	total
7	0.5354839	NA	0.3290323	0.0806452	0.0548387	310
8	0.5243902	NA	0.2926829	0.1585366	0.0243902	164
9	0.8961039	NA	0.0259740	NA	0.0779221	77

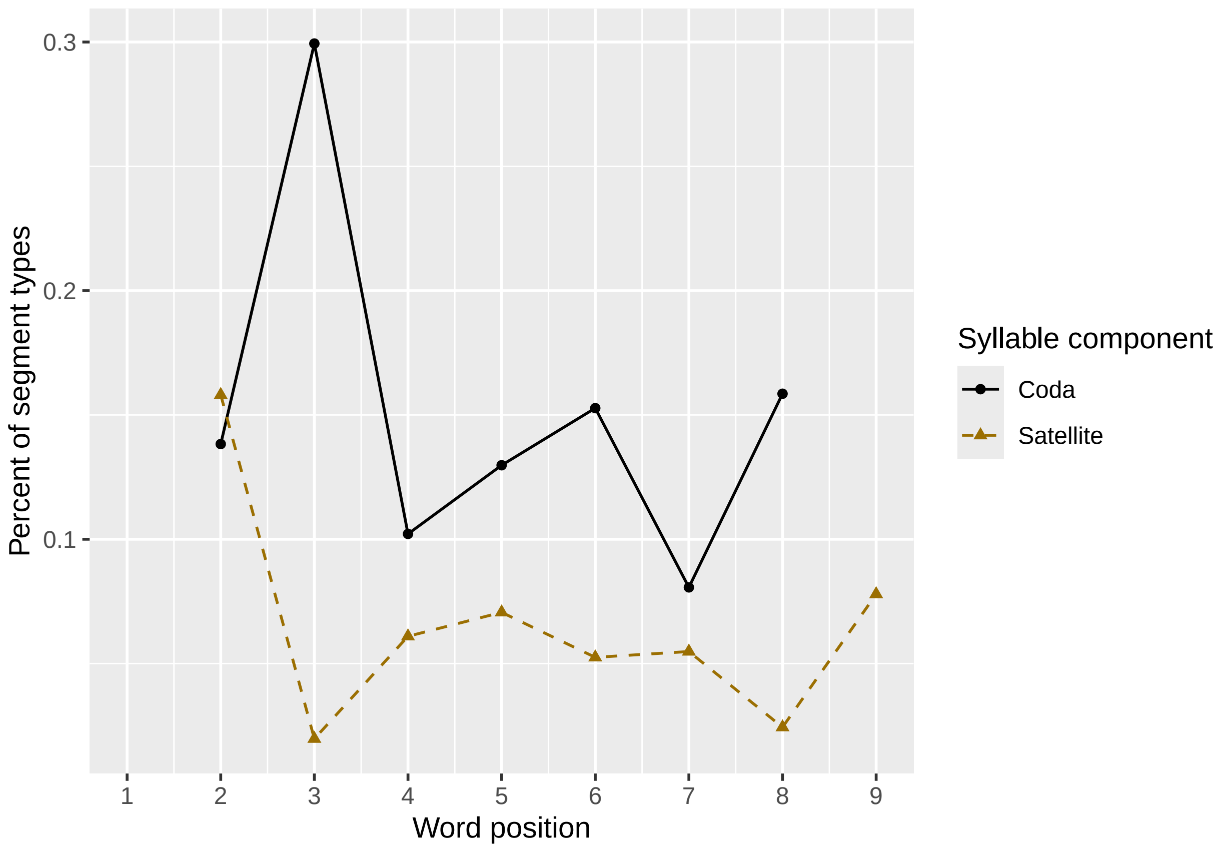
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 0.961 0.941 1     NA     NA     NA     NA     NA     NA
## 2       5 0.916 0.940 0.855  0.916 NA     NA     NA     NA     NA
## 3       6 0.933 0.938 0.914  0.924  0.876 NA     NA     NA     NA
## 4       7 0.927 0.954 0.927  0.945  0.954  0.862 NA     NA     NA
## 5       8 0.918 0.960 0.933  0.900  0.873  0.923  0.942 NA     NA
## 6       9 0.891 0.943 0.897  0.948  0.897  0.897  0.868  0.874 NA
## 7      10 0.922 0.974 0.867  0.867  0.925  0.933  0.915  0.903  0.922
```

```
# 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.
```

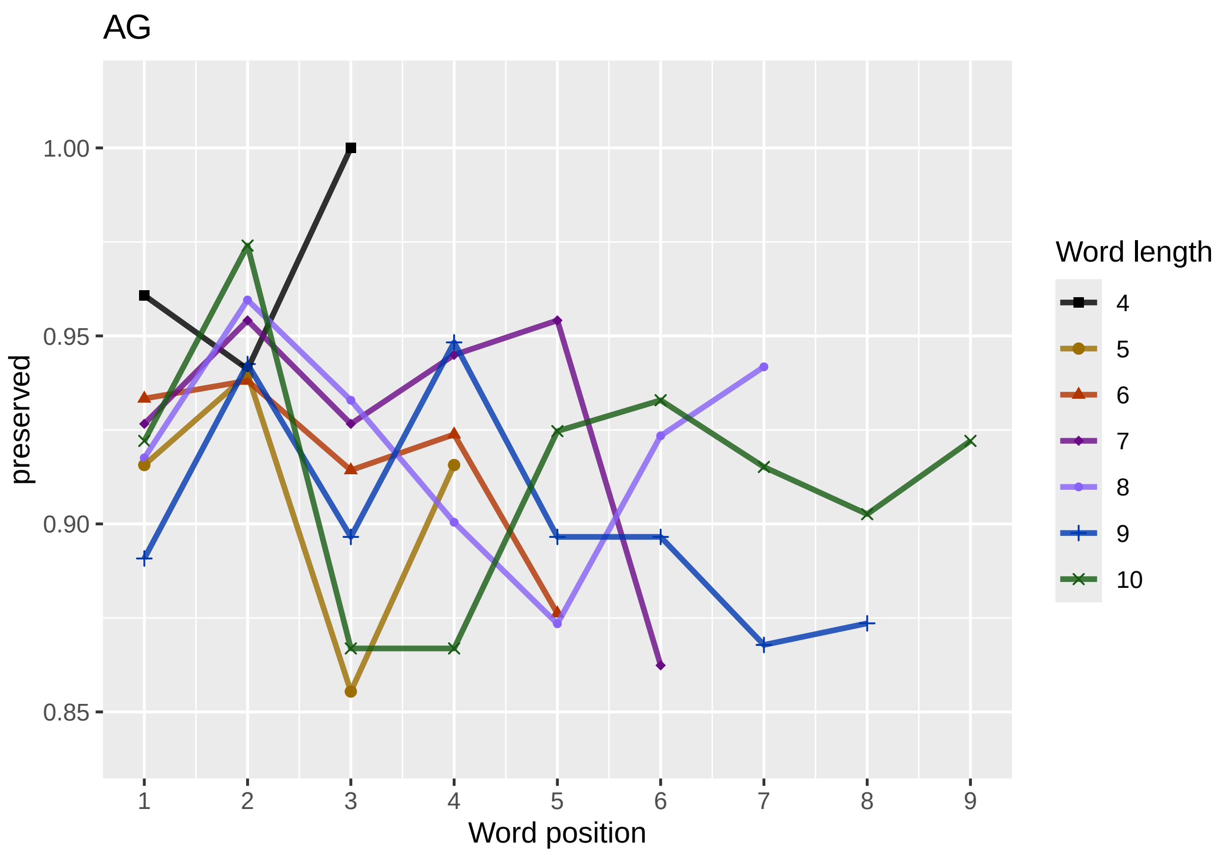
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    51    51    51    NA    NA    NA    NA    NA    NA
## 2       5    83    83    83    83    NA    NA    NA    NA    NA
## 3       6   105   105   105   105   105    NA    NA    NA    NA
## 4       7   109   109   109   109   109   109    NA    NA    NA
## 5       8   146   146   146   146   146   146   146    NA    NA
## 6       9    87    87    87    87    87    87    87    87    NA
## 7      10    77    77    77    77    77    77    77    77    77
```

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

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

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



Length and position

```
# length and position

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

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

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

## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.65773     -0.06169
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2391
## log likelihood:  -1120.213
## Nagelkerke R2:  0.003185316
## % pres/err predicted correctly:  -608.4538
## % of predictable range [ (model-null)/(1-null) ]:  0.001469611
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.775973     0.007979    -0.132372
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2392
## log likelihood:  -1119.982
## Nagelkerke R2:  0.003453151
## % pres/err predicted correctly:  -608.3341
## % of predictable range [ (model-null)/(1-null) ]:  0.001665846
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.79181     -0.02058     -0.05526
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2392
## log likelihood:  -1120.067
## Nagelkerke R2:  0.003353801
## % pres/err predicted correctly:  -608.3648
## % of predictable range [ (model-null)/(1-null) ]:  0.001615491
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.37196     -0.09078     -0.23464      0.02105
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2239  AIC: 2393
## log likelihood:  -1119.417
## Nagelkerke R2:  0.004106173
## % pres/err predicted correctly:  -608.1477
## % of predictable range [ (model-null)/(1-null) ]:  0.001971227
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.964870    -0.025849     0.009353    -0.136508
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2393
## log likelihood:  -1119.758
## Nagelkerke R2:  0.003711289
## % pres/err predicted correctly:  -608.2016
## % of predictable range [ (model-null)/(1-null) ]:  0.001882812
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.79711     -0.04954
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2244  AIC: 2394
## log likelihood:  -1121.94
## Nagelkerke R2:  0.00118733
## % pres/err predicted correctly:  -608.906
## % of predictable range [ (model-null)/(1-null) ]:  0.0007288599
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.413
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4074 Residual
## Null Deviance:       2246
## Residual Deviance: 2246  AIC: 2395
## log likelihood:  -1122.965
## Nagelkerke R2:  2.620228e-16
## % pres/err predicted correctly:  -609.3508
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.919008         -0.034529         -0.039542          0.069241          0.004649
##      stimlen:pos
##        -0.015637
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4069 Residual
## Null Deviance:       2246
## Residual Deviance: 2239  AIC: 2396
## log likelihood:  -1119.267
## Nagelkerke R2:  0.004279799
## % pres/err predicted correctly:  -608.0814
## % of predictable range [ (model-null)/(1-null) ]:  0.002079854
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ pos	2391.202	0.000000	1.0000000	0.2849845	0.0031852	2.3657733	NA	-0.0616854	NA	NA	NA
preserved ~ I(pos^2) + pos	2392.213	1.011338	0.6031001	0.1718747	0.0034532	2.2775973	NA	-0.1323722	NA	0.0079788	NA
preserved ~ stimlen + pos	2392.457	1.255184	0.5338750	0.1521464	0.0033532	2.8791806	-0.0205806	-0.0552584	NA	NA	NA

Model	AIC	DeltaAIC	ExpAIC	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	2392.745	1.543131	0.4622889	0.1317452	0.0041063	2.371956	-0.0907800	-0.2346360	0.0210517	NA	NA
preserved ~ stimlen + I(pos^2) + pos	2393.101	1.904875	0.3857994	0.1099469	0.0037112	2.3964870	-0.0258491	-0.1365082	NA	0.0093531	NA
preserved ~ stimlen	2393.692	2.493367	0.2874566	0.0819207	0.0011872	2.3797110	-0.0495376	NA	NA	NA	NA
preserved ~ 1	2394.873	3.677023	0.1590540	0.0453279	0.0000000	2.0412943	NA	NA	NA	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	2396.325	5.117904	0.0773858	0.0220538	0.0042792	2.8919008	-0.0345294	0.0692413	-0.0156366	-0.0395425	0.0046485

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.65773     -0.06169
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2391
```

```r
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.
```

```r
# 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.
```

