GC - naming - 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)
```

```
# may already be done in datafile
# if(RemoveFinalPosition){
#   PosDat <- PosDat[PosDat$pos < PosDat$stimlen,] # remove final position
# }
PosDat <- PosDat[PosDat$stimlen >= MinLength,] # include only lengths with sufficient N
PosDat <- PosDat[PosDat$stimlen <= MaxLength,]
# include only correct and nonword errors (not no response, word or w+nw errors)
PosDat <- PosDat[(PosDat$err_type == "correct") | (PosDat$err_type == "nonword"), ]
PosDat <- PosDat[(PosDat$pos) != (PosDat$stimlen), ]
# make sure final positions have been removed
PosDat$stim_number <- NumberStimuli(PosDat)
PosDat$raw_log_freq <- PosDat$log_freq
PosDat$log_freq <- PosDat$log_freq - mean(PosDat$log_freq) # centre frequency
PosDat$CumErr <- CalcCumErrFromPreserved(PosDat)
```

```
# plot distribution of complex onsets and codas
# across serial position in the stimuli -- do complexities concentrate
# on one part of the serial position curve?
# syll_component = 1 is coda
# syll_component = S is satellite

# get counts of syllable components in different serial positions
syll_comp_dist_N <- PosDat %>% mutate(pos_factor = as.factor(pos), syll_component_factor = as.factor(sy

syll_comp_dist_perc <- syll_comp_dist_N %>% ungroup() %>%
  mutate(across(O:S, ~ . / total))

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	155	4	18	NA	NA	177
2	12	NA	133	9	23	177
3	70	NA	35	71	1	177
4	92	NA	42	13	10	157
5	44	NA	47	10	10	111
6	37	1	20	17	4	79
7	34	NA	9	3	1	47
8	10	NA	8	2	NA	20
9	10	NA	NA	NA	NA	10

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8757062	0.0225989	0.1016949	NA	NA	177
2	0.0677966	NA	0.7514124	0.0508475	0.1299435	177
3	0.3954802	NA	0.1977401	0.4011299	0.0056497	177
4	0.5859873	NA	0.2675159	0.0828025	0.0636943	157
5	0.3963964	NA	0.4234234	0.0900901	0.0900901	111
6	0.4683544	0.0126582	0.2531646	0.2151899	0.0506329	79

pos_factor	O	P	V	1	S	total
7	0.7234043	NA	0.1914894	0.0638298	0.0212766	47
8	0.5000000	NA	0.4000000	0.1000000	NA	20
9	1.0000000	NA	NA	NA	NA	10

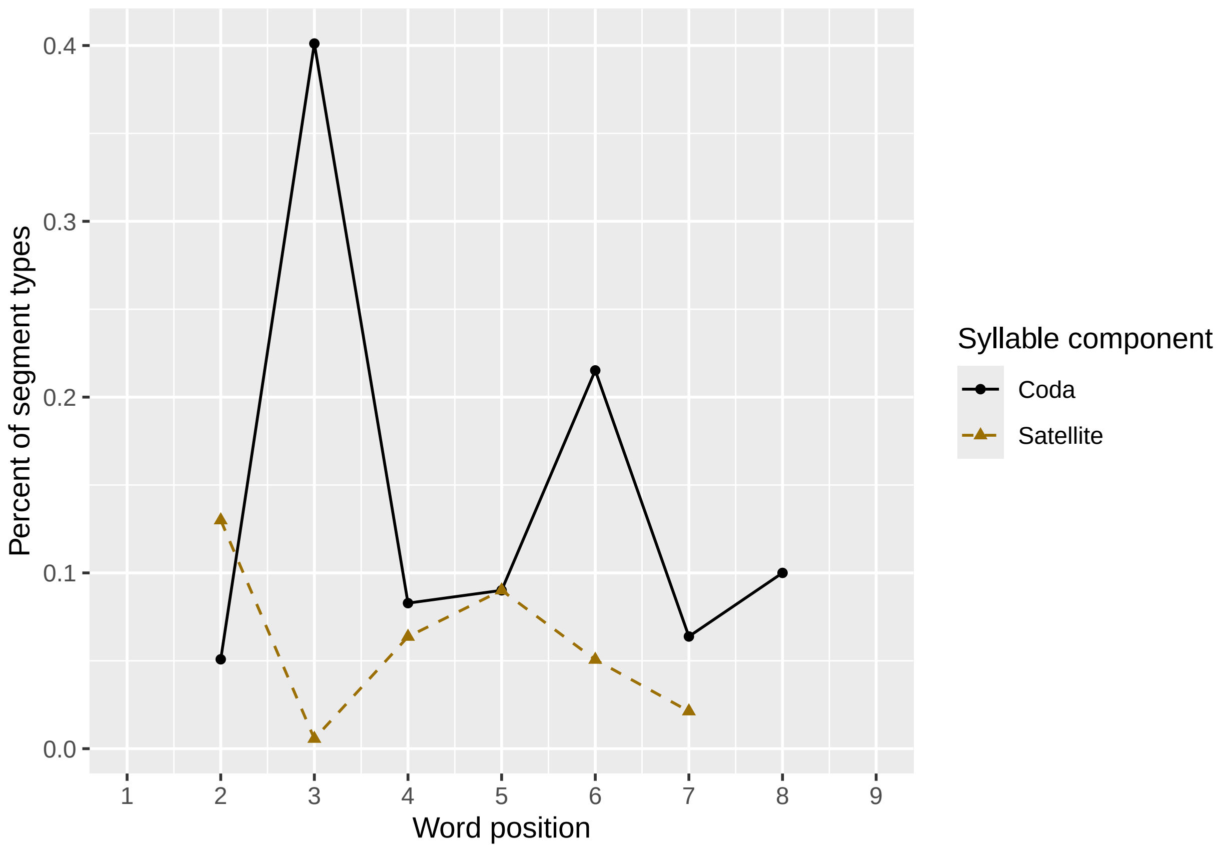
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

# 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 5 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

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

```
syll_comp_plot
```

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



```r
# 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.
```

```r
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 0.9   0.95  0.95  NA     NA     NA     NA     NA      NA
## 2       5 0.783 0.826 0.913  0.891 NA     NA     NA     NA      NA
## 3       6 0.812 0.953 0.938  0.938  0.938 NA     NA     NA      NA
## 4       7 0.875 0.969 0.906  0.938  1      0.969 NA     NA      NA
## 5       8 0.741 0.870 0.802  0.827  0.833  0.790  0.840 NA      NA
## 6       9 0.9   0.9   0.6    0.8    0.8    0.6    0.6    0.7     NA
## 7      10 0.75  0.7   0.7    0.633  0.6    0.583  0.7    0.733   0.6
```

```r
# 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.
```

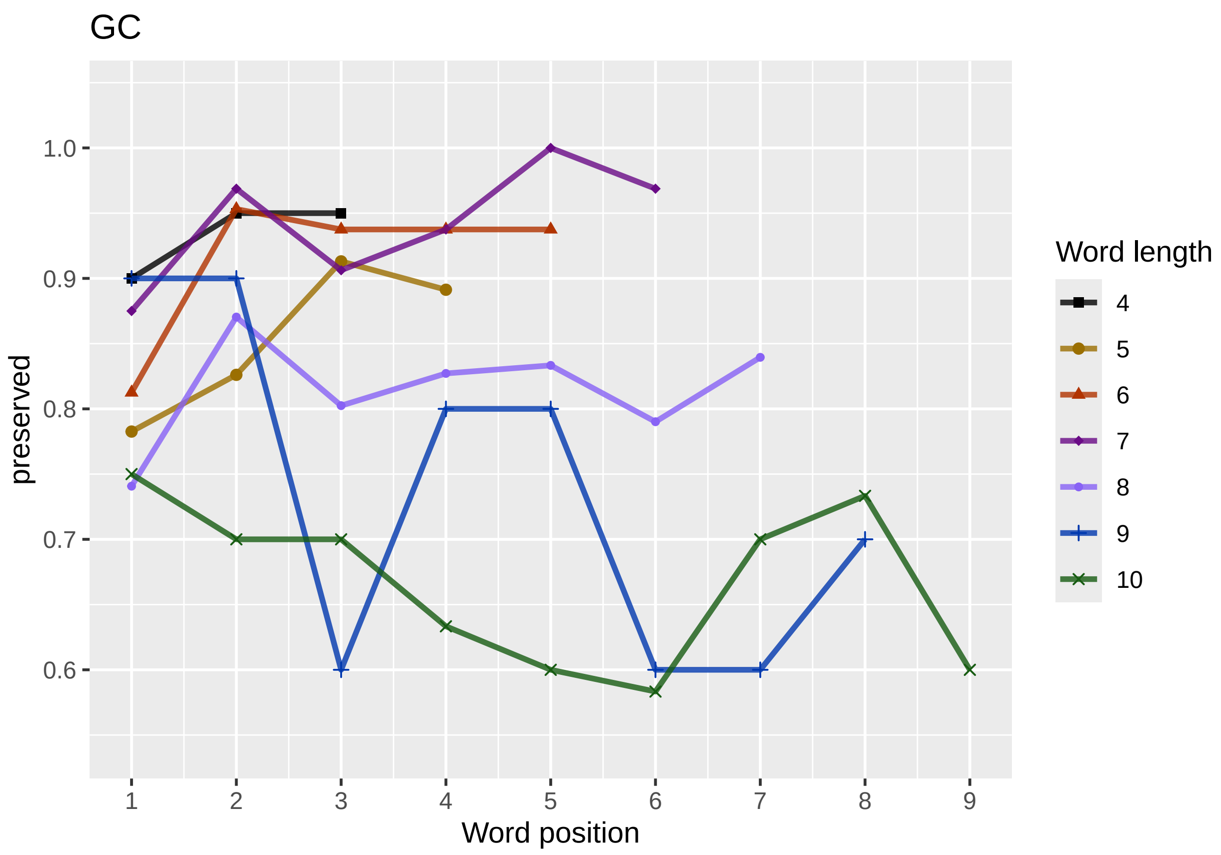
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    20    20    20    NA    NA    NA    NA    NA    NA
## 2       5    46    46    46    46    NA    NA    NA    NA    NA
## 3       6    32    32    32    32    32    NA    NA    NA    NA
## 4       7    32    32    32    32    32    32    NA    NA    NA
## 5       8    27    27    27    27    27    27    27    NA    NA
## 6       9    10    10    10    10    10    10    10    10    NA
## 7      10    10    10    10    10    10    10    10    10    10
```

