TC - naming - Serial position analysis (v5)

```
# do this in calling R script
# library(ggplot2)
# library(fmsb) # for TestModels
# library(lme4)
# library(kableExtra)
# library(MASS) # for dropterm
# library(tidyverse)
# library(dominanceanalysis)
# library(cowplot) # for multiple plots in one figure
# library(formatters) # to wrap strings for plot titles
# library(pals) # for continous color palettes
```

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

```
# for testing - set by calling script normally
# ppt_parms <- read.csv(config$patient_param_file)
# CurPat <- ppt_parms$patient
# CurTask <- ppt_parms$task
# MinLength <- ppt_parms$min_length
# MaxLength <- ppt_parms$max_length
# RemoveFinalPosition<-ppt_parms$remove_final_position
```

```
# create patient directory if it doesn't already exist
if(!dir.exists(paste0(RootDir,"/output/",CurPat))){
  dir.create(paste0(RootDir,"/output/",CurPat),showWarnings = TRUE)
}
TablesDir <- paste0(RootDir,"/output/",CurPat,"/tables/")
if(!dir.exists(TablesDir)){
  dir.create(TablesDir,showWarnings = TRUE)
}
FigDir <- paste0(RootDir,"/output/",CurPat,"/fig/")
if(!dir.exists(FigDir)){
  dir.create(FigDir,showWarnings = TRUE)
}
ReportsDir <- paste0(RootDir,"/output/",CurPat,"/reports/")
if(!dir.exists(ReportsDir)){
  dir.create(ReportsDir,showWarnings = TRUE)
}
results_report_DF <- data.frame(Description=character(), ParameterValue=double())
```

```
ModelDatFilename<-paste0(RootDir,"/data/",CurPat,"/",CurPat,"_",CurTask,"_posdat.csv")
if(!file.exists(ModelDatFilename)){
  print("**ERROR** Input data file not found")
  print(sprintf("\tfile: ",ModelDatFilename))
  print(sprintf("\troot dir: ",RootDir))
```

```
}
PosDat<-read.csv(ModelDatFilename)
```

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	220	12	27	NA	NA	259
2	26	NA	189	12	32	259
3	100	NA	56	99	4	259
4	138	NA	64	16	12	230
5	59	NA	77	20	9	165
6	63	NA	19	31	8	121
7	45	NA	16	3	3	67
8	14	NA	6	4	1	25
9	9	NA	NA	NA	NA	9

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8494208	0.046332	0.1042471	NA	NA	259
2	0.1003861	NA	0.7297297	0.0463320	0.1235521	259
3	0.3861004	NA	0.2162162	0.3822394	0.0154440	259
4	0.6000000	NA	0.2782609	0.0695652	0.0521739	230
5	0.3575758	NA	0.4666667	0.1212121	0.0545455	165
6	0.5206612	NA	0.1570248	0.2561983	0.0661157	121

pos_factor	O	P	V	1	S	total
7	0.6716418	NA	0.2388060	0.0447761	0.0447761	67
8	0.5600000	NA	0.2400000	0.1600000	0.0400000	25
9	1.0000000	NA	NA	NA	NA	9