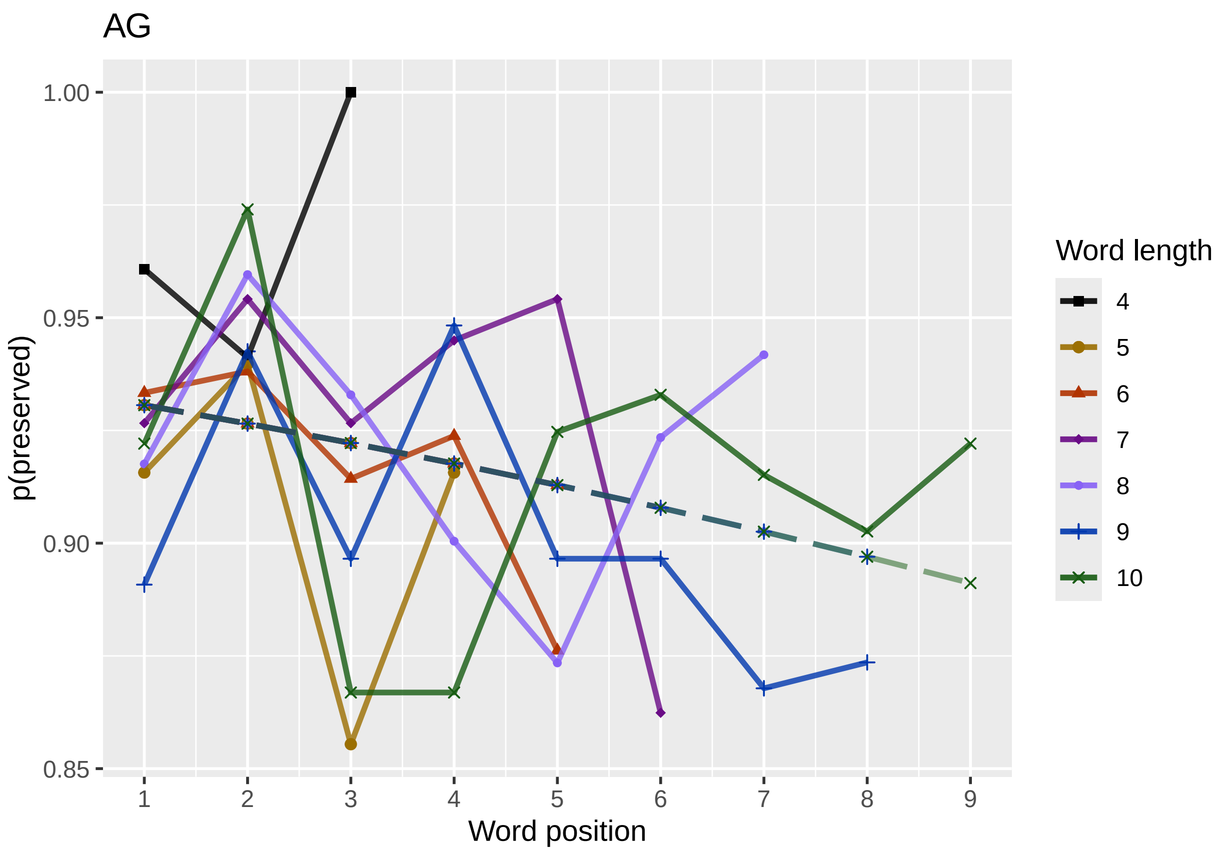
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # 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 0.931 0.927 0.922 NA     NA     NA     NA     NA     NA
## 2       5 0.931 0.927 0.922  0.918 NA     NA     NA     NA     NA
## 3       6 0.931 0.927 0.922  0.918  0.913 NA     NA     NA     NA
## 4       7 0.931 0.927 0.922  0.918  0.913  0.908 NA     NA     NA
## 5       8 0.931 0.927 0.922  0.918  0.913  0.908  0.903 NA     NA
## 6       9 0.931 0.927 0.922  0.918  0.913  0.908  0.903  0.897 NA
## 7      10 0.931 0.927 0.922  0.918  0.913  0.908  0.903  0.897  0.891
```

```
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       11   658
```

```
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 11 / 658 = 1.67 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:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.85364     -0.04729
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4048 Residual
## Null Deviance:       2127
## Residual Deviance: 2125  AIC: 2270
## log likelihood:  -1062.616
## Nagelkerke R2:  0.001058074
## % pres/err predicted correctly:  -568.7373
## % of predictable range [ (model-null)/(1-null) ]:  0.0006580808
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.487
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4049 Residual
## Null Deviance:       2127
## Residual Deviance: 2127  AIC: 2271
## log likelihood:  -1063.492
## Nagelkerke R2:  5.434919e-16
## % pres/err predicted correctly:  -569.1125
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.60903     -0.03131
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4048 Residual
## Null Deviance:       2127
## Residual Deviance: 2126  AIC: 2271
## log likelihood:  -1062.84
## Nagelkerke R2:  0.0007880719
## % pres/err predicted correctly:  -568.8972
## % of predictable range [ (model-null)/(1-null) ]:  0.0003775862
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.85300     -0.03712     -0.02006
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4047 Residual
## Null Deviance:       2127
## Residual Deviance: 2125  AIC: 2271
## log likelihood:  -1062.386
## Nagelkerke R2:  0.001336534
## % pres/err predicted correctly:  -568.682
## % of predictable range [ (model-null)/(1-null) ]:  0.0007550101
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.04375     -0.04286      0.01071     -0.11180
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4046 Residual
## Null Deviance:       2127
## Residual Deviance: 2124  AIC: 2272
## log likelihood:  -1062.016
## Nagelkerke R2:  0.001783075
## % pres/err predicted correctly:  -568.5002
## % of predictable range [ (model-null)/(1-null) ]:  0.001073896
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.729232     0.008424    -0.104782
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4047 Residual
## Null Deviance:       2127
## Residual Deviance: 2125  AIC: 2272
## log likelihood:  -1062.605
## Nagelkerke R2:  0.001071922
## % pres/err predicted correctly:  -568.7806
## % of predictable range [ (model-null)/(1-null) ]:  0.0005821197
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.17788     -0.07640     -0.12346      0.01213
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4046 Residual
## Null Deviance:       2127
## Residual Deviance: 2124  AIC: 2273
## log likelihood:  -1062.185
## Nagelkerke R2:  0.001578875
## % pres/err predicted correctly:  -568.6144
## % of predictable range [ (model-null)/(1-null) ]:  0.0008737435
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen          I(pos^2)              pos  stimlen:I(pos^2)
##       3.2006628       -0.0637711         0.0174987       -0.1901135        -0.0009152
##      stimlen:pos
##       0.0104027
##
## Degrees of Freedom: 4049 Total (i.e. Null);  4044 Residual
## Null Deviance:       2127
## Residual Deviance: 2124  AIC: 2276
## log likelihood:  -1062
## Nagelkerke R2:  0.001803004
## % pres/err predicted correctly:  -568.5035
## % of predictable range [ (model-null)/(1-null) ]:  0.001068226
## ***************************
```

```
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=FALS
NoFragLPAICSummary <- subset(NoFragLPAICSummary, select = -c(Row.names))

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen	2269.707	0.0000000	1.0000000	0.2867835	0.0010581	2.1853642	-0.0472913	NA	NA	NA	NA
preserved ~ 1	2270.568	0.8609691	0.6501940	0.1864649	0.0000000	2.0487029	NA	NA	NA	NA	NA
preserved ~ pos	2271.142	1.4348404	0.4880096	0.1399531	0.0007882	2.1609035	NA	-0.0313063	NA	NA	NA
preserved ~ stimlen + pos	2271.406	1.6993610	0.4275510	0.1226140	0.0013362	2.5853001	-0.0371248	-0.0200598	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos	2271.803	2.0963249	0.3505801	0.1005410	0.0017831	3.1043748	-0.0428570	-0.1117964	NA	0.0107091	NA
preserved ~ I(pos^2) + pos	2272.093	2.3862560	0.3032701	0.0869732	0.0010719	2.9729232	NA	-0.1047817	NA	0.0084244	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	2272.736	3.0292429	0.2198910	0.0630610	0.0015789	3.177884	-0.0764029	-0.1234646	0.0121273	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	2275.803	6.0960883	0.0474510	0.0136083	0.0018030	3.200663	-0.0637710	-0.1901135	0.0104027	0.0174987	-0.0009152

```
# plot no fragment data

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

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

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

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

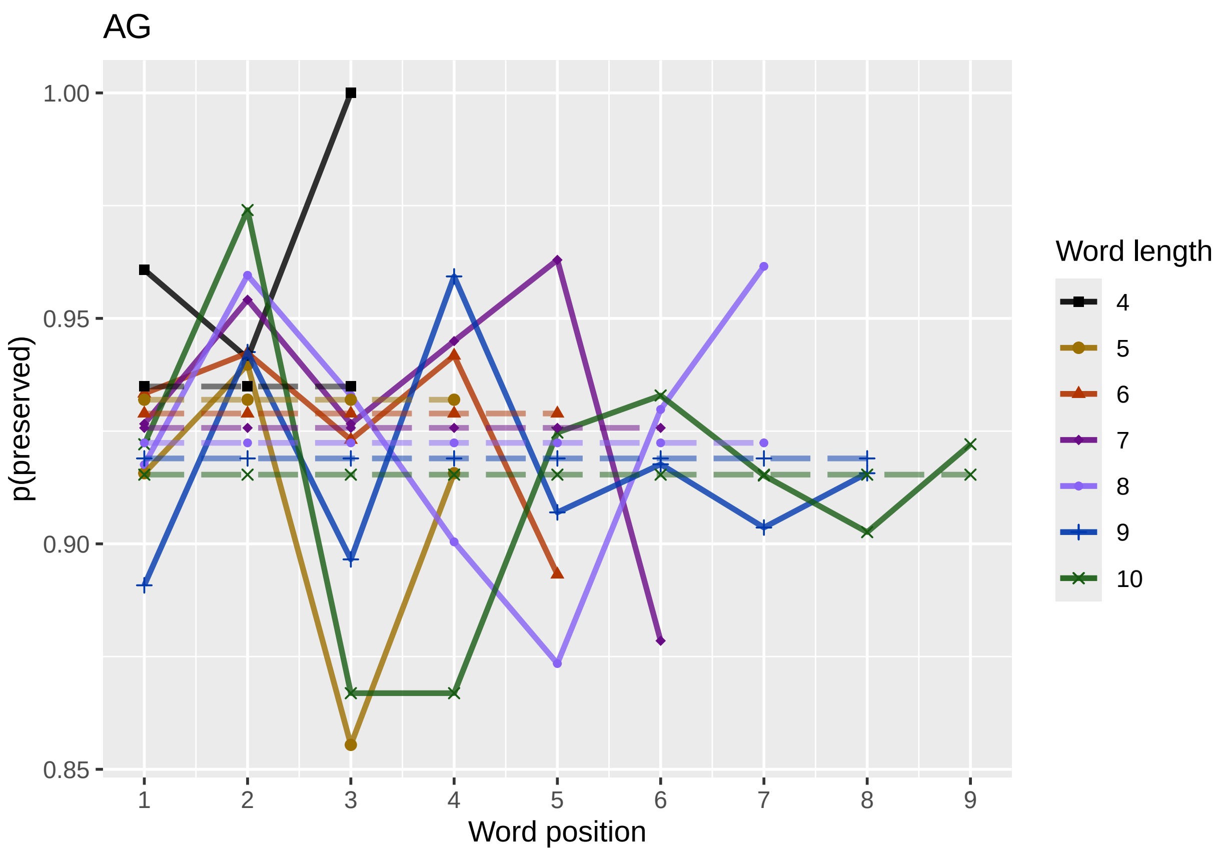
```
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.935 0.935 0.935 NA     NA     NA     NA     NA     NA
## 2       5 0.932 0.932 0.932  0.932 NA     NA     NA     NA     NA
## 3       6 0.929 0.929 0.929  0.929  0.929 NA     NA     NA     NA
## 4       7 0.926 0.926 0.926  0.926  0.926  0.926 NA     NA     NA
## 5       8 0.922 0.922 0.922  0.922  0.922  0.922  0.922 NA     NA
## 6       9 0.919 0.919 0.919  0.919  0.919  0.919  0.919  0.919 NA
## 7      10 0.915 0.915 0.915  0.915  0.915  0.915  0.915  0.915  0.915
```