```
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:  5
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.20254      0.02583      0.94501     -0.10965
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 737.1     AIC: 776.2
## log likelihood:  -368.5394
## Nagelkerke R2:  0.08152908
## % pres/err predicted correctly:  -225.0261
## % of predictable range [ (model-null)/(1-null) ]:  0.0548473
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##         -0.20930           0.21049          -0.14587           2.00303           0.01743
##      stimlen:pos
##         -0.24129
##
## Degrees of Freedom: 954 Total (i.e. Null);  949 Residual
## Null Deviance:       781.6
## Residual Deviance: 735.3     AIC: 778.6
## log likelihood:  -367.6406
## Nagelkerke R2:  0.08474055
## % pres/err predicted correctly:  -224.5309
## % of predictable range [ (model-null)/(1-null) ]:  0.05691783
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.11265     -0.28146     -0.04494      0.40047
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 746.7     AIC: 786.8
## log likelihood:  -373.343
## Nagelkerke R2:  0.06426254
## % pres/err predicted correctly:  -227.8486
## % of predictable range [ (model-null)/(1-null) ]:  0.04304453
## ***************************
## model index:  2
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.8643      -0.2938
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 752.3     AIC: 787.9
## log likelihood:  -376.1746
## Nagelkerke R2:  0.0540028
## % pres/err predicted correctly:  -229.0446
## % of predictable range [ (model-null)/(1-null) ]:  0.03804359
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.86399     -0.30769      0.02816
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 752   AIC: 789.4
## log likelihood:  -376.0057
## Nagelkerke R2:  0.05461655
## % pres/err predicted correctly:  -229.052
## % of predictable range [ (model-null)/(1-null) ]:  0.03801243
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.21654     -0.06288      0.44929
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 767.6     AIC: 809.3
## log likelihood:  -383.78
## Nagelkerke R2:  0.02614282
## % pres/err predicted correctly:  -233.8641
## % of predictable range [ (model-null)/(1-null) ]:  0.01789041
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     2.03997     -0.08165
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 778.5     AIC: 817.7
## log likelihood:  -389.2286
## Nagelkerke R2:  0.005908924
## % pres/err predicted correctly:  -237.1464
## % of predictable range [ (model-null)/(1-null) ]:  0.004164951
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.749
##
## Degrees of Freedom: 954 Total (i.e. Null);  954 Residual
## Null Deviance:       781.6
## Residual Deviance: 781.6     AIC: 819
## log likelihood:  -390.8081
## Nagelkerke R2:  1.986502e-16
## % pres/err predicted correctly:  -238.1424
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	776.1540	0.000000	1.000000	0.7635305	0.0081529	1.2025370	0.0258328	0.9450124	-0.109654	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	778.5582	2.404114	0.3005753	0.2294999	0.0847406	-0.2093028	0.2104912	0.0030335	-0.241295	-0.1458657	0.0174332

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	786.7731	10.618281	0.0049462	0.0037706	0.0642635	3.1126547	-0.2814578	0.4004720	NA	-0.0449446	NA
preserved ~ stimlen	787.8758	11.721052	0.0028497	0.0021759	0.0540038	3.8642858	-0.2937663	NA	NA	NA	NA
preserved ~ stimlen + pos	789.4066	13.251834	0.0013256	0.0010121	0.0546163	3.8639892	-0.3076918	0.0281609	NA	NA	NA
preserved ~ I(pos^2) + pos	809.3023	33.147584	0.0000001	0.0000000	0.0261412	1.2165408	NA	0.4492852	NA	-0.0628761	NA
preserved ~ pos	817.7024	41.547810	0.0000000	0.0000000	0.0059082	2.0399712	NA	-0.0816527	NA	NA	NA
preserved ~ 1	819.0194	42.864698	0.0000000	0.0000000	0.0000000	1.7490462	NA	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.20254      0.02583      0.94501     -0.10965
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 737.1     AIC: 776.2
```

```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.
```

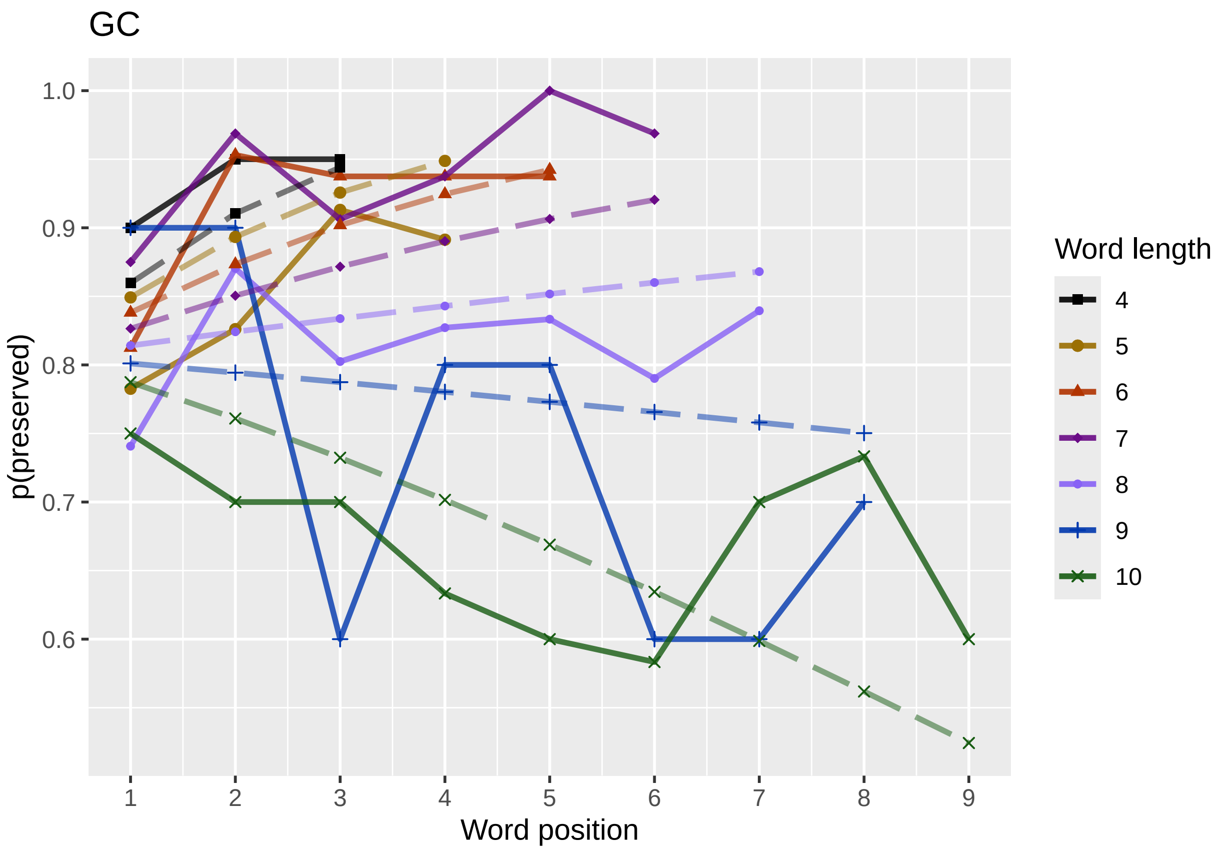
```r
fitted_pos_len_table <- fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_from = fitted)
fitted_pos_len_table
```

```
## # 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 0.860 0.910 0.944 NA     NA     NA     NA     NA     NA
## 2       5 0.849 0.893 0.926  0.949 NA     NA     NA     NA     NA
## 3       6 0.838 0.873 0.902  0.925  0.942 NA     NA     NA     NA
## 4       7 0.826 0.850 0.872  0.890  0.906  0.920 NA     NA     NA
## 5       8 0.814 0.824 0.834  0.843  0.852  0.860  0.868 NA     NA
## 6       9 0.801 0.794 0.787  0.780  0.773  0.766  0.758  0.750 NA
## 7      10 0.787 0.761 0.732  0.702  0.669  0.635  0.599  0.562  0.524
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1        6   177
```

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

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

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

```
## [1] "The number of responses with fragments was 6 / 177 = 3.39 percent"
```

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

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

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

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

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

```
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.38186     -0.01351      0.82500     -0.08478
##
## Degrees of Freedom: 931 Total (i.e. Null);  928 Residual
## Null Deviance:       701.9
## Residual Deviance: 676.8     AIC: 711.5
## log likelihood:  -338.3933
## Nagelkerke R2:  0.05018813
## % pres/err predicted correctly:  -200.7342
## % of predictable range [ (model-null)/(1-null) ]:  0.0305889
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          0.22054           0.15049          -0.11412           1.67635           0.01474
##      stimlen:pos
##         -0.19957
##
## Degrees of Freedom: 931 Total (i.e. Null);  926 Residual
## Null Deviance:       701.9
## Residual Deviance: 675.6     AIC: 714.2
## log likelihood:  -337.7772
## Nagelkerke R2:  0.05261882
## % pres/err predicted correctly:  -200.3381
## % of predictable range [ (model-null)/(1-null) ]:  0.03249243
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      3.2719      -0.2548       0.1315
##
## Degrees of Freedom: 931 Total (i.e. Null);  929 Residual
## Null Deviance:       701.9
## Residual Deviance: 684.1     AIC: 716.6
## log likelihood:  -342.0446
## Nagelkerke R2:  0.03571532
## % pres/err predicted correctly:  -202.2664
## % of predictable range [ (model-null)/(1-null) ]:  0.02322624
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     2.83077     -0.24018     -0.03049      0.36978
##
## Degrees of Freedom: 931 Total (i.e. Null);  928 Residual
## Null Deviance:       701.9
## Residual Deviance: 682.3     AIC: 717.2
## log likelihood:  -341.1454
## Nagelkerke R2:  0.03928984
## % pres/err predicted correctly:  -202.051
## % of predictable range [ (model-null)/(1-null) ]:  0.02426125
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.3238      -0.2002
##
## Degrees of Freedom: 931 Total (i.e. Null);  930 Residual
## Null Deviance:       701.9
## Residual Deviance: 690   AIC: 721.1
## log likelihood:  -345.0049
## Nagelkerke R2:  0.0238981
## % pres/err predicted correctly:  -203.762
## % of predictable range [ (model-null)/(1-null) ]:  0.01603949
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.20884     -0.04701      0.42163
##
## Degrees of Freedom: 931 Total (i.e. Null);  929 Residual
## Null Deviance:       701.9
## Residual Deviance: 696.5     AIC: 732.2
## log likelihood:  -348.2728
## Nagelkerke R2:  0.01076532
## % pres/err predicted correctly:  -205.8612
## % of predictable range [ (model-null)/(1-null) ]:  0.005951651
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.91
```

```
##
## Degrees of Freedom: 931 Total (i.e. Null);  931 Residual
## Null Deviance:       701.9
## Residual Deviance: 701.9     AIC: 733.1
## log likelihood:  -350.9346
## Nagelkerke R2:  -4.196763e-16
## % pres/err predicted correctly:  -207.0998
## % 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
##     1.74388      0.05031
##
## Degrees of Freedom: 931 Total (i.e. Null);  930 Residual
## Null Deviance:       701.9
## Residual Deviance: 700.9     AIC: 734.2
## log likelihood:  -350.4749
## Nagelkerke R2:  0.001863568
## % pres/err predicted correctly:  -206.8599
## % of predictable range [ (model-null)/(1-null) ]:  0.001152943
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	711.4952	0.000000	1.0000000	0.7128710	0.0501881	1.381865	-0.0135107	0.8249965	-0.0847774	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	714.1877	2.692456	0.2602200	0.1855035	0.0526188	2.220537	0.1504911	1.6763522	-0.1995729	-0.1141218	0.0147406
preserved ~ stimlen + pos	716.5965	5.101306	0.0780307	0.0556259	0.0357153	3.271923	-0.2548318	0.1315293	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	717.2485	5.753363	0.0563210	0.0401499	0.0392892	2.830770	-0.2401762	0.3697835	NA	-0.0304932	NA
preserved ~ stimlen	721.1179	9.621798	0.0081405	0.0058032	0.0238983	3.323808	-0.2001787	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	732.2062	20.711643	0.0000318	0.0000227	0.0107651	3.208841	NA	0.4216334	NA	-0.0470081	NA
preserved ~ 1	733.0854	21.590146	0.0000205	0.0000146	0.0000000	0.910415	NA	NA	NA	NA	NA
preserved ~ pos	734.1632	22.668410	0.0000120	0.0000085	0.0018636	1.743883	NA	0.0503056	NA	NA	NA

```r
# 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.
```

```r
# 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.
```