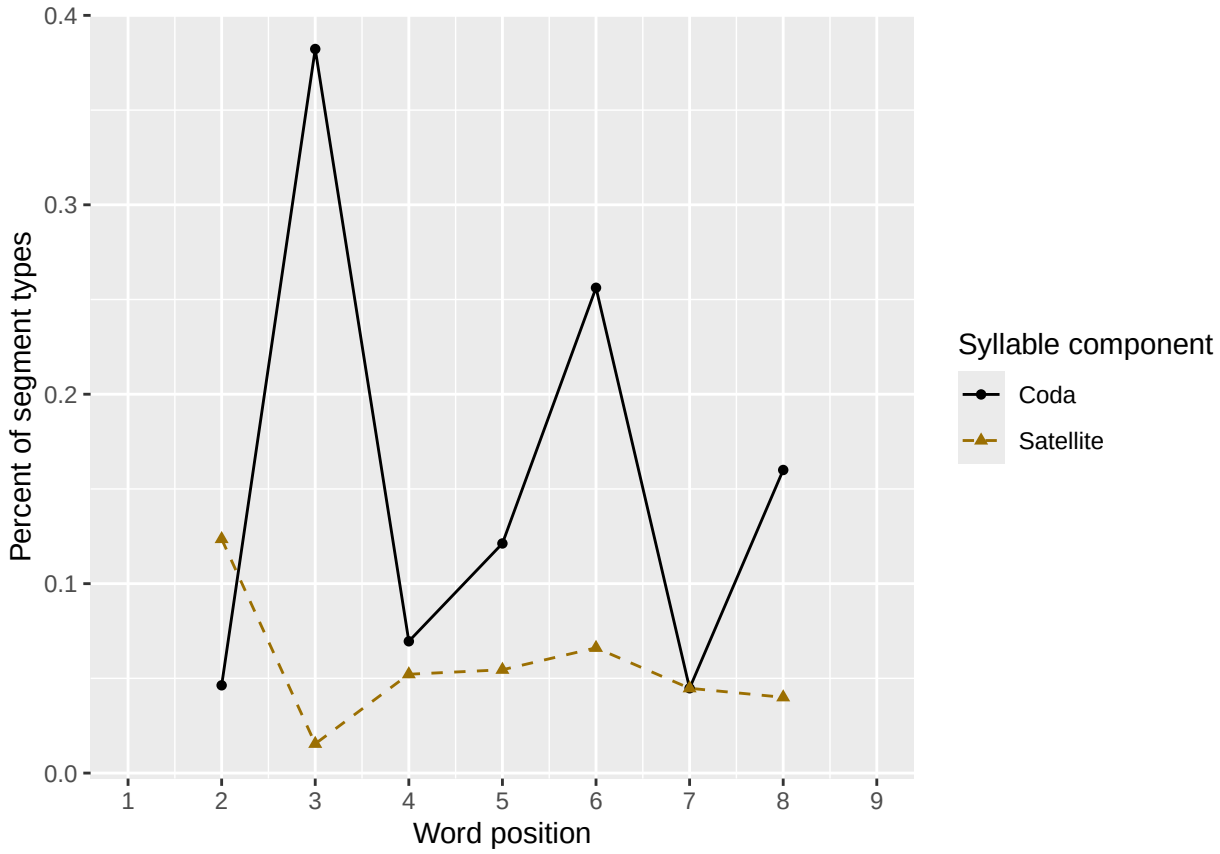
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1       4 0.966 0.897 0.966 NA     NA     NA     NA     NA       NA
## 2       5 0.908 0.938 0.923  0.892 NA     NA     NA     NA       NA
## 3       6 0.807 0.761 0.784  0.693  0.602 NA     NA     NA       NA
## 4       7 0.889 0.870 0.802  0.765  0.704  0.728 NA     NA       NA
## 5       8 0.877 0.845 0.786  0.817  0.806  0.75   0.762 NA       NA
## 6       9 0.984 0.875 0.766  0.766  0.719  0.797  0.75   0.688  NA
## 7      10 0.937 0.849 0.587  0.587  0.603  0.5    0.603  0.5     0.5
```

```
# 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.
```

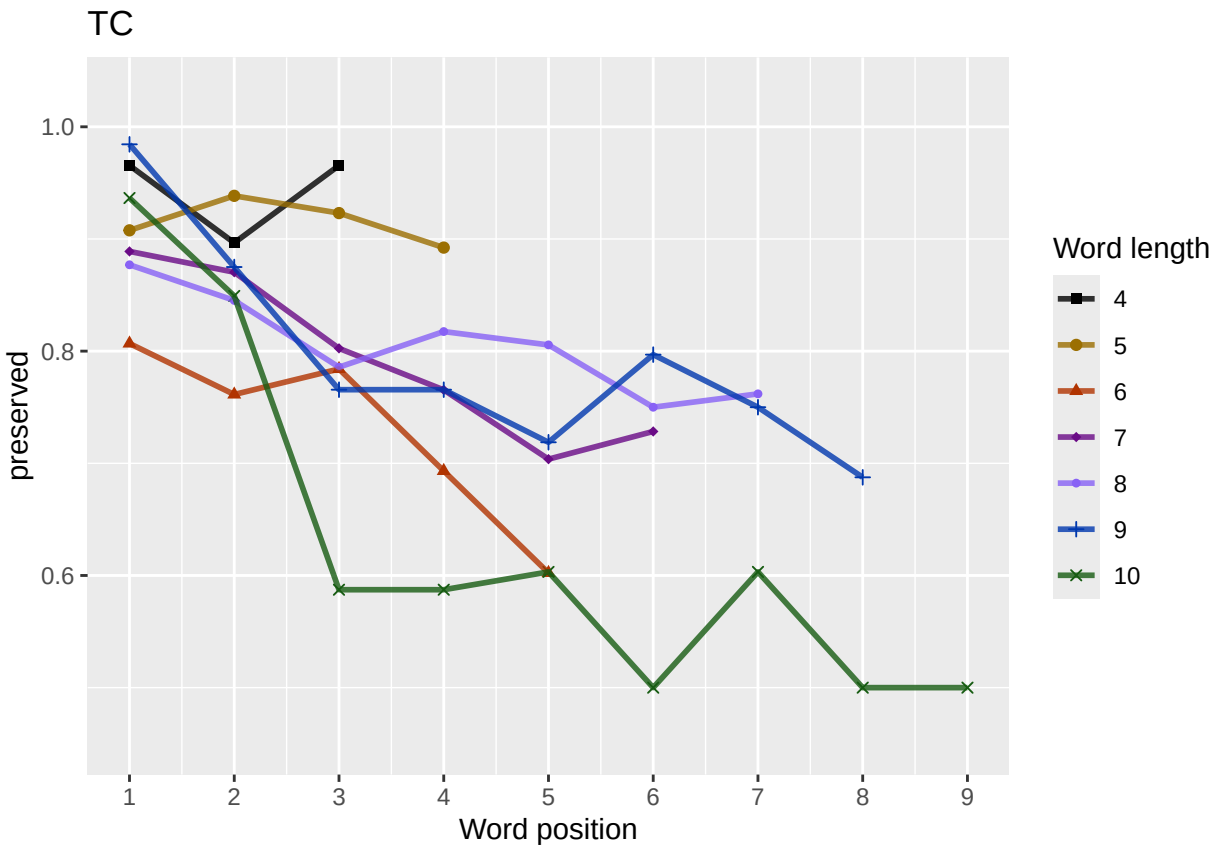
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    29    29    29    NA    NA    NA    NA    NA    NA
## 2       5    65    65    65    65    NA    NA    NA    NA    NA
## 3       6    44    44    44    44    44    NA    NA    NA    NA
## 4       7    54    54    54    54    54    54    NA    NA    NA
## 5       8    42    42    42    42    42    42    42    NA    NA
## 6       9    16    16    16    16    16    16    16    16    NA
## 7      10     9     9     9     9     9     9     9     9     9
```