```
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: 0.84 - 1.01"
```

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

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

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

# in case min is close to the end or we are not using a min (for non-U-shaped)
# functions, we need to get differences _first_ and then average those (e.g.
```

```r
# 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: "
```

```r
print(OA_mean_len_diff)
```

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

```r
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): "
```

```r
print(OA_mean_pos_diff)
```

```
## [1] -0.004931826
```

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

```r
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"
```

```r
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:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.63149     -0.04805      0.13262
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2221  AIC: 2371
## log likelihood:  -1110.627
## Nagelkerke R2:  0.01424861
## % pres/err predicted correctly:  -605.0022
## % of predictable range [ (model-null)/(1-null) ]:  0.007124849
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.44897           0.02461           0.36520          -0.05557          -0.02887
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4070 Residual
## Null Deviance:       2246
## Residual Deviance: 2219  AIC: 2372
## log likelihood:  -1109.255
## Nagelkerke R2:  0.01582738
## % pres/err predicted correctly:  -604.5794
## % of predictable range [ (model-null)/(1-null) ]:  0.007817436
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.44883           0.01980           0.36915          -0.04603          -0.03774
##     log_freq:pos
##          0.01603
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4069 Residual
## Null Deviance:       2246
## Residual Deviance: 2217  AIC: 2373
## log likelihood:  -1108.72
## Nagelkerke R2:  0.0164428
## % pres/err predicted correctly:  -604.3983
## % of predictable range [ (model-null)/(1-null) ]:  0.008114217
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.62171      -0.04493       0.11043       0.00551
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2221  AIC: 2373
## log likelihood:  -1110.548
## Nagelkerke R2:  0.01433919
## % pres/err predicted correctly:  -604.9676
## % of predictable range [ (model-null)/(1-null) ]:  0.007181527
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.45346      0.02697     -0.05557      0.13847
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2221  AIC: 2373
## log likelihood:  -1110.399
## Nagelkerke R2:  0.01451032
## % pres/err predicted correctly:  -604.9309
## % of predictable range [ (model-null)/(1-null) ]:  0.007241541
## ***************************
## 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
##          2.732177           0.006502          -0.105946          -0.015207          -0.008232
##      pos:log_freq
##          0.079446
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4069 Residual
## Null Deviance:       2246
## Residual Deviance: 2219  AIC: 2373
## log likelihood:  -1109.291
## Nagelkerke R2:  0.01578577
## % pres/err predicted correctly:  -604.3991
## % of predictable range [ (model-null)/(1-null) ]:  0.008112955
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         2.595598          0.020083          0.360588          0.007897         -0.124038
## stimlen:log_freq
##        -0.028308
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4069 Residual
## Null Deviance:       2246
## Residual Deviance: 2218  AIC: 2373
## log likelihood:  -1109.038
## Nagelkerke R2:  0.01607688
## % pres/err predicted correctly:  -604.4273
## % of predictable range [ (model-null)/(1-null) ]:  0.008066769
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##         2.454678         -0.004558          0.365006         -0.028858
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2222  AIC: 2373
## log likelihood:  -1111.137
## Nagelkerke R2:  0.01366035
## % pres/err predicted correctly:  -605.2051
## % of predictable range [ (model-null)/(1-null) ]:  0.006792385
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    2.616546     0.021939     0.008809    -0.132006     0.138221
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4070 Residual
## Null Deviance:       2246
## Residual Deviance: 2220  AIC: 2374
## log likelihood:  -1110.128
## Nagelkerke R2:  0.01482258
## % pres/err predicted correctly:  -604.7542
## % of predictable range [ (model-null)/(1-null) ]:  0.007531153
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          2.578691           0.015877           0.238261           0.006037          -0.101313
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.034098          -0.006837           0.076724
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4067 Residual
## Null Deviance:       2246
## Residual Deviance: 2216  AIC: 2374
## log likelihood:  -1107.856
## Nagelkerke R2:  0.01743618
## % pres/err predicted correctly:  -603.9807
## % of predictable range [ (model-null)/(1-null) ]:  0.008798385
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##    2.459099    -0.002199     0.138325
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2225  AIC: 2374
## log likelihood:  -1112.282
## Nagelkerke R2:  0.01234196
## % pres/err predicted correctly:  -605.5784
## % of predictable range [ (model-null)/(1-null) ]:  0.006180696
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     2.455156      0.025482     -0.052585      0.119966      0.004514
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4070 Residual
## Null Deviance:       2246
## Residual Deviance: 2221  AIC: 2375
## log likelihood:  -1110.348
## Nagelkerke R2:  0.01456994
## % pres/err predicted correctly:  -604.9097
## % of predictable range [ (model-null)/(1-null) ]:  0.007276383
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          2.575282           0.021353           0.005306          -0.101888          -0.008216
## I(pos^2):log_freq       pos:log_freq
##         -0.008341           0.079400
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4068 Residual
## Null Deviance:       2246
## Residual Deviance: 2218  AIC: 2375
## log likelihood:  -1109.155
## Nagelkerke R2:  0.01594212
## % pres/err predicted correctly:  -604.3726
## % of predictable range [ (model-null)/(1-null) ]:  0.008156244
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.65773     -0.06169
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2391
## log likelihood:  -1120.213
## Nagelkerke R2:  0.003185316
## % pres/err predicted correctly:  -608.4538
## % of predictable range [ (model-null)/(1-null) ]:  0.001469611
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.775973     0.007979    -0.132372
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2392
## log likelihood:  -1119.982
## Nagelkerke R2:  0.003453151
## % pres/err predicted correctly:  -608.3341
## % of predictable range [ (model-null)/(1-null) ]:  0.001665846
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.79181     -0.02058     -0.05526
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2392
## log likelihood:  -1120.067
## Nagelkerke R2:  0.003353801
## % pres/err predicted correctly:  -608.3648
## % of predictable range [ (model-null)/(1-null) ]:  0.001615491
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.37196     -0.09078     -0.23464      0.02105
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2239  AIC: 2393
## log likelihood:  -1119.417
## Nagelkerke R2:  0.004106173
## % pres/err predicted correctly:  -608.1477
## % of predictable range [ (model-null)/(1-null) ]:  0.001971227
## ***************************
## model index:  20
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##    2.964870    -0.025849     0.009353    -0.136508
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2393
## log likelihood:  -1119.758
## Nagelkerke R2:  0.003711289
## % pres/err predicted correctly:  -608.2016
## % of predictable range [ (model-null)/(1-null) ]:  0.001882812
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.79711     -0.04954
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2244  AIC: 2394
## log likelihood:  -1121.94
## Nagelkerke R2:  0.00118733
## % pres/err predicted correctly:  -608.906
## % of predictable range [ (model-null)/(1-null) ]:  0.0007288599
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.413
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4074 Residual
## Null Deviance:       2246
## Residual Deviance: 2246  AIC: 2395
## log likelihood:  -1122.965
## Nagelkerke R2:  2.620228e-16
## % pres/err predicted correctly:  -609.3508
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         2.919008         -0.034529         -0.039542          0.069241          0.004649
##      stimlen:pos
##        -0.015637
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4069 Residual
## Null Deviance:       2246
## Residual Deviance: 2239  AIC: 2396
## log likelihood:  -1119.267
## Nagelkerke R2:  0.004279799
## % pres/err predicted correctly:  -608.0814
## % of predictable range [ (model-null)/(1-null) ]:  0.002079854
## ***************************
```

```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]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	stimlen:I(pos^2)
preserved ~ pos + log_freq	2371.0640	0.000000	1.0000000	[illegible]	[illegible]	[illegible]	NA	[illegible]	[illegible]	-0.0480460	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	2372.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	2372.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.0160286	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	2372.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	NA	-0.0449341	0.0055096	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	2373.[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0555698	NA	NA	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	[illegible]	I(pos^2)	I(pos^2):log_freq	[illegible]	stimlen:pos	stimlen:I(pos^2)
preserved ~ (I(pos^2) + pos) * log_freq	2373[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0152073	NA	-0.1059460	0.0794401	NA	0.0065018	-0.0082318	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	2373[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0283077	[illegible]	NA	NA	0.0078968	NA	NA	NA	NA
preserved ~ stimlen * log_freq	2373[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0045578	0.3650059	-0.0288577	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	2373[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.1320062	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	2374[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	-0.0068366	NA	NA
preserved ~ stimlen + log_freq	2374[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0021992	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	2374[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0525854	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	2375[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0082158	NA	-0.1018880	[illegible]	NA	0.0053058	-0.0083410	NA	NA	NA
preserved ~ pos	2391[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.0616854	NA	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	2392[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.1323722	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen + pos	2392[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0205806	NA	NA	-0.0552584	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	2392[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0907800	NA	NA	-0.2346360	NA	NA	NA	NA	NA	0.0210517	NA
preserved ~ stimlen + I(pos^2) + pos	2393[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0258491	NA	NA	-0.1365082	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen	2393[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0495376	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	2394[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICexpAICwAICNagR2(Intercept)stimlenlog_freqstimlenpos_pos:log_freqlog_freqI(pos^2)pos^2)log_freqstimlenI(pos^2)stimlen:I(pos^2)									
preserved ~ stimlen * (I(pos^2) + pos)	2396.322553890033000742791900 8	-0.0345294	NA	NA	0.0692NA13	NA	-0.0395425	NA	NA	-0.0156366	0.0046485

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.63149     -0.04805      0.13262
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2221  AIC: 2371
```

```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.
```