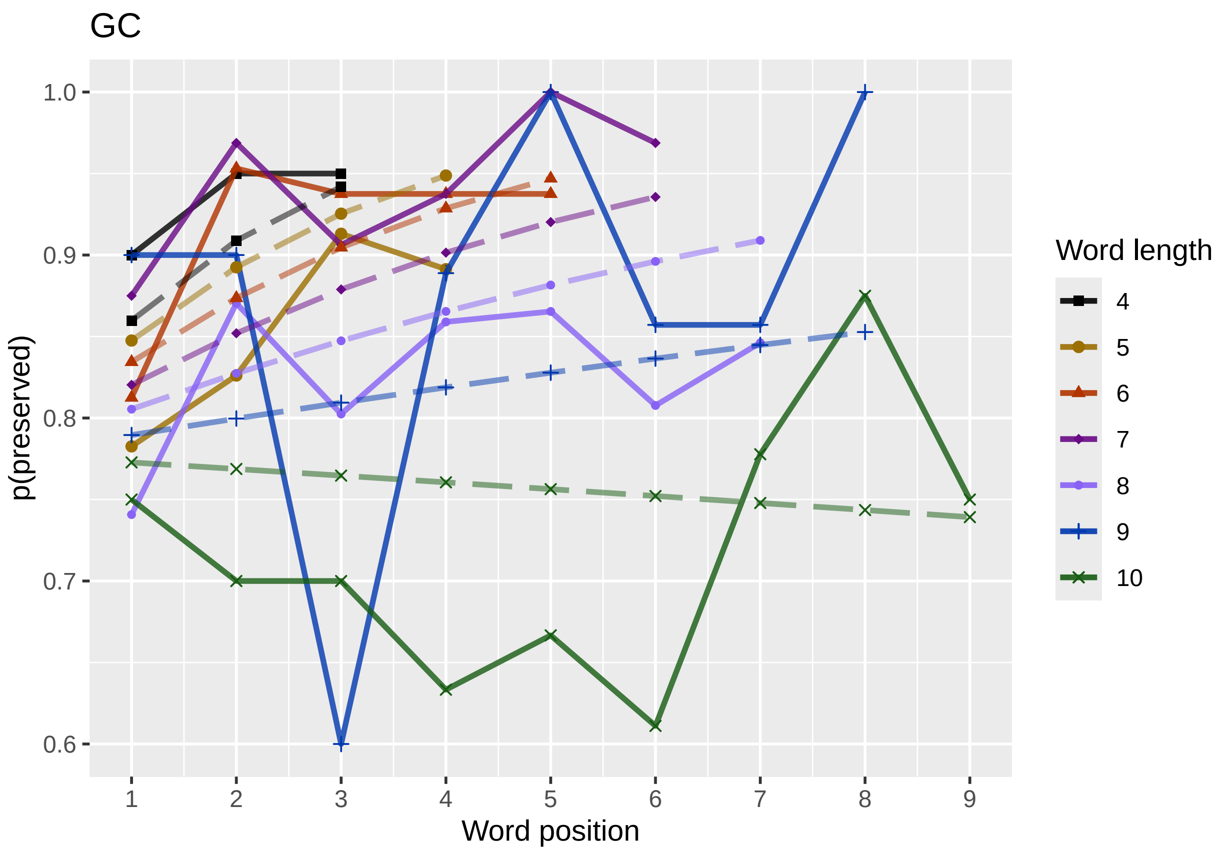
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.860 0.909 0.942 NA     NA     NA     NA     NA     NA
## 2       5 0.848 0.892 0.925  0.949 NA     NA     NA     NA     NA
## 3       6 0.834 0.874 0.905  0.929  0.947 NA     NA     NA     NA
## 4       7 0.820 0.852 0.879  0.901  0.920  0.936 NA     NA     NA
## 5       8 0.805 0.827 0.847  0.865  0.882  0.896  0.909 NA     NA
## 6       9 0.790 0.800 0.809  0.819  0.828  0.836  0.845  0.853 NA
## 7      10 0.773 0.769 0.765  0.761  0.756  0.752  0.748  0.744  0.739
```

```r
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: 0.54 - 1.04"
```

```
# 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
```

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

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

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

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

```r
print(OA_mean_len_diff)
```

```
## [1] -0.04324557
```

```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] 0.01827111
```

```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:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           2.11348           -0.16169           -0.05879            0.56390           -0.18971
## I(pos^2):log_freq       pos:log_freq
##          -0.02678            0.29699
##
## Degrees of Freedom: 954 Total (i.e. Null);  948 Residual
## Null Deviance:       781.6
## Residual Deviance: 702.4     AIC: 746.6
## log likelihood:  -351.2073
## Nagelkerke R2:  0.1424049
## % pres/err predicted correctly:  -214.9702
## % of predictable range [ (model-null)/(1-null) ]:  0.09689705
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##           2.01423           -0.14220           -0.35412           -0.05983            0.56236
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##           0.02618           -0.02777            0.29495
##
```

```
## Degrees of Freedom: 954 Total (i.e. Null);  947 Residual
## Null Deviance:       781.6
## Residual Deviance: 702   AIC: 748.2
## log likelihood:  -351.0217
## Nagelkerke R2:  0.143045
## % pres/err predicted correctly:  -214.5551
## % of predictable range [ (model-null)/(1-null) ]:  0.09863273
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.83554      -0.18350       0.12715       0.04861       0.10095
##
## Degrees of Freedom: 954 Total (i.e. Null);  950 Residual
## Null Deviance:       781.6
## Residual Deviance: 708.2     AIC: 748.3
## log likelihood:  -354.0991
## Nagelkerke R2:  0.1324011
## % pres/err predicted correctly:  -215.9062
## % of predictable range [ (model-null)/(1-null) ]:  0.09298293
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.78400          -0.17256          -0.04433           0.12264           0.01557
##    log_freq:pos
##         0.09554
##
## Degrees of Freedom: 954 Total (i.e. Null);  949 Residual
## Null Deviance:       781.6
## Residual Deviance: 708.1     AIC: 750.2
## log likelihood:  -354.0312
## Nagelkerke R2:  0.1326366
## % pres/err predicted correctly:  -215.639
## % of predictable range [ (model-null)/(1-null) ]:  0.09410051
## ***************************
## 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
##          1.02612           -0.06970            0.59738           -0.12055           -0.02766
##     pos:log_freq
##          0.29633
```

```
##
## Degrees of Freedom: 954 Total (i.e. Null);  949 Residual
## Null Deviance:       781.6
## Residual Deviance: 708.4     AIC: 752.2
## log likelihood:  -354.1925
## Nagelkerke R2:  0.1320769
## % pres/err predicted correctly:  -217.3121
## % of predictable range [ (model-null)/(1-null) ]:  0.0871044
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     2.29106     -0.14908     -0.04553      0.40685      0.39596
##
## Degrees of Freedom: 954 Total (i.e. Null);  950 Residual
## Null Deviance:       781.6
## Residual Deviance: 712.5     AIC: 753.1
## log likelihood:  -356.2506
## Nagelkerke R2:  0.1249188
## % pres/err predicted correctly:  -218.1323
## % of predictable range [ (model-null)/(1-null) ]:  0.08367428
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##         2.095676         -0.111230          0.004161         -0.043531          0.390359
## stimlen:log_freq
##         0.054650
##
## Degrees of Freedom: 954 Total (i.e. Null);  949 Residual
## Null Deviance:       781.6
## Residual Deviance: 710.5     AIC: 753.1
## log likelihood:  -355.2546
## Nagelkerke R2:  0.1283866
## % pres/err predicted correctly:  -216.7337
## % of predictable range [ (model-null)/(1-null) ]:  0.08952294
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.78752          -0.11619          -0.04885           0.06176
##
```

```
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 715.6     AIC: 753.6
## log likelihood:  -357.7932
## Nagelkerke R2:  0.1195331
## % pres/err predicted correctly:  -217.3027
## % of predictable range [ (model-null)/(1-null) ]:  0.08714332
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      3.0511      -0.1609       0.3947
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 718.1     AIC: 754.1
## log likelihood:  -359.0561
## Nagelkerke R2:  0.115111
## % pres/err predicted correctly:  -218.9233
## % of predictable range [ (model-null)/(1-null) ]:  0.08036691
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen          log_freq               pos  stimlen:log_freq
##         2.78663         -0.13094          -0.04941           0.03006           0.06187
##
## Degrees of Freedom: 954 Total (i.e. Null);  950 Residual
## Null Deviance:       781.6
## Residual Deviance: 715.2     AIC: 755.1
## log likelihood:  -357.6129
## Nagelkerke R2:  0.1201635
## % pres/err predicted correctly:  -217.3892
## % of predictable range [ (model-null)/(1-null) ]:  0.08678171
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.05059     -0.17544      0.02945      0.39491
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 717.8     AIC: 755.6
## log likelihood:  -358.8795
## Nagelkerke R2:  0.1157302
## % pres/err predicted correctly:  -219.0008
## % of predictable range [ (model-null)/(1-null) ]:  0.08004285
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)          pos      log_freq  pos:log_freq
##      1.73705      0.06964       0.12023       0.09836
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 716.3     AIC: 756.2
## log likelihood:  -358.1685
## Nagelkerke R2:  0.1182202
## % pres/err predicted correctly:  -218.8478
## % of predictable range [ (model-null)/(1-null) ]:  0.08068232
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     2.00005     -0.02472      0.45406
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 725.3     AIC: 762.9
## log likelihood:  -362.6489
## Nagelkerke R2:  0.1024668
## % pres/err predicted correctly:  -221.7553
## % of predictable range [ (model-null)/(1-null) ]:  0.06852461
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     1.20254      0.02583      0.94501     -0.10965
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 737.1     AIC: 776.2
## log likelihood:  -368.5394
## Nagelkerke R2:  0.08152908
## % pres/err predicted correctly:  -225.0261
## % of predictable range [ (model-null)/(1-null) ]:  0.0548473
## ***************************
## 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)
##        -0.20930           0.21049          -0.14587           2.00303           0.01743
##     stimlen:pos
##        -0.24129
##
## Degrees of Freedom: 954 Total (i.e. Null);  949 Residual
## Null Deviance:       781.6
## Residual Deviance: 735.3     AIC: 778.6
## log likelihood:  -367.6406
## Nagelkerke R2:  0.08474055
## % pres/err predicted correctly:  -224.5309
## % of predictable range [ (model-null)/(1-null) ]:  0.05691783
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.11265     -0.28146     -0.04494      0.40047
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 746.7     AIC: 786.8
## log likelihood:  -373.343
## Nagelkerke R2:  0.06426254
## % pres/err predicted correctly:  -227.8486
## % of predictable range [ (model-null)/(1-null) ]:  0.04304453
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.8643      -0.2938
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 752.3     AIC: 787.9
## log likelihood:  -376.1746
## Nagelkerke R2:  0.0540028
## % pres/err predicted correctly:  -229.0446
## % of predictable range [ (model-null)/(1-null) ]:  0.03804359
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.86399     -0.30769      0.02816
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 752   AIC: 789.4
## log likelihood:  -376.0057
## Nagelkerke R2:  0.05461655
## % pres/err predicted correctly:  -229.052
## % of predictable range [ (model-null)/(1-null) ]:  0.03801243
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.21654     -0.06288      0.44929
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 767.6     AIC: 809.3
## log likelihood:  -383.78
## Nagelkerke R2:  0.02614282
## % pres/err predicted correctly:  -233.8641
## % of predictable range [ (model-null)/(1-null) ]:  0.01789041
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.03997     -0.08165
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 778.5     AIC: 817.7
## log likelihood:  -389.2286
## Nagelkerke R2:  0.005908924
## % pres/err predicted correctly:  -237.1464
## % of predictable range [ (model-null)/(1-null) ]:  0.004164951
## ***************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.749
##
## Degrees of Freedom: 954 Total (i.e. Null);  954 Residual
## Null Deviance:       781.6
## Residual Deviance: 781.6     AIC: 819
## log likelihood:  -390.8081
## Nagelkerke R2:  1.986502e-16
## % pres/err predicted correctly:  -238.1424
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC DeltaAICAICexpAICwtNagR2(Intercept)	stimlen log_freq	stimlen:log_freq pos	log_freq:pos I(pos^2)	I(pos^2):log_freq	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	746.60.2100.000.00424845224014934800	- NA 0.16108581 97131	0.5638929369926	- - 0.058708267818	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	748.20.5030.3248648985214320.5042298	- 0.0261794623570 0.1422035041215	0.2949509	NA - 0.0598343 0.0277741	NA	NA	NA
preserved ~ stimlen + pos * log_freq	748.3427405547188314953296240835373	0.0486093 0.1835046	0.12704730945A3	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	750.13.77955196566733015523783999 65	- 0.015572022683 0.172560843319	0.0955387	NA	NA	NA	NA