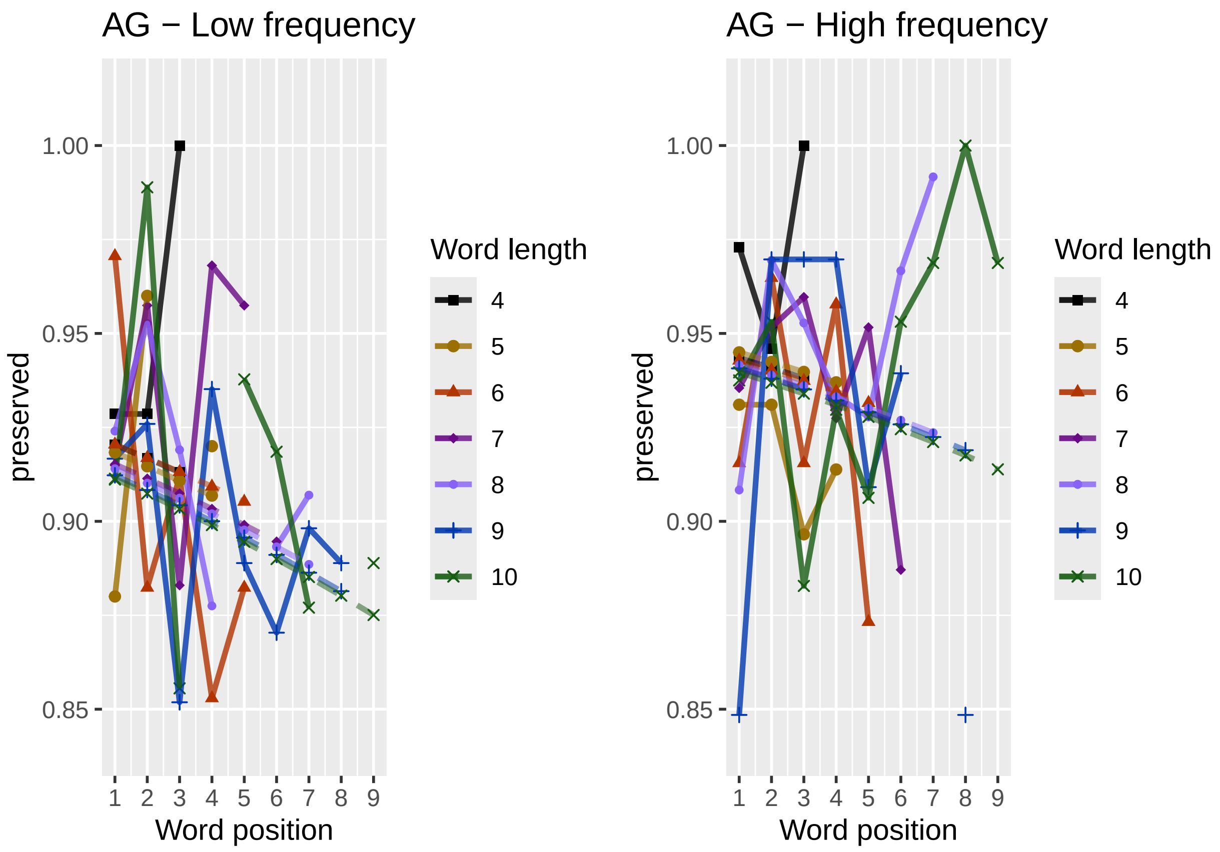
```r
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preser
```

```
## `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 5 rows 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()`).

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

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



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

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

## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.7389      -0.8641
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2221
## log likelihood:  -1042.047
## Nagelkerke R2:  0.0918929
## % pres/err predicted correctly:  -570.316
## % of predictable range [ (model-null)/(1-null) ]:  0.06395465
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.25232      0.06381
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2241  AIC: 2391
## log likelihood:  -1120.464
## Nagelkerke R2:  0.002895157
## % pres/err predicted correctly:  -608.4979
## % of predictable range [ (model-null)/(1-null) ]:  0.00139748
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.65773     -0.06169
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2391
## log likelihood:  -1120.213
## Nagelkerke R2:  0.003185316
## % pres/err predicted correctly:  -608.4538
## % of predictable range [ (model-null)/(1-null) ]:  0.001469611
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.775973     0.007979    -0.132372
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2392
## log likelihood:  -1119.982
## Nagelkerke R2:  0.003453151
## % pres/err predicted correctly:  -608.3341
## % of predictable range [ (model-null)/(1-null) ]:  0.001665846
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.79711     -0.04954
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2244  AIC: 2394
## log likelihood:  -1121.94
## Nagelkerke R2:  0.00118733
## % pres/err predicted correctly:  -608.906
## % of predictable range [ (model-null)/(1-null) ]:  0.0007288599
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.413
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4074 Residual
## Null Deviance:       2246
## Residual Deviance: 2246  AIC: 2395
## log likelihood:  -1122.965
## Nagelkerke R2:  2.620228e-16
## % pres/err predicted correctly:  -609.3508
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

MEAICSummary<-data.frame(Model=MERes$Model,
                       AIC=MERes$AIC,row.names=MERes$Model)
```

```
MEAICSummary$DeltaAIC<-MEAICSummary$AIC-MEAICSummary$AIC[1]
MEAICSummary$AICexp<-exp(-0.5*MEAICSummary$DeltaAIC)
MEAICSummary$AICwt<-MEAICSummary$AICexp/sum(MEAICSummary$AICexp)
MEAICSummary$NagR2<-MERes$NagR2

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	2220.789	0.0000	1	1	0.0918929	2.738863	NA	-0.8641068	NA	NA	NA
preserved ~ CumPres	2391.156	170.3669	0	0	0.0028952	2.252316	0.0638104	NA	NA	NA	NA
preserved ~ pos	2391.202	170.4133	0	0	0.0031853	3.657733	NA	NA	NA	-0.0616854	NA
preserved ~ (I(pos^2) + pos)	2392.213	171.4246	0	0	0.0034532	2.775973	NA	NA	0.0079788	-0.1323722	NA
preserved ~ stimlen	2393.695	172.9066	0	0	0.0011873	3.797110	NA	NA	NA	NA	-0.0495376
preserved ~ 1	2394.879	174.0903	0	0	0.0000000	2.412943	NA	NA	NA	NA	NA

```
if(DoSimulations){
      BestMEModelFormulaRnd <- BestMEModelFormula
    if(grepl("CumPres",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumPres","RndCumPres",BestMEModelFormulaRnd)
    }else if(grepl("CumErr",BestMEModelFormulaRnd)){
      BestMEModelFormulaRnd <- gsub("CumErr","RndCumErr",BestMEModelFormulaRnd)
    }

    RndModelAIC<-numeric(length=RandomSamples)
    for(rindex in seq(1,RandomSamples)){
    # Shuffle cumulative values
      PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
      PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
      BestModelRnd <- glm(as.formula(BestMEModelFormulaRnd),
                          family="binomial",data=PosDat)
      RndModelAIC[rindex] <- BestModelRnd$aic
    }
    ModelNames<-c(paste0("***",BestMEModelFormula),
                  rep(BestMEModelFormulaRnd,RandomSamples))
    AICValues <- c(BestMEModel$aic,RndModelAIC)
    BestMEModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
    BestMEModelRndDF <- BestMEModelRndDF %>% arrange(AIC)
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random average"),
                                         AIC=c(mean(RndModelAIC))))
    BestMEModelRndDF <- rbind(BestMEModelRndDF,
                              data.frame(Name=c("Random SD"),
                                        AIC=c(sd(RndModelAIC))))

    write.csv(BestMEModelRndDF,
              paste0(TablesDir,CurPat,"_",CurTask,
                     "_best_main_effects_model_with_random_cum_term.csv"),
              row.names = FALSE)
}
```

```
syll_component_summary <- PosDat %>%
  group_by(syll_component) %>% summarise(MeanPres = mean(preserved),
                                        N = n())
write.csv(syll_component_summary,paste0(TablesDir,CurPat,"_",CurTask,"_syllable_component_summary.csv")
kable(syll_component_summary)
```

syll_component	MeanPres	N
1	0.9083802	533
O	0.9067503	1880
P	0.7333333	30
S	0.9458333	240
V	0.9354988	1392

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.7331      -0.8941
##
## Degrees of Freedom: 3804 Total (i.e. Null);  3803 Residual
## Null Deviance:       2098
## Residual Deviance: 1940  AIC: 2075
## log likelihood:  -969.8034
## Nagelkerke R2:  0.09634803
## % pres/err predicted correctly:  -531.576
## % of predictable range [ (model-null)/(1-null) ]:  0.06825395
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.90015      0.01309     -0.18982
##
## Degrees of Freedom: 3804 Total (i.e. Null);  3802 Residual
## Null Deviance:       2098
## Residual Deviance: 2090  AIC: 2242
## log likelihood:  -1044.888
## Nagelkerke R2:  0.005259211
## % pres/err predicted correctly:  -569.1694
## % of predictable range [ (model-null)/(1-null) ]:  0.002484024
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.70284     -0.07325
##
## Degrees of Freedom: 3804 Total (i.e. Null);  3803 Residual
## Null Deviance:       2098
## Residual Deviance: 2091  AIC: 2242
## log likelihood:  -1045.473
## Nagelkerke R2:  0.004535364
## % pres/err predicted correctly:  -569.4193
## % of predictable range [ (model-null)/(1-null) ]:  0.002046934
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.94847     -0.06936
##
## Degrees of Freedom: 3804 Total (i.e. Null);  3803 Residual
## Null Deviance:       2098
## Residual Deviance: 2095  AIC: 2244
## log likelihood:  -1047.261
## Nagelkerke R2:  0.002321062
## % pres/err predicted correctly:  -569.8413
## % of predictable range [ (model-null)/(1-null) ]:  0.001308547
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.26182      0.06245
##
## Degrees of Freedom: 3804 Total (i.e. Null);  3803 Residual
## Null Deviance:       2098
## Residual Deviance: 2094  AIC: 2245
## log likelihood:  -1047.101
## Nagelkerke R2:  0.002519934
## % pres/err predicted correctly:  -569.8912
## % of predictable range [ (model-null)/(1-null) ]:  0.001221295
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.409
##
## Degrees of Freedom: 3804 Total (i.e. Null);  3804 Residual
## Null Deviance:       2098
## Residual Deviance: 2098  AIC: 2247
## log likelihood:  -1049.134
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -570.5893
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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	2075.257	0.0000	1	1	0.0963480	2.733072	NA	-0.8941137	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2241.658	166.4007	0	0	0.0052592	2.900148	NA	NA	0.0130862	-0.1898175	NA
preserved ~ pos	2241.664	166.4067	0	0	0.0045354	2.702837	NA	NA	NA	-0.0732469	NA
preserved ~ stimlen	2243.808	168.5513	0	0	0.0023211	2.948471	NA	NA	NA	NA	-0.069362
preserved ~ CumPres	2244.514	169.2573	0	0	0.0025199	2.261815	0.0624478	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ 1	2247.084	171.8265	0	0	0.0000000	2.408799	NA	NA	NA	NA	NA

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.739       -1.031
##
## Degrees of Freedom: 3271 Total (i.e. Null);  3270 Residual
## Null Deviance:       1783
## Residual Deviance: 1651  AIC: 1764
## log likelihood:  -825.7224
## Nagelkerke R2:  0.09377975
## % pres/err predicted correctly:  -449.2824
## % of predictable range [ (model-null)/(1-null) ]:  0.06874729
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.75011     -0.08002
##
## Degrees of Freedom: 3271 Total (i.e. Null);  3270 Residual
## Null Deviance:       1783
## Residual Deviance: 1775  AIC: 1899
## log likelihood:  -887.557
## Nagelkerke R2:  0.005702584
## % pres/err predicted correctly:  -481.2785
## % of predictable range [ (model-null)/(1-null) ]:  0.002574556
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.876647     0.008619    -0.156636
##
## Degrees of Freedom: 3271 Total (i.e. Null);  3269 Residual
## Null Deviance:       1783
## Residual Deviance: 1775  AIC: 1900
## log likelihood:  -887.3217
## Nagelkerke R2:  0.006044097
## % pres/err predicted correctly:  -481.1627
## % of predictable range [ (model-null)/(1-null) ]:  0.00281393
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.84477     -0.05372
##
## Degrees of Freedom: 3271 Total (i.e. Null);  3270 Residual
## Null Deviance:       1783
## Residual Deviance: 1781  AIC: 1904
## log likelihood:  -890.5163
## Nagelkerke R2:  0.001403369
## % pres/err predicted correctly:  -482.1201
## % of predictable range [ (model-null)/(1-null) ]:  0.0008339061
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.429
##
## Degrees of Freedom: 3271 Total (i.e. Null);  3271 Residual
## Null Deviance:       1783
## Residual Deviance: 1783  AIC: 1905
## log likelihood:  -891.4811
## Nagelkerke R2:  5.285405e-16
## % pres/err predicted correctly:  -482.5233
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.36054      0.03361
##
## Degrees of Freedom: 3271 Total (i.e. Null);  3270 Residual
## Null Deviance:       1783
## Residual Deviance: 1782  AIC: 1906
## log likelihood:  -891.0903
## Nagelkerke R2:  0.0005685129
## % pres/err predicted correctly:  -482.3696
## % of predictable range [ (model-null)/(1-null) ]:  0.000318006
## ***************************
```

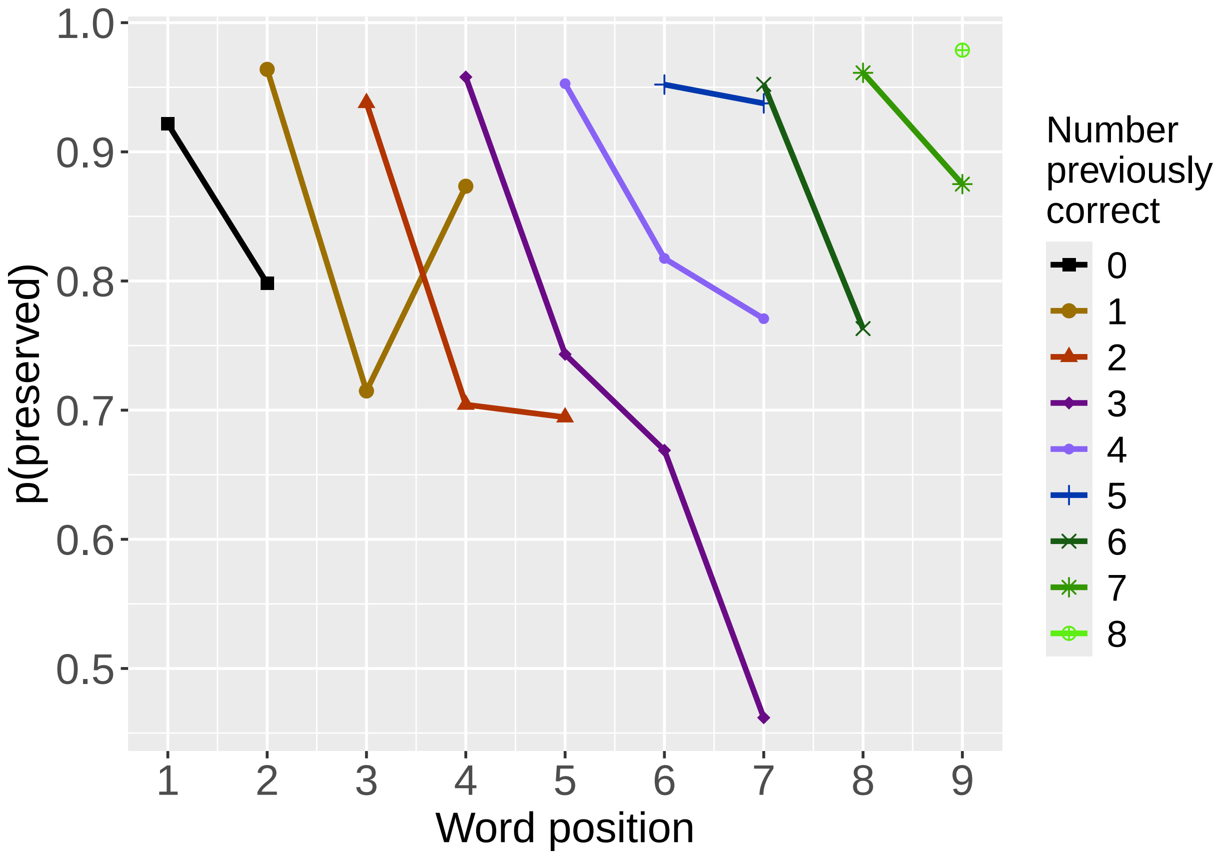
```r
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1763.583	0.0000	1	1	0.0937798	2.738523	NA	-1.031378	NA	NA	NA
preserved ~ pos	1899.105	135.5214	0	0	0.0057026	2.750113	NA	NA	NA	-0.0800229	NA
preserved ~ (I(pos^2) + pos)	1900.142	136.5584	0	0	0.0060441	2.876647	NA	NA	0.0086188	-0.1566363	NA
preserved ~ stimlen	1904.322	140.7381	0	0	0.0014031	2.844775	NA	NA	NA	NA	-0.0537226
preserved ~ 1	1905.193	141.6093	0	0	0.0000000	2.428577	NA	NA	NA	NA	NA
preserved ~ CumPres	1906.131	142.5474	0	0	0.0005685	2.360540	0.033606	NA	NA	NA	NA

```r
# 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.
```

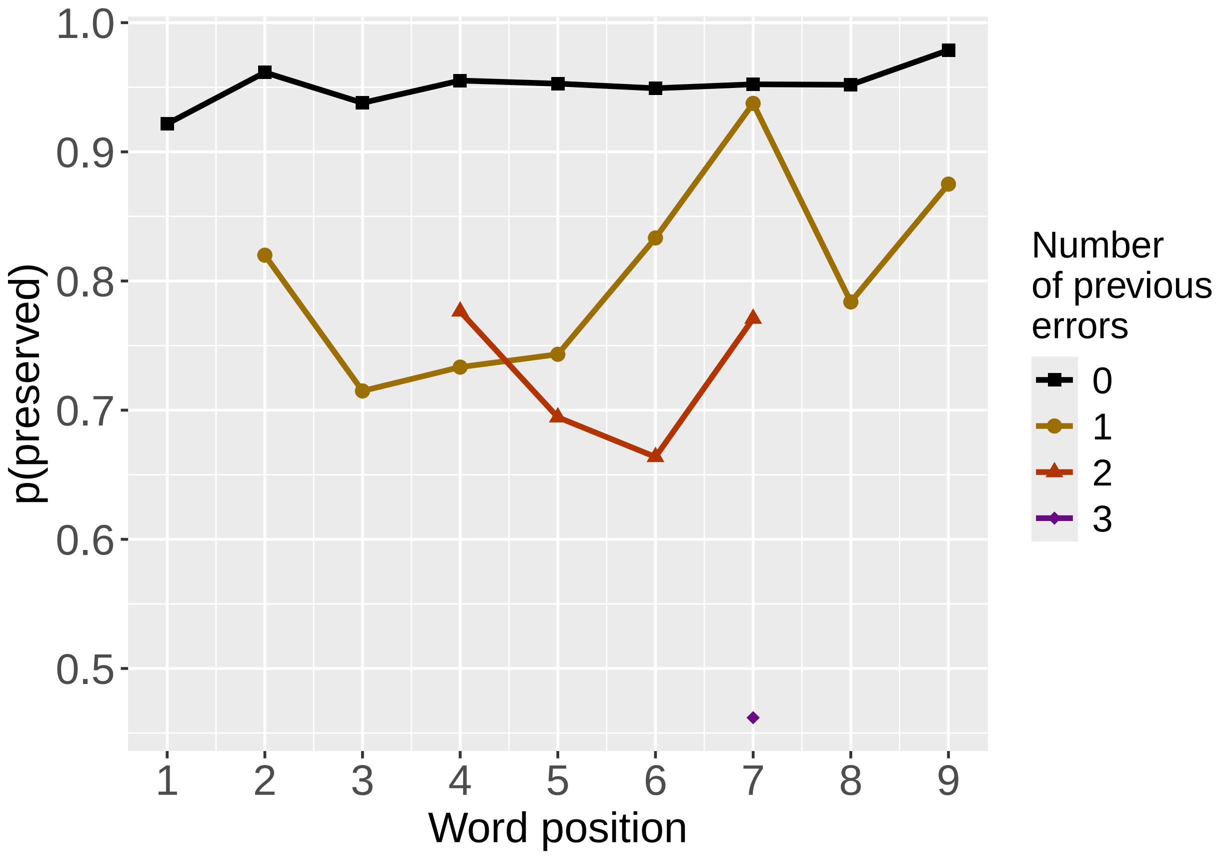
```r
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



```r
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
```