Model	AIC	DeltaAIC	expAIC	AICw	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1205452	NA	[illegible]	[illegible]	[illegible]	-0.0696[illegible]	-0.0727664[illegible]	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1490842	0.3959[illegible]	NA	0.4068[illegible]	NA	-0.0455256	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1112299	0.0040[illegible]	[illegible]	[illegible]	NA	-0.0435307	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1160[illegible]	0.0617[illegible]	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1609196	0.3947208	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1300[illegible]	0.0618[illegible]	[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1754426	0.3949[illegible]	NA	0.0294[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.1202[illegible]	NA	0.0696[illegible]	[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.4540[illegible]	NA	-0.0247210	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.9450[illegible]	NA	NA	NA	NA	NA	-0.109654	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	-0.2093028	0.2104[illegible]	NA	2.0030[illegible]	NA	NA	-0.1458657	NA	NA	-0.241295	0.0174332
preserved ~ stimlen + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2814578	NA	NA	0.4004[illegible]	NA	-0.0449446	NA	NA	NA	NA
preserved ~ stimlen	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2937663	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.3076918	NA	NA	0.0281[illegible]	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	0.4492[illegible]	NA	-0.0628761	NA	NA	NA	NA
preserved ~ pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	NA	-0.0816527	NA	NA	NA	NA	NA	NA

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

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           2.11348           -0.16169           -0.05879            0.56390           -0.18971
## I(pos^2):log_freq       pos:log_freq
##          -0.02678            0.29699
##
## Degrees of Freedom: 954 Total (i.e. Null);  948 Residual
## Null Deviance:       781.6
## Residual Deviance: 702.4     AIC: 746.6
```

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

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

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

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

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

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

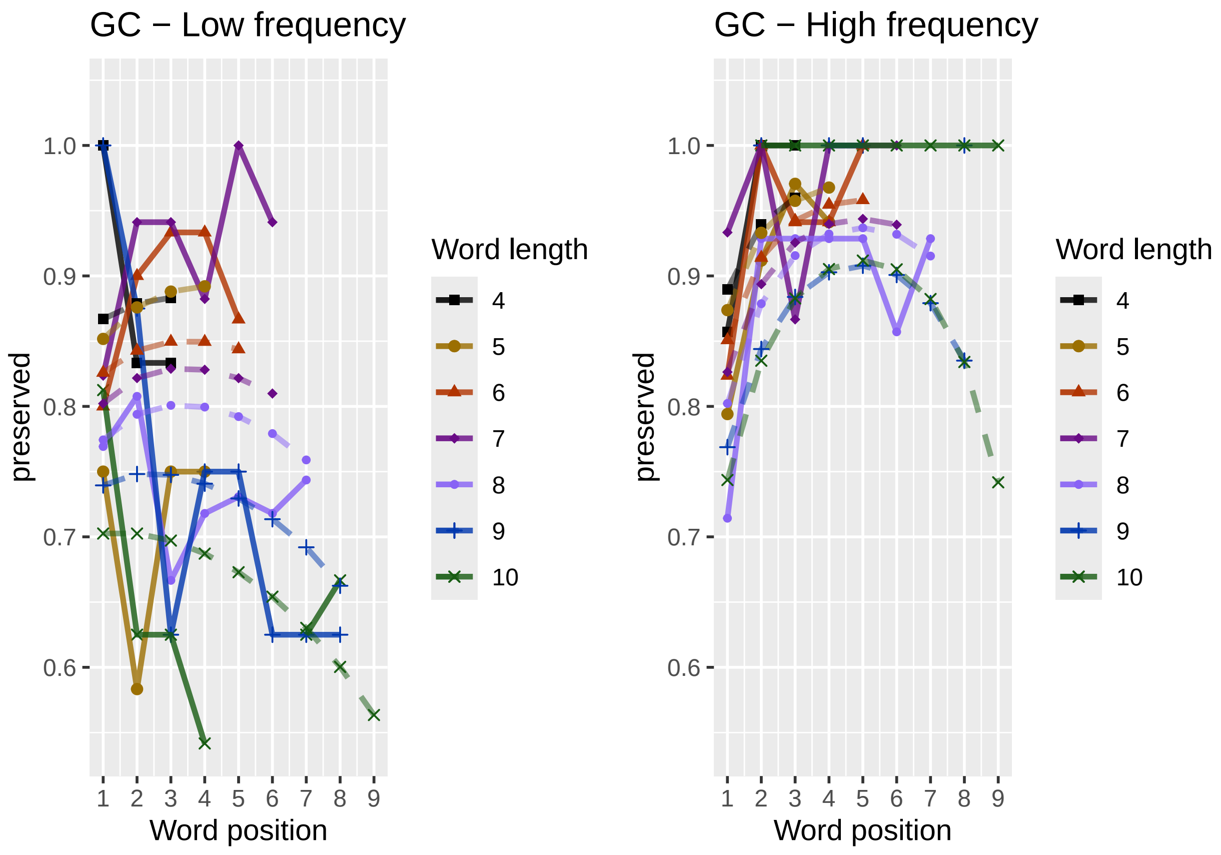
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale 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 3 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 5 rows containing missing values or values outside the scale range
## (`geom_point()`).

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

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



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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3846      -0.9931
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 637.8     AIC: 666.6
## log likelihood:  -318.8842
## Nagelkerke R2:  0.2501965
## % pres/err predicted correctly:  -185.3615
## % of predictable range [ (model-null)/(1-null) ]:  0.2207091
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.1208       0.3827
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 741.2     AIC: 778.2
## log likelihood:  -370.6031
## Nagelkerke R2:  0.0741326
## % pres/err predicted correctly:  -228.3142
## % of predictable range [ (model-null)/(1-null) ]:  0.04109785
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.8643      -0.2938
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 752.3     AIC: 787.9
## log likelihood:  -376.1746
## Nagelkerke R2:  0.0540028
## % pres/err predicted correctly:  -229.0446
## % of predictable range [ (model-null)/(1-null) ]:  0.03804359
## ***************************
## model index:  3
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.21654     -0.06288      0.44929
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 767.6     AIC: 809.3
## log likelihood:  -383.78
## Nagelkerke R2:  0.02614282
## % pres/err predicted correctly:  -233.8641
## % of predictable range [ (model-null)/(1-null) ]:  0.01789041
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.03997     -0.08165
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 778.5     AIC: 817.7
## log likelihood:  -389.2286
## Nagelkerke R2:  0.005908924
## % pres/err predicted correctly:  -237.1464
## % of predictable range [ (model-null)/(1-null) ]:  0.004164951
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.749
##
## Degrees of Freedom: 954 Total (i.e. Null);  954 Residual
## Null Deviance:       781.6
## Residual Deviance: 781.6     AIC: 819
## log likelihood:  -390.8081
## Nagelkerke R2:  1.986502e-16
## % pres/err predicted correctly:  -238.1424
## % 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	666.5611	0.0000	1	1	0.2501962	3.384593	NA	-0.9930642	NA	NA	NA
preserved ~ CumPres	778.2287	111.6676	0	0	0.0741326	1.120767	0.3826592	NA	NA	NA	NA
preserved ~ stimlen	787.8758	121.3147	0	0	0.0540028	3.864286	NA	NA	NA	NA	-0.2937663
preserved ~ (I(pos^2) + pos)	809.3023	142.7412	0	0	0.0261428	1.216541	NA	NA	-0.0628761	0.4492852	NA
preserved ~ pos	817.7026	151.1415	0	0	0.0059089	2.039971	NA	NA	NA	-0.0816527	NA
preserved ~ 1	819.0194	152.4584	0	0	0.0000000	1.749046	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	0.8293333	125
O	0.8232759	464
P	0.8000000	5
S	0.8775510	49
V	0.9001068	312

```
#  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
##       2.347       -1.006
##
## Degrees of Freedom: 900 Total (i.e. Null);  899 Residual
## Null Deviance:       739.8
## Residual Deviance: 610   AIC: 638.2
## log likelihood:  -304.983
## Nagelkerke R2:  0.2396224
## % pres/err predicted correctly:  -177.9819
## % of predictable range [ (model-null)/(1-null) ]:  0.2111978
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##       1.141        0.385
##
## Degrees of Freedom: 900 Total (i.e. Null);  899 Residual
## Null Deviance:       739.8
## Residual Deviance: 703.8     AIC: 741
## log likelihood:  -351.8891
## Nagelkerke R2:  0.06998098
## % pres/err predicted correctly:  -217.2781
## % of predictable range [ (model-null)/(1-null) ]:  0.03801326
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.7815      -0.2837
##
## Degrees of Freedom: 900 Total (i.e. Null);  899 Residual
## Null Deviance:       739.8
## Residual Deviance: 713.8     AIC: 749.4
## log likelihood:  -356.9015
## Nagelkerke R2:  0.05078628
## % pres/err predicted correctly:  -217.7738
## % of predictable range [ (model-null)/(1-null) ]:  0.03582853
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.20790     -0.06246      0.44861
##
## Degrees of Freedom: 900 Total (i.e. Null);  898 Residual
## Null Deviance:       739.8
## Residual Deviance: 726.3     AIC: 768
## log likelihood:  -363.1663
## Nagelkerke R2:  0.02649341
## % pres/err predicted correctly:  -221.7836
## % of predictable range [ (model-null)/(1-null) ]:  0.01815681
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.02799     -0.08059
##
## Degrees of Freedom: 900 Total (i.e. Null);  899 Residual
## Null Deviance:       739.8
## Residual Deviance: 736.8     AIC: 776
## log likelihood:  -368.4147
## Nagelkerke R2:  0.005879843
## % pres/err predicted correctly:  -224.9583
## % of predictable range [ (model-null)/(1-null) ]:  0.004165172
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.74
##
## Degrees of Freedom: 900 Total (i.e. Null);  900 Residual
## Null Deviance:       739.8
## Residual Deviance: 739.8     AIC: 777.1
## log likelihood:  -369.9007
## Nagelkerke R2:  -3.964744e-16
## % pres/err predicted correctly:  -225.9034
## % 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	638.2040	0.0000	1	1	0.2396222	2.346855	NA	-1.006142	NA	NA	NA
preserved ~ CumPres	741.0028	102.7987	0	0	0.0699810	1.140957	0.3850491	NA	NA	NA	NA
preserved ~ stimlen	749.3662	111.1622	0	0	0.0507863	3.781454	NA	NA	NA	NA	-0.2837381
preserved ~ (I(pos^2) + pos)	768.0419	129.8378	0	0	0.0264934	1.207901	NA	NA	-0.0624566	0.4486121	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ pos	776.0031	137.7991	0	0	0.0058798	2.027985	NA	NA	NA	-0.0805869	NA
preserved ~ 1	777.1255	138.9215	0	0	0.0000000	1.740271	NA	NA	NA	NA	NA

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.281       -1.055
##
## Degrees of Freedom: 775 Total (i.e. Null);  774 Residual
## Null Deviance:       627.6
## Residual Deviance: 538.9     AIC: 564.1
## log likelihood:  -269.446
## Nagelkerke R2:  0.1947886
## % pres/err predicted correctly:  -157.8363
## % of predictable range [ (model-null)/(1-null) ]:  0.1711084
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.6655      -0.2654
```

```
##
## Degrees of Freedom: 775 Total (i.e. Null);  774 Residual
## Null Deviance:       627.6
## Residual Deviance: 607.8     AIC: 639.2
## log likelihood:  -303.9024
## Nagelkerke R2:  0.04542636
## % pres/err predicted correctly:  -184.3961
## % of predictable range [ (model-null)/(1-null) ]:  0.03250568
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.2923       0.3526
##
## Degrees of Freedom: 775 Total (i.e. Null);  774 Residual
## Null Deviance:       627.6
## Residual Deviance: 607.2     AIC: 640.7
## log likelihood:  -303.5924
## Nagelkerke R2:  0.04683032
## % pres/err predicted correctly:  -185.5997
## % of predictable range [ (model-null)/(1-null) ]:  0.0262247
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.20532     -0.06521      0.47694
##
## Degrees of Freedom: 775 Total (i.e. Null);  773 Residual
## Null Deviance:       627.6
## Residual Deviance: 615.1     AIC: 652
## log likelihood:  -307.5414
## Nagelkerke R2:  0.02886388
## % pres/err predicted correctly:  -186.7896
## % of predictable range [ (model-null)/(1-null) ]:  0.02001491
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.02479     -0.07309
##
## Degrees of Freedom: 775 Total (i.e. Null);  774 Residual
## Null Deviance:       627.6
## Residual Deviance: 625.4     AIC: 659.8
## log likelihood:  -312.692
## Nagelkerke R2:  0.005153813
## % pres/err predicted correctly:  -189.9064
## % of predictable range [ (model-null)/(1-null) ]:  0.003749931
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.768
##
## Degrees of Freedom: 775 Total (i.e. Null);  775 Residual
## Null Deviance:       627.6
## Residual Deviance: 627.6     AIC: 660.3
## log likelihood:  -313.8026
## Nagelkerke R2:  2.001865e-16
## % pres/err predicted correctly:  -190.625
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

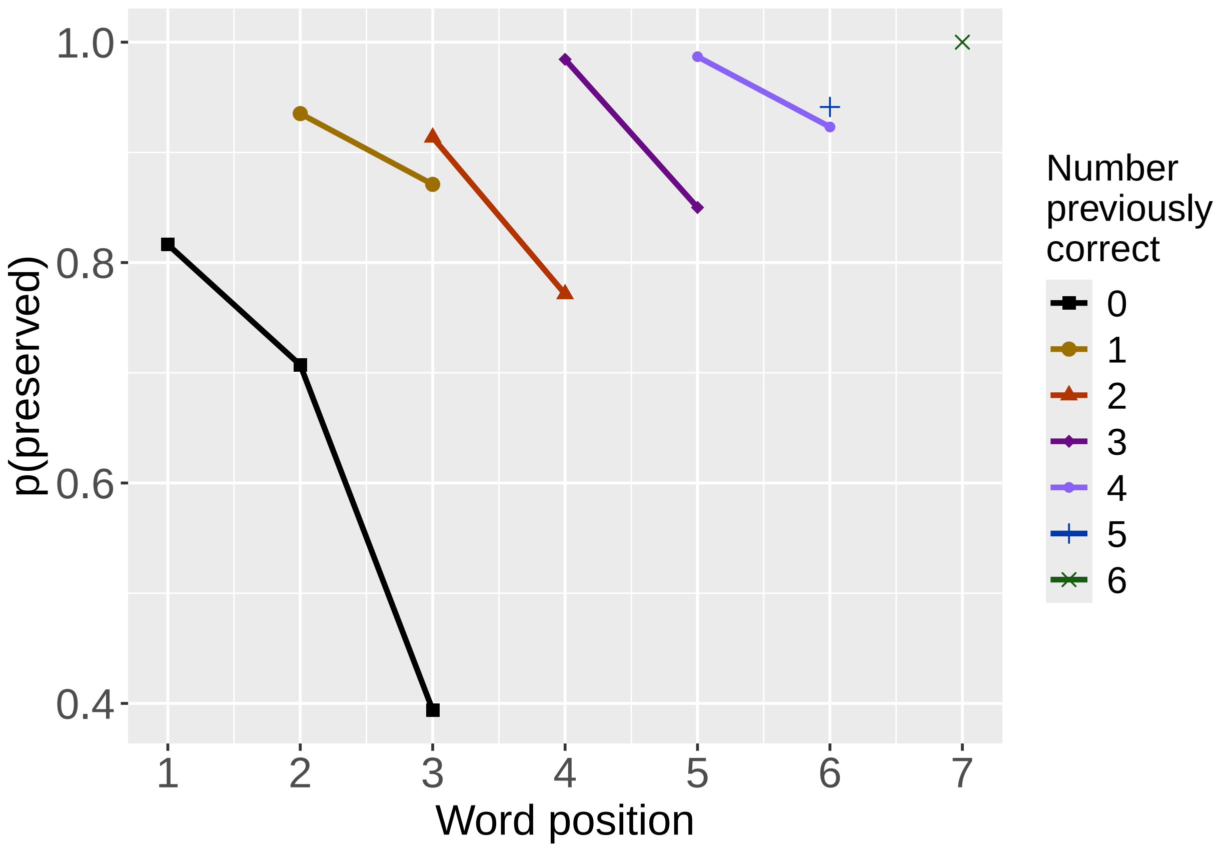
```r
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	564.1380	0.00000	1	1	0.1947882	2.281016	NA	-1.054532	NA	NA	NA
preserved ~ stimlen	639.1522	75.01420	0	0	0.0454264	1.665503	NA	NA	NA	NA	-0.2654032
preserved ~ CumPres	640.6760	76.53794	0	0	0.0468303	1.292252	0.3526083	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos)	652.0095	87.87150	0	0	0.0288639	1.205323	NA	NA	-0.0652088	0.4769383	NA
preserved ~ pos	659.8304	95.69239	0	0	0.0051538	2.024793	NA	NA	NA	-0.0730909	NA
preserved ~ 1	660.2730	96.13498	0	0	0.0000000	1.767662	NA	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.
```