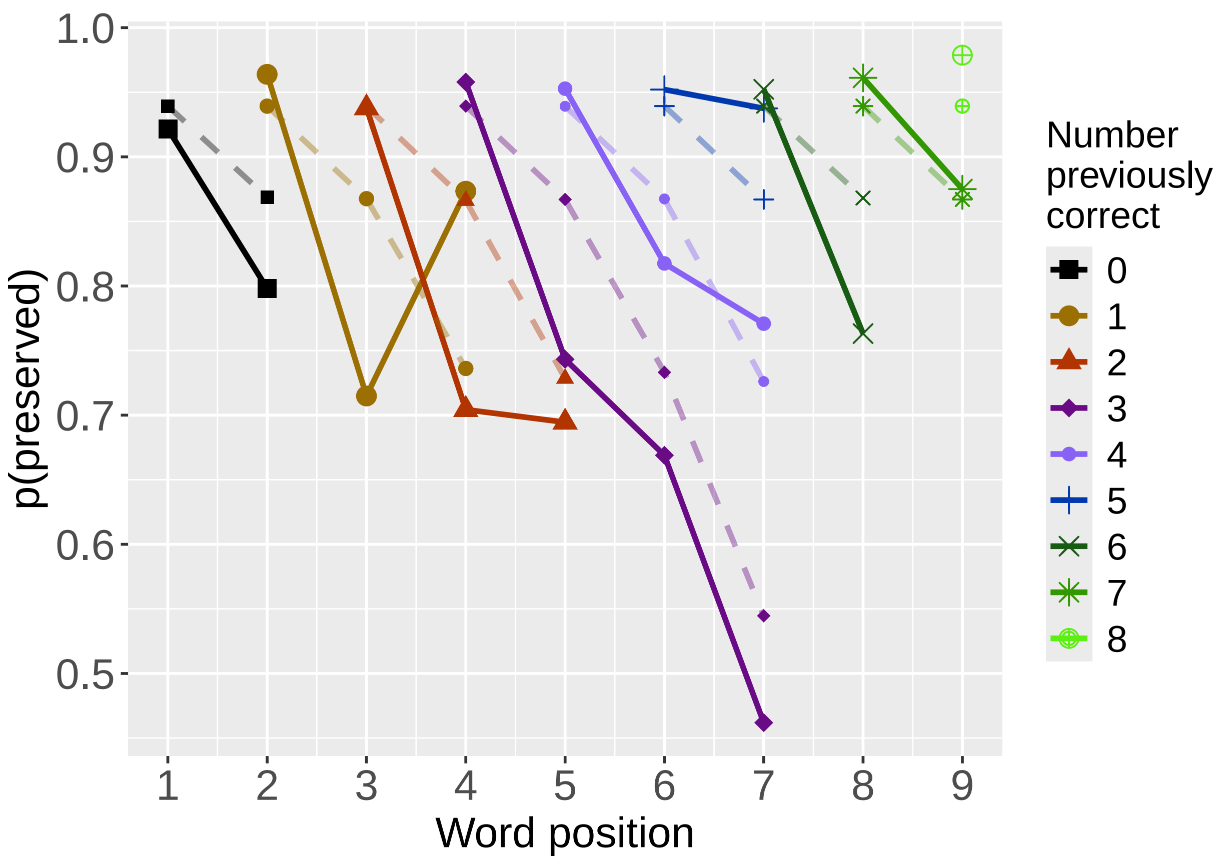
```r
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

```r
# 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.
```

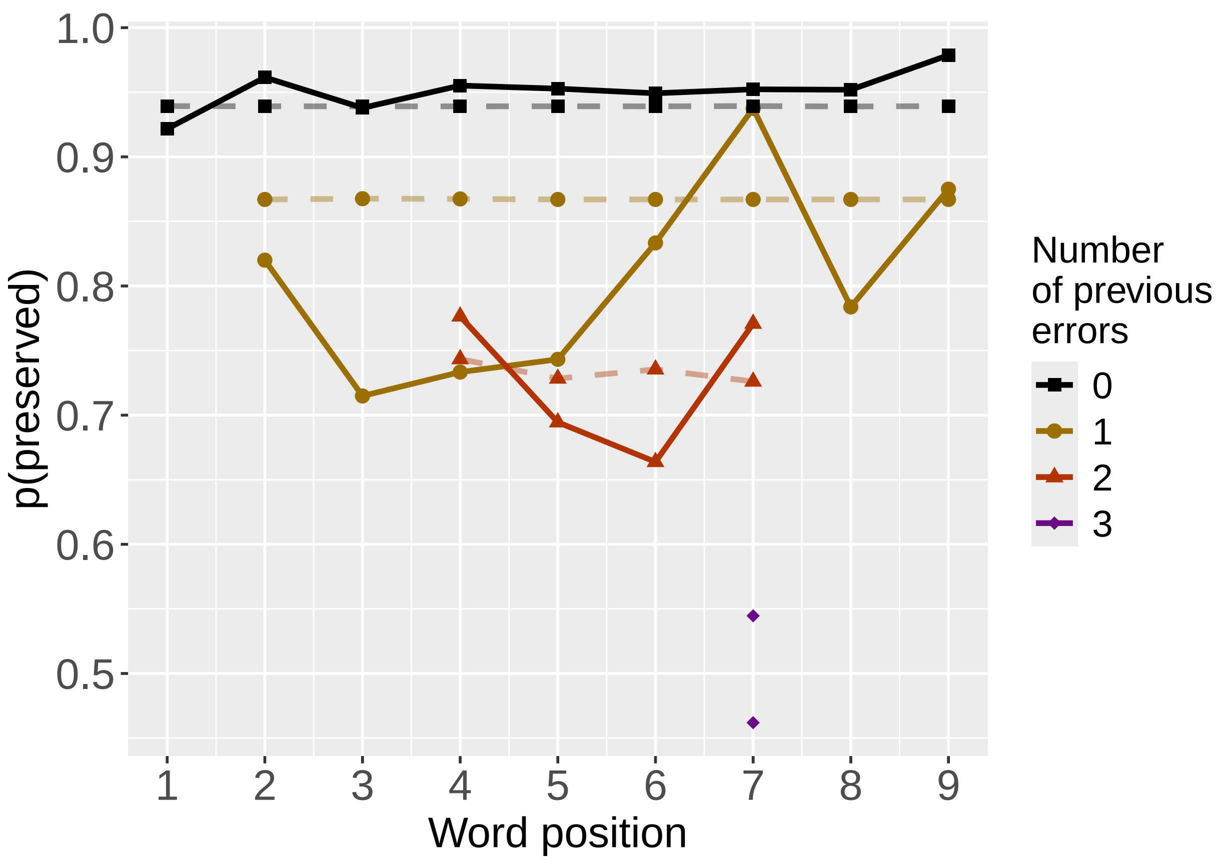
```r
print(PrevCorPlot)
```



```
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
##     2.70126     -0.95305      0.01650     -0.06511
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2076  AIC: 2214
## log likelihood:  -1037.964
## Nagelkerke R2:  0.0964336
## % pres/err predicted correctly:  -568.1572
## % of predictable range [ (model-null)/(1-null) ]:  0.06749171
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.7389      -0.8641
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2221
## log likelihood:  -1042.047
## Nagelkerke R2:  0.0918929
## % pres/err predicted correctly:  -570.316
## % of predictable range [ (model-null)/(1-null) ]:  0.06395465
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##    2.775973     0.007979    -0.132372
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2392
## log likelihood:  -1119.982
## Nagelkerke R2:  0.003453151
## % pres/err predicted correctly:  -608.3341
## % of predictable range [ (model-null)/(1-null) ]:  0.001665846
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	2214.492	0.000000	1.0000000	0.958838	0.0964336	2.701256	-0.9530498	0.0165016	-0.0651065

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	2220.789	6.296411	0.0429291	0.041162	0.0918929	2.738863	-0.8641068	NA	NA
preserved ~ I(pos^2) + pos	2392.213	177.721007	0.0000000	0.000000	0.0034532	2.775973	NA	0.0079788	-0.1323722

```r
########
# 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
##      2.7389      -0.8641
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2221
## log likelihood:  -1042.047
## Nagelkerke R2:  0.0918929
## % pres/err predicted correctly:  -570.316
## % of predictable range [ (model-null)/(1-null) ]:  0.06395465
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.53347     -0.87449      0.02708
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2223
## log likelihood:  -1041.771
## Nagelkerke R2:  0.09219977
## % pres/err predicted correctly:  -570.1369
## % of predictable range [ (model-null)/(1-null) ]:  0.06424819
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.79711     -0.04954
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2244  AIC: 2394
## log likelihood:  -1121.94
## Nagelkerke R2:  0.00118733
## % pres/err predicted correctly:  -608.906
## % of predictable range [ (model-null)/(1-null) ]:  0.0007288599
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	2220.789	0.000000	1.0000000	0.723114	0.0918929	2.738863	-0.8641068	NA
preserved ~ CumErr + stimlen	2222.709	1.919922	0.3829078	0.276886	0.0921998	2.533469	-0.8744902	0.0270827
preserved ~ stimlen	2393.695	172.906624	0.0000000	0.000000	0.0011873	2.797110	NA	-0.0495376

```r
########
# 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
##     2.54644     -0.87063      0.07717
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2078  AIC: 2215
## log likelihood:  -1038.841
## Nagelkerke R2:  0.09545897
## % pres/err predicted correctly:  -568.8893
## % of predictable range [ (model-null)/(1-null) ]:  0.06629226
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.7389      -0.8641
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2221
## log likelihood:  -1042.047
## Nagelkerke R2:  0.0918929
## % pres/err predicted correctly:  -570.316
## % of predictable range [ (model-null)/(1-null) ]:  0.06395465
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.25232      0.06381
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2241  AIC: 2391
## log likelihood:  -1120.464
## Nagelkerke R2:  0.002895157
## % pres/err predicted correctly:  -608.4979
## % of predictable range [ (model-null)/(1-null) ]:  0.00139748
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	2215.377	0.000000	1.0000000	0.937381	0.0954590	2.546445	-0.8706311	0.0771687
preserved ~ CumErr	2220.789	5.412042	0.0668021	0.062619	0.0918929	2.738863	-0.8641068	NA
preserved ~ CumPres	2391.156	175.778894	0.0000000	0.000000	0.0028952	2.252316	NA	0.0638104