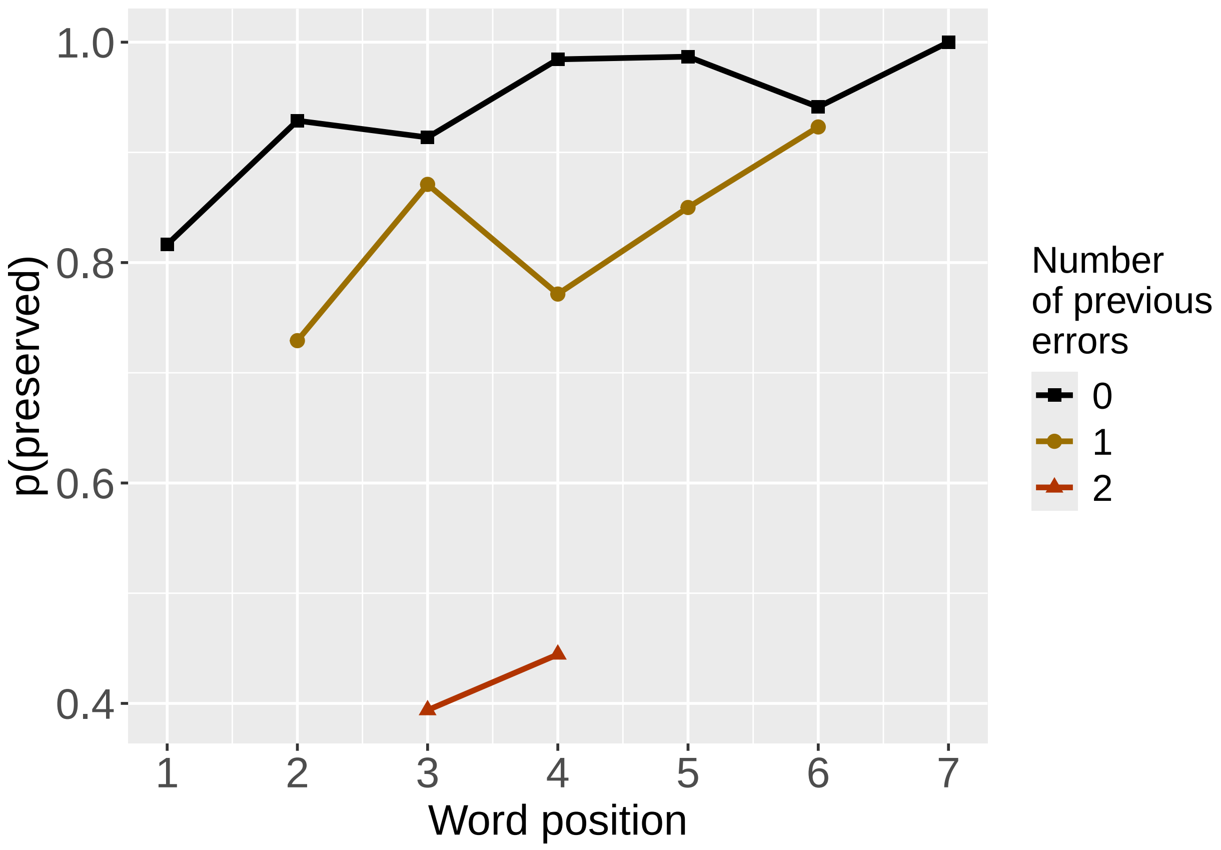
```r
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

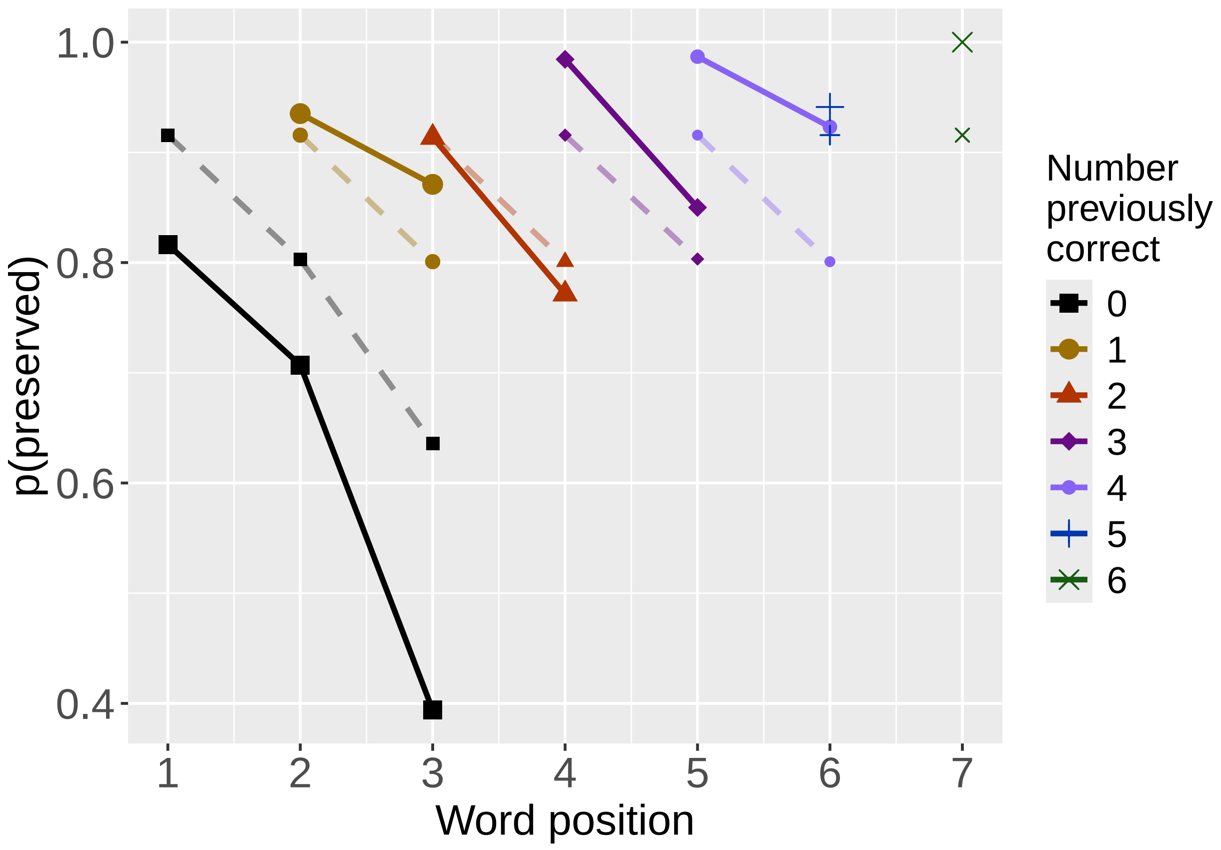
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

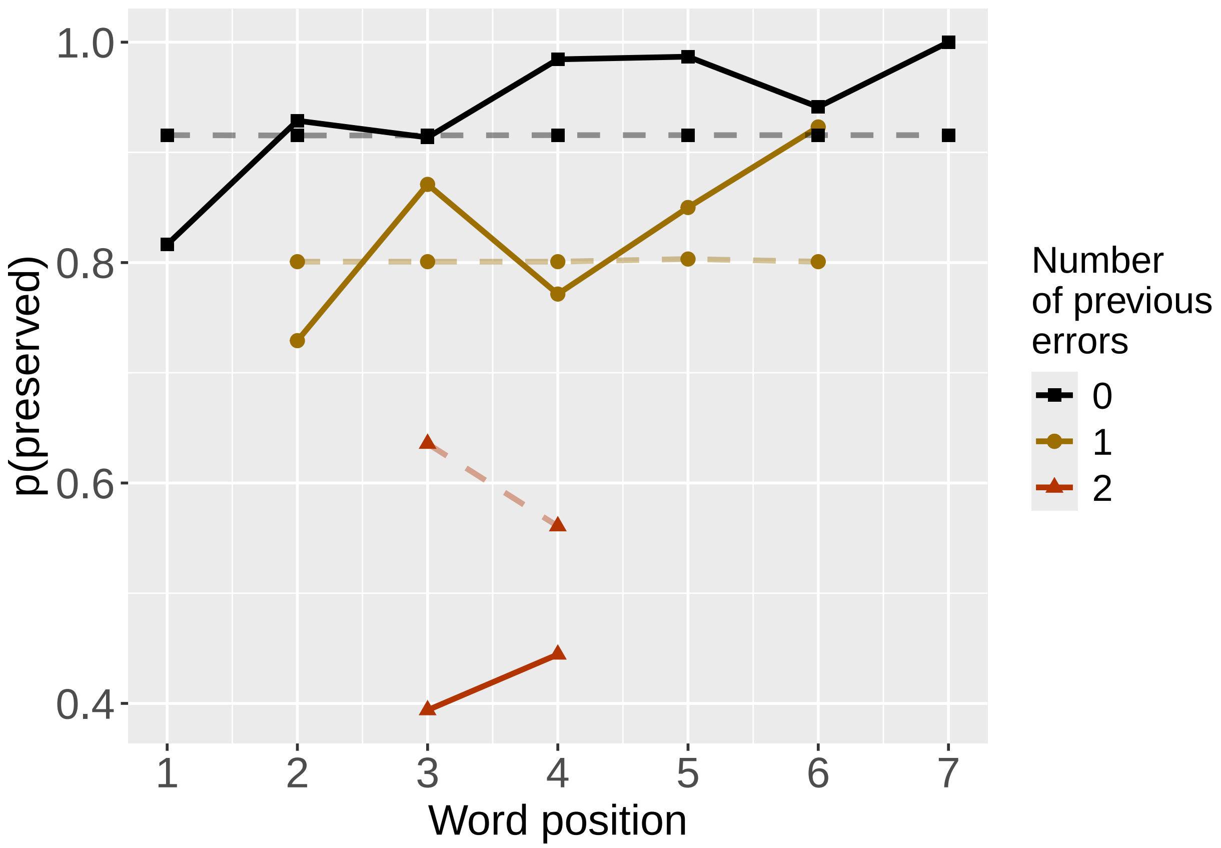
```
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     1.01645     -1.34774     -0.03642      0.64512
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 606   AIC: 636.7
## log likelihood:  -303.0075
## Nagelkerke R2:  0.300529
## % pres/err predicted correctly:  -178.2075
## % of predictable range [ (model-null)/(1-null) ]:  0.2506242
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3846      -0.9931
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 637.8     AIC: 666.6
## log likelihood:  -318.8842
## Nagelkerke R2:  0.2501965
## % pres/err predicted correctly:  -185.3615
## % of predictable range [ (model-null)/(1-null) ]:  0.2207091
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     1.21654     -0.06288      0.44929
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 767.6     AIC: 809.3
## log likelihood:  -383.78
## Nagelkerke R2:  0.02614282
## % pres/err predicted correctly:  -233.8641
## % of predictable range [ (model-null)/(1-null) ]:  0.01789041
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	636.7211	0.00000	1e+00	0.9999997	0.3005290	1.016449	-1.3477359	-0.0364164	0.6451218

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	666.5611	29.84002	3e-07	0.0000003	0.2501965	2.384593	-0.9930642	NA	NA
preserved ~ I(pos^2) + pos	809.3023	172.58127	0e+00	0.0000000	0.0261428	1.216541	NA	-0.0628761	0.4492852

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     3.05958     -0.94797     -0.09932
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 635.2     AIC: 665.1
## log likelihood:  -317.6218
## Nagelkerke R2:  0.2542601
## % pres/err predicted correctly:  -184.3516
## % of predictable range [ (model-null)/(1-null) ]:  0.2249323
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3846      -0.9931
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 637.8     AIC: 666.6
## log likelihood:  -318.8842
## Nagelkerke R2:  0.2501965
## % pres/err predicted correctly:  -185.3615
## % of predictable range [ (model-null)/(1-null) ]:  0.2207091
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.8643      -0.2938
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 752.3     AIC: 787.9
## log likelihood:  -376.1746
## Nagelkerke R2:  0.0540028
## % pres/err predicted correctly:  -229.0446
## % of predictable range [ (model-null)/(1-null) ]:  0.03804359
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	665.1057	0.000000	1.000000	0.6742961	0.2542601	3.059585	-0.9479663	-0.0993218
preserved ~ CumErr	666.5611	1.455361	0.483028	0.3257039	0.2501965	2.384593	-0.9930642	NA
preserved ~ stimlen	787.8758	122.770072	0.000000	0.0000000	0.0540028	3.864286	NA	-0.2937663