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     2.46928     -0.94780      0.07717
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2078  AIC: 2215
## log likelihood:  -1038.841
## Nagelkerke R2:  0.09545897
## % pres/err predicted correctly:  -568.8893
## % of predictable range [ (model-null)/(1-null) ]:  0.06629226
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.7389      -0.8641
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2221
## log likelihood:  -1042.047
## Nagelkerke R2:  0.0918929
## % pres/err predicted correctly:  -570.316
## % of predictable range [ (model-null)/(1-null) ]:  0.06395465
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     2.65773     -0.06169
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2240  AIC: 2391
## log likelihood:  -1120.213
## Nagelkerke R2:  0.003185316
## % pres/err predicted correctly:  -608.4538
## % of predictable range [ (model-null)/(1-null) ]:  0.001469611
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	2215.377	0.000000	1.0000000	0.937381	0.0954590	2.469276	-0.9477998	0.0771687
preserved ~ CumErr	2220.789	5.412042	0.0668021	0.062619	0.0918929	2.738863	-0.8641068	NA
preserved ~ pos	2391.202	175.825299	0.0000000	0.000000	0.0031853	2.657733	NA	-0.0616854

```r
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	2214.492	0.000000	1.0000000	0.958838	0.0964332	2.701256	-0.9530498	0.0165016	-0.0651065	NA	NA
preserved ~ CumErr + CumPres	2215.377	0.000000	1.0000000	0.937381	0.0954590	2.546445	-0.8706311	NA	NA	NA	0.0771687
preserved ~ CumErr + pos	2215.377	0.000000	1.0000000	0.937381	0.0954590	2.469276	-0.9477998	NA	0.0771687	NA	NA
preserved ~ CumErr	2220.789	6.296411	0.0429291	0.041162	0.0918929	2.738863	-0.8641068	NA	NA	NA	NA
preserved ~ CumErr	2220.789	0.000000	1.0000000	0.723114	0.0918929	2.738863	-0.8641068	NA	NA	NA	NA
preserved ~ CumErr	2220.789	5.412042	0.0668021	0.062619	0.0918929	2.738863	-0.8641068	NA	NA	NA	NA
preserved ~ CumErr	2220.789	5.412042	0.0668021	0.062619	0.0918929	2.738863	-0.8641068	NA	NA	NA	NA
preserved ~ CumErr + stimlen	2222.709	1.919922	0.3829078	0.276886	0.0921998	2.533469	-0.8744902	NA	NA	0.0270827	NA
preserved ~ CumPres	2391.156	175.778894	0.0000000	0.000000	0.0028952	2.252316	NA	NA	NA	NA	0.0638104
preserved ~ pos	2391.202	175.825299	0.0000000	0.000000	0.0031853	2.657733	NA	NA	-0.0616854	NA	NA
preserved ~ I(pos^2) + pos	2392.213	177.721007	0.0000000	0.000000	0.0034532	2.775973	NA	0.0079788	-0.1323722	NA	NA
preserved ~ stimlen	2393.695	172.906624	0.0000000	0.000000	0.0011872	2.797110	NA	NA	NA	-0.0495376	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
##     2.69644     -0.94343      0.01813     -0.06540      0.12388
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4070 Residual
## Null Deviance:       2246
## Residual Deviance: 2060  AIC: 2198
## log likelihood:  -1030.167
## Nagelkerke R2:  0.1050791
## % pres/err predicted correctly:  -566.4596
## % of predictable range [ (model-null)/(1-null) ]:  0.07027309
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     2.51530     -0.94408      0.01693     -0.06147      0.02470      0.12897
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4069 Residual
## Null Deviance:       2246
## Residual Deviance: 2060  AIC: 2200
## log likelihood:  -1029.986
## Nagelkerke R2:  0.1052796
## % pres/err predicted correctly:  -566.4109
## % of predictable range [ (model-null)/(1-null) ]:  0.07035283
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.70126     -0.95305      0.01650     -0.06511
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2076  AIC: 2214
## log likelihood:  -1037.964
## Nagelkerke R2:  0.0964336
## % pres/err predicted correctly:  -568.1572
## % of predictable range [ (model-null)/(1-null) ]:  0.06749171
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     2.82621     -0.95236      0.01740     -0.06781     -0.01713
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4070 Residual
## Null Deviance:       2246
## Residual Deviance: 2076  AIC: 2216
## log likelihood:  -1037.869
## Nagelkerke R2:  0.09653887
## % pres/err predicted correctly:  -568.1262
## % of predictable range [ (model-null)/(1-null) ]:  0.06754254
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       2.413
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4074 Residual
## Null Deviance:       2246
## Residual Deviance: 2246  AIC: 2395
## log likelihood:  -1122.965
## Nagelkerke R2:  2.620228e-16
## % pres/err predicted correctly:  -609.3508
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

```r
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	2198.340	0.000000	1.0000000	0.7444804	0.1050792	2.1696440	-0.9434251	0.0181343	-0.0653961	0.1238829	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	2200.482	2.141539	0.3427447	0.2551681	0.1052792	2.6515299	-0.9440835	0.0169267	-0.0614674	0.1289702	0.0247015
preserved ~ CumErr + I(pos^2) + pos	2214.492	16.151977	0.0003109	0.0002315	0.0964332	2.6701256	-0.9530498	0.0165016	-0.0651065	NA	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen	2215.876	17.535709	0.0001557	0.0001159	0.0965382	2.3826214	-0.9523645	0.0173994	-0.0678074	NA	-0.0171268
preserved ~ 1	2394.879	196.538668	0.0000000	0.0000000	0.0000000	2.412943	NA	NA	NA	NA	NA

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

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

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

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

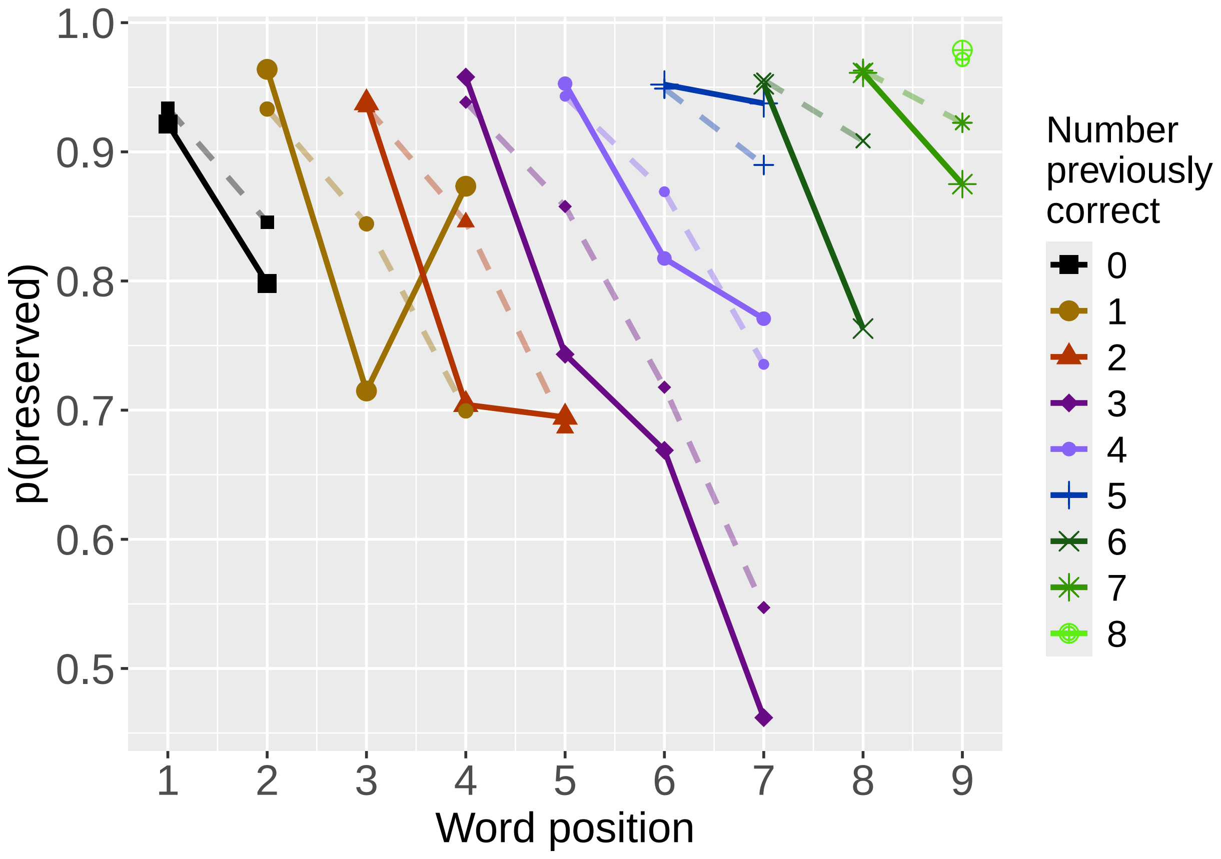
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + log_freq
##          Df Deviance    AIC
## CumErr    1   2220.6 2356.6
## log_freq  1   2075.9 2211.9
## I(pos^2)  1   2062.4 2198.4
## <none>        2060.3 2198.3
## pos       1   2060.7 2196.7
```

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

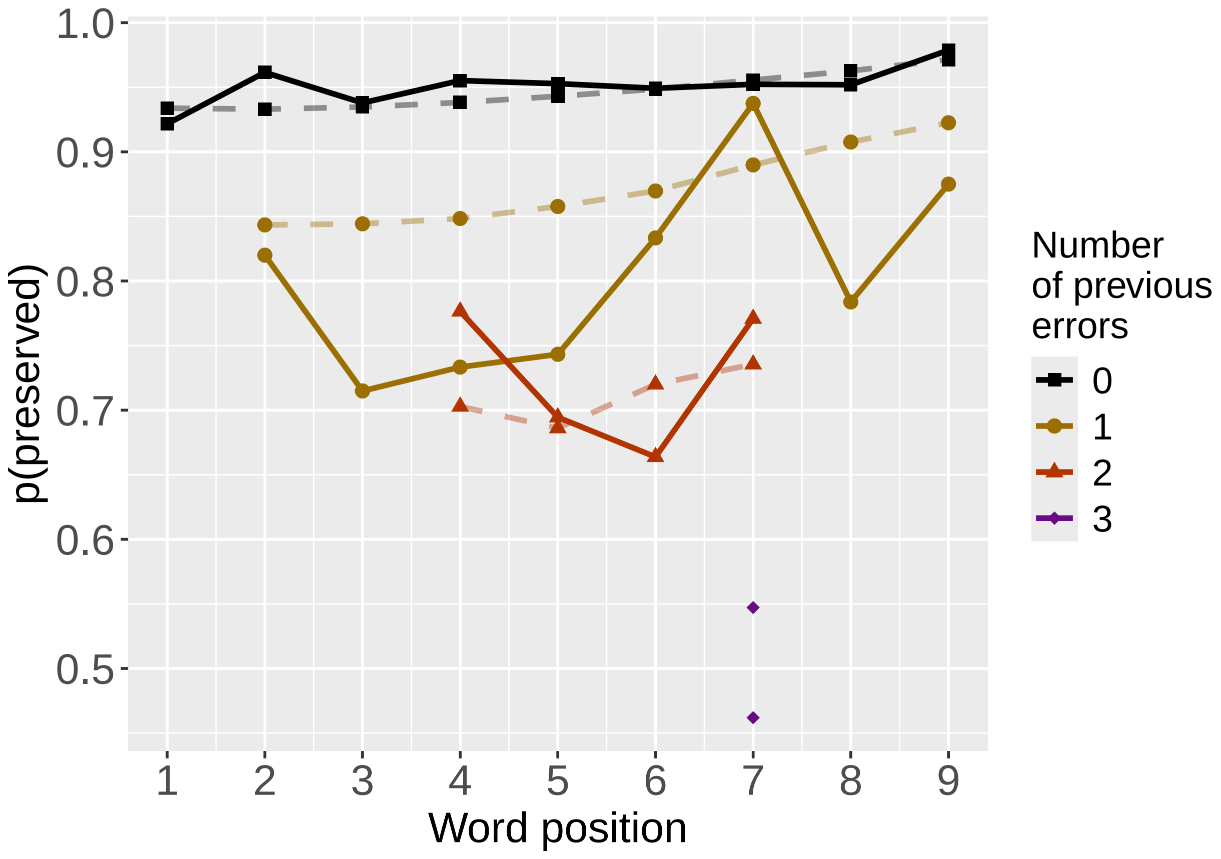
```
print(PrevCorPlot)
```