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr       CumPres
##      1.7629      -0.9839        0.3737
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 607.9     AIC: 636.4
## log likelihood:  -303.9395
## Nagelkerke R2:  0.2976206
## % pres/err predicted correctly:  -178.6663
## % of predictable range [ (model-null)/(1-null) ]:  0.2487057
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3846      -0.9931
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 637.8     AIC: 666.6
## log likelihood:  -318.8842
## Nagelkerke R2:  0.2501965
## % pres/err predicted correctly:  -185.3615
## % of predictable range [ (model-null)/(1-null) ]:  0.2207091
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.1208       0.3827
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 741.2     AIC: 778.2
## log likelihood:  -370.6031
## Nagelkerke R2:  0.0741326
## % pres/err predicted correctly:  -228.3142
## % of predictable range [ (model-null)/(1-null) ]:  0.04109785
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	636.4244	0.00000	1e+00	0.9999997	0.2976206	1.762911	-0.9838772	0.3736570
preserved ~ CumErr	666.5611	30.13669	3e-07	0.0000003	0.2501965	2.384593	-0.9930642	NA
preserved ~ CumPres	778.2287	141.80430	0e+00	0.0000000	0.0741326	1.120767	NA	0.3826592

```
########
# 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
##      1.3893      -1.3575       0.3737
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 607.9     AIC: 636.4
## log likelihood:  -303.9395
## Nagelkerke R2:  0.2976206
## % pres/err predicted correctly:  -178.6663
## % of predictable range [ (model-null)/(1-null) ]:  0.2487057
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3846      -0.9931
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 637.8     AIC: 666.6
## log likelihood:  -318.8842
## Nagelkerke R2:  0.2501965
## % pres/err predicted correctly:  -185.3615
## % of predictable range [ (model-null)/(1-null) ]:  0.2207091
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##     2.03997     -0.08165
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 778.5     AIC: 817.7
## log likelihood:  -389.2286
## Nagelkerke R2:  0.005908924
## % pres/err predicted correctly:  -237.1464
## % of predictable range [ (model-null)/(1-null) ]:  0.004164951
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	636.4244	0.00000	1e+00	0.9999997	0.2976206	1.389254	-1.3575342	0.3736570
preserved ~ CumErr	666.5611	30.13669	3e-07	0.0000003	0.2501965	2.384593	-0.9930642	NA
preserved ~ pos	817.7026	181.27817	0e+00	0.0000000	0.0059089	2.039971	NA	-0.0816527

```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 + CumPres	636.4244	0.000000	1.000000	0.9999997	0.2976201	1.6762911	-0.9838772	NA	NA	NA	0.3736570
preserved ~ CumErr + pos	636.4244	0.000000	1.000000	0.9999997	0.2976201	1.389254	-1.3575342	NA	0.3736570	NA	NA
preserved ~ CumErr + I(pos^2) + pos	636.7210	0.000000	1.000000	0.9999997	0.3005291	1.0016449	-1.3477350	-0.0364164	0.6451218	NA	NA
preserved ~ CumErr + stimlen	665.1057	0.000000	1.000000	0.0674296	0.1254263	1.059585	-0.9479663	NA	NA	-0.0993218	NA
preserved ~ CumErr	666.5611	29.840023	0.0000003	0.0000003	0.2501965	2.384593	-0.9930642	NA	NA	NA	NA
preserved ~ CumErr	666.5611	1.455361	0.483028	0.3257039	0.2501965	2.384593	-0.9930642	NA	NA	NA	NA
preserved ~ CumErr	666.5611	30.136693	0.0000003	0.0000003	0.2501965	2.384593	-0.9930642	NA	NA	NA	NA
preserved ~ CumErr	666.5611	30.136693	0.0000003	0.0000003	0.2501965	2.384593	-0.9930642	NA	NA	NA	NA
preserved ~ CumPres	778.2287	141.804299	0.0000000	0.0000000	0.0074132	1.6120767	NA	NA	NA	NA	0.3826592
preserved ~ stimlen	787.8758	122.770072	0.0000000	0.0000000	0.0054002	2.8864286	NA	NA	NA	-0.2937663	NA
preserved ~ I(pos^2) + pos	809.3023	172.581265	0.0000000	0.0000000	0.0026142	1.8216541	NA	-0.0628761	0.4492852	NA	NA
preserved ~ pos	817.7026	181.278170	0.0000000	0.0000000	0.0059089	2.039971	NA	NA	-0.0816527	NA	NA

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres      stimlen     log_freq
##      2.6122      -0.8408       0.4235      -0.1329       0.3022
##
## Degrees of Freedom: 954 Total (i.e. Null);  950 Residual
## Null Deviance:       781.6
## Residual Deviance: 580.6     AIC: 609.7
## log likelihood:  -290.3222
## Nagelkerke R2:  0.3395587
## % pres/err predicted correctly:  -170.165
## % of predictable range [ (model-null)/(1-null) ]:  0.2842548
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq
##      1.7703      -0.8843       0.3901       0.3501
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 584.4     AIC: 612.7
## log likelihood:  -292.1817
## Nagelkerke R2:  0.3339021
## % pres/err predicted correctly:  -171.5992
## % of predictable range [ (model-null)/(1-null) ]:  0.2782575
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres      stimlen
##      3.1859      -0.8862       0.4354      -0.2251
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 595.9     AIC: 623.8
## log likelihood:  -297.9251
## Nagelkerke R2:  0.3162911
## % pres/err predicted correctly:  -174.5776
## % of predictable range [ (model-null)/(1-null) ]:  0.2658033
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      1.7629      -0.9839       0.3737
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 607.9     AIC: 636.4
## log likelihood:  -303.9395
## Nagelkerke R2:  0.2976206
## % pres/err predicted correctly:  -178.6663
## % of predictable range [ (model-null)/(1-null) ]:  0.2487057
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##       1.749
##
## Degrees of Freedom: 954 Total (i.e. Null);  954 Residual
## Null Deviance:       781.6
## Residual Deviance: 781.6     AIC: 819
## log likelihood:  -390.8081
## Nagelkerke R2:  1.986502e-16
## % pres/err predicted correctly:  -238.1424
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres	log_freq	stimlen
preserved ~ CumErr + CumPres + stimlen + log_freq	609.7473	0.000000	1.0000000	0.8133024	0.4339558	2.612172	-0.8407878	0.4235017	0.3021543	-0.1328929
preserved ~ CumErr + CumPres + log_freq	612.6984	2.951059	0.2286577	0.1859678	0.3339021	1.770330	-0.8843474	0.3901493	0.3501070	NA
preserved ~ CumErr + CumPres + stimlen	623.7830	14.035686	0.0008958	0.0007285	0.3162913	1.185944	-0.8861564	0.4354480	NA	-0.2250711
preserved ~ CumErr + CumPres	636.4244	26.677090	0.0000016	0.0000013	0.2976201	1.676291	-0.9838772	0.3736570	NA	NA
preserved ~ 1	819.0194	209.272109	0.0000000	0.0000000	0.0000000	1.749046	NA	NA	NA	NA

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

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

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

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

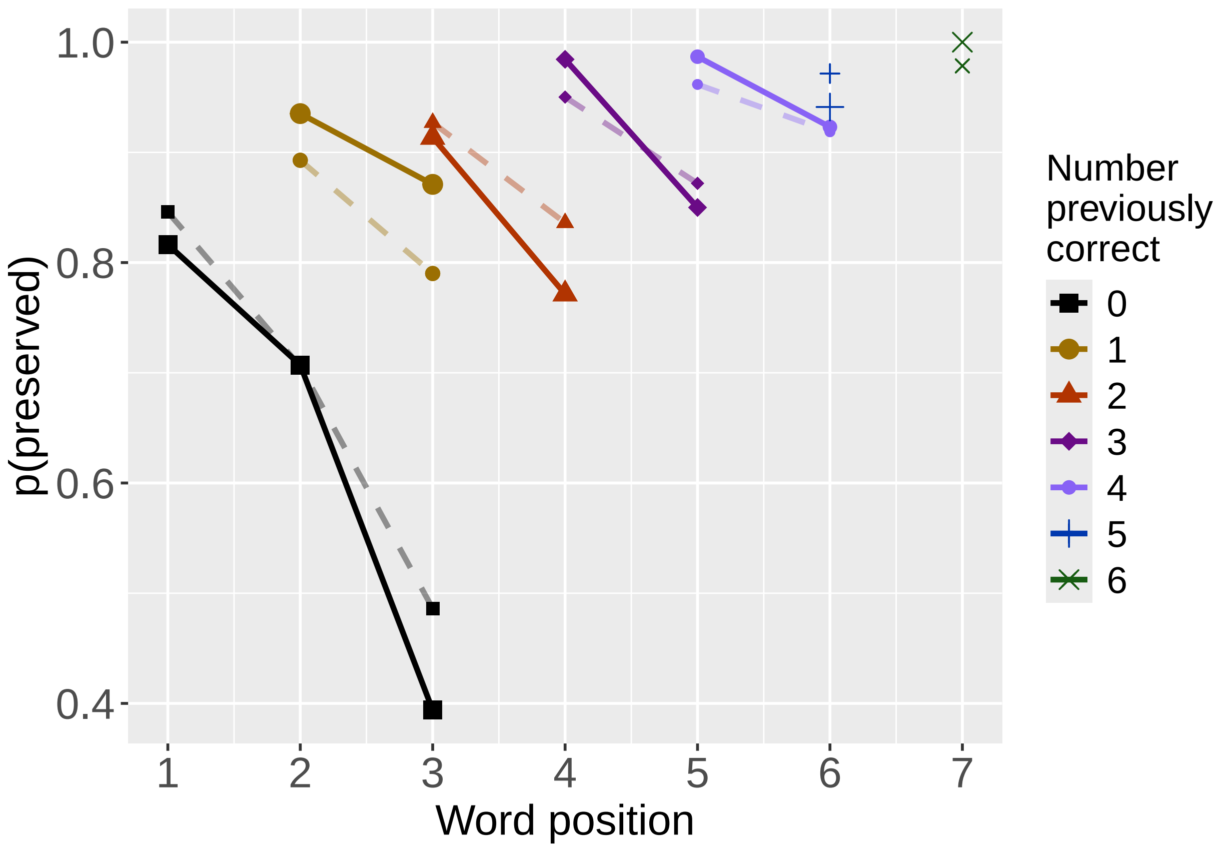
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + CumPres + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   661.21 688.31
## CumPres   1   617.07 644.17
## log_freq  1   595.85 622.95
## stimlen   1   584.36 611.47
## <none>        580.64 609.75
```

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

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

```r
# 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.
```

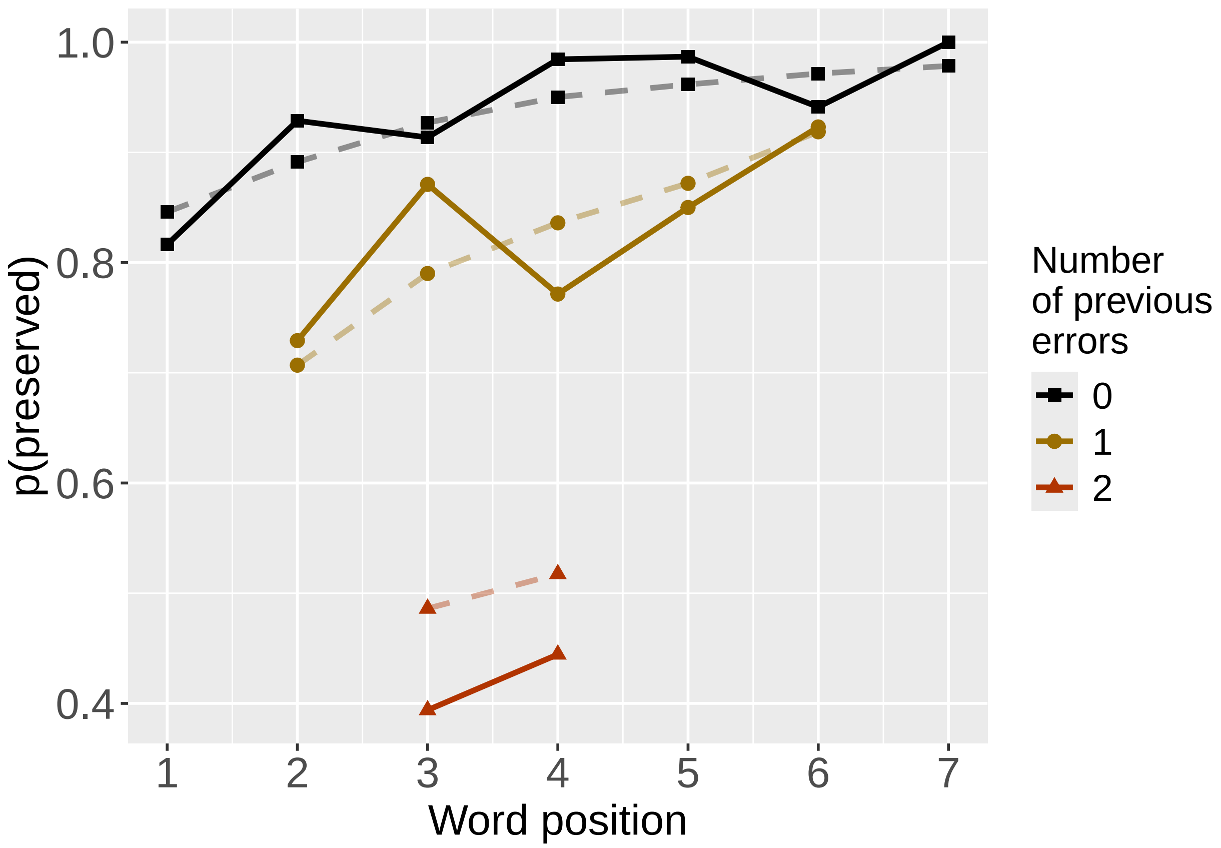
```r
print(PrevCorPlot)
```



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

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

```
print(PrevErrPlot)
```



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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      2.3846      -0.9931
##
## Degrees of Freedom: 954 Total (i.e. Null);  953 Residual
## Null Deviance:       781.6
## Residual Deviance: 637.8     AIC: 666.6
## log likelihood:  -318.8842
## Nagelkerke R2:  0.2501965
## % pres/err predicted correctly:  -185.3615
## % of predictable range [ (model-null)/(1-null) ]:  0.2207091
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      1.7629      -0.9839       0.3737
##
## Degrees of Freedom: 954 Total (i.e. Null);  952 Residual
## Null Deviance:       781.6
## Residual Deviance: 607.9     AIC: 636.4
## log likelihood:  -303.9395
## Nagelkerke R2:  0.2976206
## % pres/err predicted correctly:  -178.6663
## % of predictable range [ (model-null)/(1-null) ]:  0.2487057
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq
##      1.7703      -0.8843       0.3901       0.3501
##
## Degrees of Freedom: 954 Total (i.e. Null);  951 Residual
## Null Deviance:       781.6
## Residual Deviance: 584.4     AIC: 612.7
## log likelihood:  -292.1817
## Nagelkerke R2:  0.3339021
## % pres/err predicted correctly:  -171.5992
## % of predictable range [ (model-null)/(1-null) ]:  0.2782575
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres     log_freq      stimlen
##      2.6122      -0.8408       0.4235       0.3022      -0.1329
##
## Degrees of Freedom: 954 Total (i.e. Null);  950 Residual
## Null Deviance:       781.6
## Residual Deviance: 580.6     AIC: 609.7
## log likelihood:  -290.3222
## Nagelkerke R2:  0.3395587
## % pres/err predicted correctly:  -170.165
## % of predictable range [ (model-null)/(1-null) ]:  0.2842548
## ***************************
```

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

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

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

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

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

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

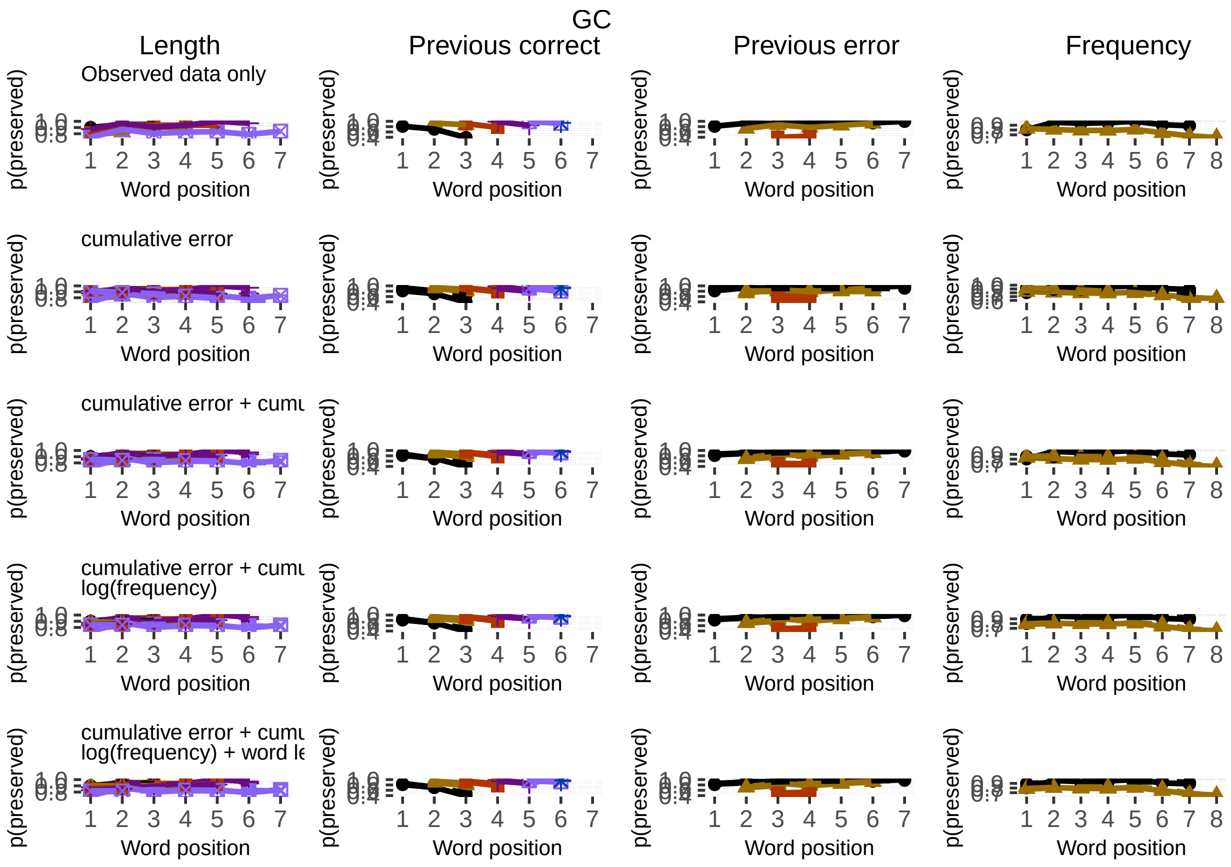
```
## Warning: Removed 1 row containing missing values or values outside the scale range (`geom_point()`).
## Removed 1 row containing missing values or values outside the scale range (`geom_point()`).
```

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

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

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 7 values. Consider specifying shapes manually if you need that many have
##   them.
## Warning: Removed 1 row containing missing values or values outside the scale range (`geom_point()`).
## Removed 1 row containing missing values or values outside the scale range (`geom_point()`).
```

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



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

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

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

	CumErr	CumPres	stimlen	log_freq
McFadden	0.1452410	0.0546624	0.0227271	0.0433021
SquaredCorrelation	0.1099220	0.0416096	0.0180976	0.0338564
Nagelkerke	0.1911897	0.0723725	0.0314775	0.0588872
Estrella	0.1271590	0.0478349	0.0197123	0.0376943

```
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 + CumPres + log_freq + stimlen CumErr + CumPres + log_freq + stimlen 580.6443
## CumErr + CumPres + log_freq                     CumErr + CumPres + log_freq 584.3634
## CumErr + CumPres                                           CumErr + CumPres 607.8789
## CumErr                                                               CumErr 637.7684
## null                                                                   null 781.6163
##                                       deviance_explained percent_explained
## CumErr + CumPres + log_freq + stimlen           200.9719          25.71235
## CumErr + CumPres + log_freq                     197.2529          25.23653
## CumErr + CumPres                                173.7374          22.22796
## CumErr                                          143.8479          18.40390
## null                                              0.0000           0.00000
##                                       percent_of_explained_deviance increment_in_explained
## CumErr + CumPres + log_freq + stimlen                     100.00000               1.850541
## CumErr + CumPres + log_freq                                98.14946              11.700886
## CumErr + CumPres                                           86.44857              14.872459
## CumErr                                                     71.57611              71.576114
## null                                                             NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + CumPres + log_freq + stimlen	580.6443	200.9719
CumErr + CumPres + log_freq	584.3634	197.2529
CumErr + CumPres	607.8789	173.7374
CumErr	637.7684	143.8479
null	781.6163	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + CumPres + log_freq + stimlen	25.71235	100.00000	1.850541
CumErr + CumPres + log_freq	25.23653	98.14946	11.700886
CumErr + CumPres	22.22796	86.44857	14.872459
CumErr	18.40390	71.57611	71.576114
null	0.00000	NA	0.000000

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

```
##          Nagelkerke
## CumErr   0.54019553
## CumPres  0.20448423
## stimlen  0.08893781
## log_freq 0.16638243
```

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

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

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

## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumErr'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'CumPres'. You can override using the `.groups` argument.
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.3001074	637.7684
preserved ~ CumErr+CumPres	0.7183148	607.8789
preserved ~ CumErr+CumPres+log_freq	0.7371128	584.3634
preserved ~ CumErr+CumPres+log_freq+stimlen	0.7558880	580.6443

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

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

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

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

```
##                                        model p_accounted_for model_deviance diff_CumErr
## 1                         preserved ~ CumErr       0.3001074       637.7684   0.0000000
## 2                 preserved ~ CumErr+CumPres       0.7183148       607.8789   0.4182075
## 3        preserved ~ CumErr+CumPres+log_freq       0.7371128       584.3634   0.4370054
## 4 preserved ~ CumErr+CumPres+log_freq+stimlen       0.7558880       580.6443   0.4557806
##   diff_CumErr+CumPres diff_CumErr+CumPres+log_freq diff_CumErr+CumPres+log_freq+stimlen
## 1         -0.41820745                  -0.43700540                          -0.45578058
## 2          0.00000000                  -0.01879795                          -0.03757313
## 3          0.01879795                   0.00000000                          -0.01877518
## 4          0.03757313                   0.01877518                           0.00000000
```

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

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

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

model	diff_CumErr	diff_CumErr+CumPres	diff_CumErr+CumPres+log_freq
preserved ~ CumErr	0.0000000	-0.4182075	-0.4370054
preserved ~ CumErr+CumPres	0.4182075	0.0000000	-0.0187979
preserved ~ CumErr+CumPres+log_freq	0.4370054	0.0187979	0.0000000
preserved ~ CumErr+CumPres+log_freq+stimlen	0.4557806	0.0375731	0.0187752