```r
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.
```

```r
print(PrevErrPlot)
```



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

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

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

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

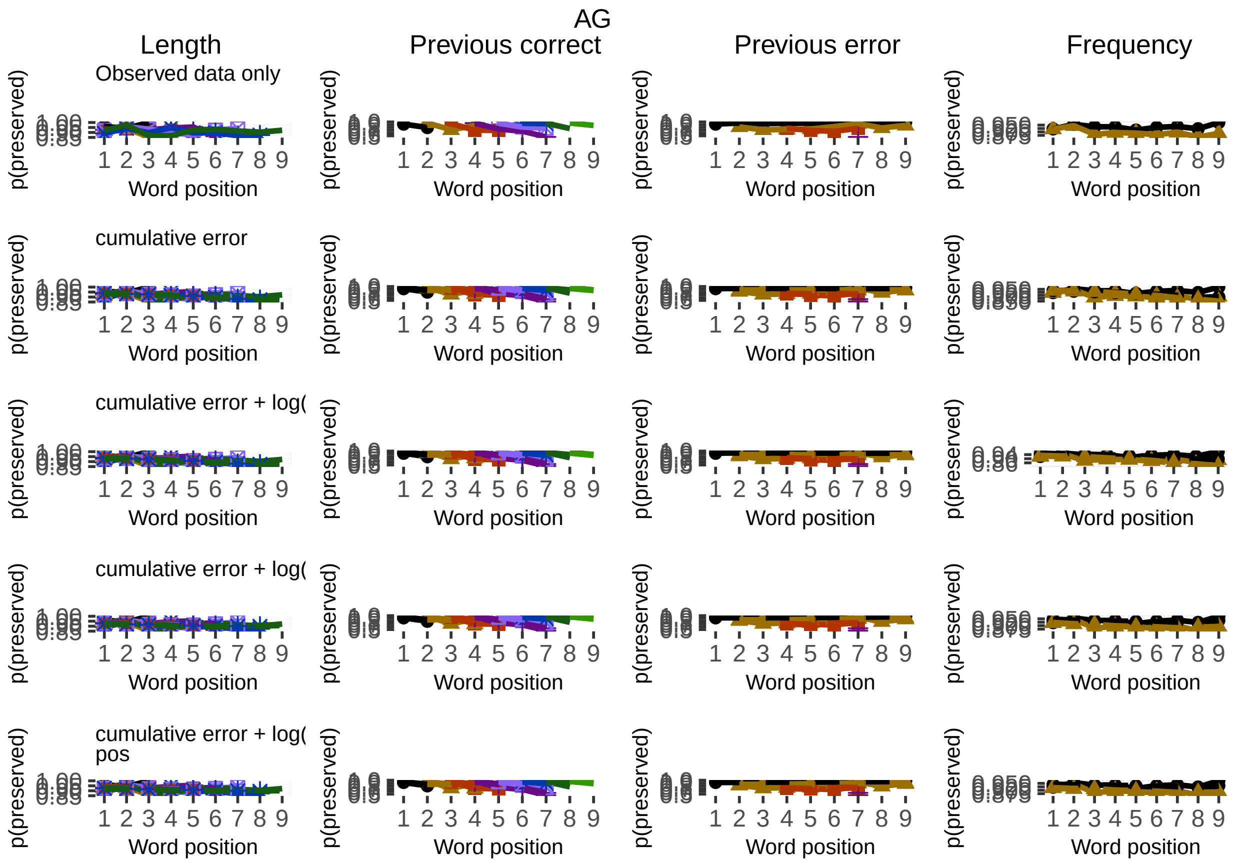
```
if(DoSimulations){
  BestModelFormulaL3Rnd <- BestModelFormulaL3
  if(grepl("CumPres",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumPres","RndCumPres",BestModelFormulaL3Rnd)
  }
  if(grepl("CumErr",BestModelFormulaL3Rnd)){
    BestModelFormulaL3Rnd <- gsub("CumErr","RndCumErr",BestModelFormulaL3Rnd)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

  RndModelAIC<-numeric(length=RandomSamples)
  for(rindex in seq(1,RandomSamples)){
  # Shuffle cumulative values
    PosDat$RndCumPres <- ShuffleWithinWord(PosDat,"CumPres")
    PosDat$RndCumErr <- ShuffleWithinWord(PosDat,"CumErr")
    BestModelRnd <- glm(as.formula(BestModelFormulaL3Rnd),
                        family="binomial",data=PosDat)
    RndModelAIC[rindex] <- BestModelRnd$aic
  }
  ModelNames<-c(paste0("***",BestModelFormulaL3),
                rep(BestModelFormulaL3Rnd,RandomSamples))
  AICValues <- c(BestModelL3$aic,RndModelAIC)
  BestModelRndDF <- data.frame(Name=ModelNames,AIC=AICValues)
  BestModelRndDF <- BestModelRndDF %>% arrange(AIC)
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random average"),
                                     AIC=c(mean(RndModelAIC))))
  BestModelRndDF <- rbind(BestModelRndDF,
                          data.frame(Name=c("Random SD"),
                                     AIC=c(sd(RndModelAIC))))
  write.csv(BestModelRndDF,paste0(TablesDir,CurPat,"_",CurTask,
                                  "_best_model_with_random_cum_term.csv"),
            row.names = FALSE)
}
```

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.7389      -0.8641
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4073 Residual
## Null Deviance:       2246
## Residual Deviance: 2084  AIC: 2221
## log likelihood:  -1042.047
## Nagelkerke R2:  0.0918929
## % pres/err predicted correctly:  -570.316
## % of predictable range [ (model-null)/(1-null) ]:  0.06395465
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq
##      2.7564      -0.8406       0.1118
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4072 Residual
## Null Deviance:       2246
## Residual Deviance: 2071  AIC: 2208
## log likelihood:  -1035.602
## Nagelkerke R2:  0.09905556
## % pres/err predicted correctly:  -569.0605
## % of predictable range [ (model-null)/(1-null) ]:  0.0660117
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq     I(pos^2)          pos
##     2.69644     -0.94343      0.12388      0.01813     -0.06540
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4070 Residual
## Null Deviance:       2246
## Residual Deviance: 2060  AIC: 2198
## log likelihood:  -1030.167
## Nagelkerke R2:  0.1050791
## % pres/err predicted correctly:  -566.4596
## % of predictable range [ (model-null)/(1-null) ]:  0.07027309
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq     I(pos^2)
##     2.58068     -0.94657      0.12387      0.01113
##
## Degrees of Freedom: 4074 Total (i.e. Null);  4071 Residual
## Null Deviance:       2246
## Residual Deviance: 2061  AIC: 2197
## log likelihood:  -1030.336
## Nagelkerke R2:  0.1048919
## % pres/err predicted correctly:  -566.7137
## % of predictable range [ (model-null)/(1-null) ]:  0.06985679
## ***************************
## `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",compress
# use \blandscape and \elandscape to make markdown plots landscape if needed
FactorPlot
```



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

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

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

	CumErr	I(pos^2)	pos	log_freq
McFadden	0.0734937	0.0018238	0.0016005	0.0085601
SquaredCorrelation	0.0420814	0.0010368	0.0009195	0.0049169
Nagelkerke	0.0420814	0.0010368	0.0009195	0.0049169
Estrella	0.0439546	0.0010970	0.0009553	0.0051098

```
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 + log_freq + I(pos^2) + pos CumErr + log_freq + I(pos^2) + pos 2060.334
## CumErr + log_freq + I(pos^2)             CumErr + log_freq + I(pos^2) 2060.673
## CumErr + log_freq                                   CumErr + log_freq 2071.205
## CumErr                                                         CumErr 2084.093
## null                                                             null 2245.930
##                                    deviance_explained percent_explained
## CumErr + log_freq + I(pos^2) + pos           185.5956          8.263640
## CumErr + log_freq + I(pos^2)                 185.2573          8.248580
## CumErr + log_freq                            174.7250          7.779629
## CumErr                                       161.8364          7.205762
## null                                           0.0000          0.000000
##                                    percent_of_explained_deviance increment_in_explained
## CumErr + log_freq + I(pos^2) + pos                     100.00000              0.1822371
## CumErr + log_freq + I(pos^2)                            99.81776              5.6748811
## CumErr + log_freq                                       94.14288              6.9444771
## CumErr                                                  87.19840             87.1984046
## null                                                          NA              0.0000000
```

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

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

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

	deviance	deviance_explained
CumErr + log_freq + I(pos^2) + pos	2060.334	185.5956
CumErr + log_freq + I(pos^2)	2060.673	185.2573
CumErr + log_freq	2071.205	174.7250
CumErr	2084.093	161.8364
null	2245.930	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + log_freq + I(pos^2) + pos	8.263640	100.00000	0.1822371
CumErr + log_freq + I(pos^2)	8.248580	99.81776	5.6748811
CumErr + log_freq	7.779629	94.14288	6.9444771
CumErr	7.205762	87.19840	87.1984046
null	0.000000	NA	0.0000000

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

```
##           Nagelkerke
## CumErr    0.85960052
## I(pos^2) 0.02117920
## pos       0.01878269
## log_freq 0.10043759
```

```
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+log_freq+I(pos^2)+pos	0.8410670	2060.334
preserved ~ CumErr+log_freq+I(pos^2)	0.8432606	2060.673
preserved ~ CumErr	0.8524505	2084.093
preserved ~ CumErr+log_freq	0.8596801	2071.205

```
## 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
## 1 preserved ~ CumErr+log_freq+I(pos^2)+pos       0.8410670       2060.334
## 2     preserved ~ CumErr+log_freq+I(pos^2)       0.8432606       2060.673
## 3                       preserved ~ CumErr       0.8524505       2084.093
## 4              preserved ~ CumErr+log_freq       0.8596801       2071.205
##   diff_CumErr+log_freq+I(pos^2)+pos diff_CumErr+log_freq+I(pos^2)  diff_CumErr
## 1                       0.000000000                  -0.002193538 -0.011383454
## 2                       0.002193538                   0.000000000 -0.009189916
## 3                       0.011383454                   0.009189916  0.000000000
## 4                       0.018613099                   0.016419561  0.007229645
##   diff_CumErr+log_freq
## 1         -0.018613099
## 2         -0.016419561
## 3         -0.007229645
## 4          0.000000000
```

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

model	diff_CumErr+log_freq+I(pos^2)+pos	diff_CumErr+log_freq+I(pos^2)	diff_CumErr
preserved ~ CumErr+log_freq+I(pos^2)+pos	0.0000000	-0.0021935	-0.0113835
preserved ~ CumErr+log_freq+I(pos^2)	0.0021935	0.0000000	-0.0091899
preserved ~ CumErr	0.0113835	0.0091899	0.0000000
preserved ~ CumErr+log_freq	0.0186131	0.0164196	0.0072296

```
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")
```