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

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

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



Length and position

```
# length and position

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

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

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

```
## **************************
## model index:  7
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##      3.4650      -0.1423       0.0261      -0.4005
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1245  AIC: 1367
## log likelihood:  -622.544
## Nagelkerke R2:  0.06143366
## % pres/err predicted correctly:  -404.4203
## % of predictable range [ (model-null)/(1-null) ]:  0.04222235
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.9989      -0.1247      -0.1859
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1248  AIC: 1369
## log likelihood:  -623.7784
## Nagelkerke R2:  0.05861699
## % pres/err predicted correctly:  -404.9615
## % of predictable range [ (model-null)/(1-null) ]:  0.04094385
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.52112     -0.19317     -0.34440      0.02002
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1247  AIC: 1370
## log likelihood:  -623.4173
## Nagelkerke R2:  0.0594415
## % pres/err predicted correctly:  -405.0324
## % of predictable range [ (model-null)/(1-null) ]:  0.04077624
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        3.3262518        -0.1179943         0.0300570        -0.3706408        -0.0000415
##      stimlen:pos
##       -0.0075552
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
## Residual Deviance: 1245  AIC: 1371
## log likelihood:  -622.5133
## Nagelkerke R2:  0.06150363
## % pres/err predicted correctly:  -404.3236
## % of predictable range [ (model-null)/(1-null) ]:  0.04245082
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2957      -0.2325
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1254  AIC: 1378
## log likelihood:  -626.9605
## Nagelkerke R2:  0.05133276
## % pres/err predicted correctly:  -407.7824
## % of predictable range [ (model-null)/(1-null) ]:  0.03427957
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.51751      0.01579     -0.36625
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1253  AIC: 1379
## log likelihood:  -626.486
## Nagelkerke R2:  0.05242125
## % pres/err predicted correctly:  -407.7274
## % of predictable range [ (model-null)/(1-null) ]:  0.03440942
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      3.0110      -0.2236
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1271  AIC: 1387
## log likelihood:  -635.5173
## Nagelkerke R2:  0.03157967
## % pres/err predicted correctly:  -412.2753
## % of predictable range [ (model-null)/(1-null) ]:  0.02366538
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.44
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1393 Residual
## Null Deviance:       1298
## Residual Deviance: 1298  AIC: 1421
## log likelihood:  -648.9827
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -422.2927
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	1367.479	0.000000	1.000000	0.5325155	0.0614337	3.7465048	-0.1423146	-0.4005349	NA	0.0260954	NA
preserved ~ stimlen + pos	1369.013	1.533279	0.4645707	0.2473907	0.0586172	3.0998930	-0.1247496	-0.1859006	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	1370.172	2.699614	0.259290	0.138076	0.105944	3.5521122	-0.1931728	-3.3444016	0.0200172	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	1371.325	3.845598	0.146197	0.077852	0.106150	3.6326252	-0.1179943	-3.3706408	0.0075552	0.0300570	-4.15e-05
preserved ~ pos	1378.213	10.733185	0.004670	0.002486	0.095133	2.8295727	NA	-0.2324707	NA	NA	NA
preserved ~ I(pos^2) + pos	1379.029	11.549323	0.003105	0.001653	0.095242	2.3517509	NA	-0.3662451	NA	0.0157920	NA
preserved ~ stimlen	1387.492	20.014810	0.000045	0.000024	0.003157	3.7011001	-0.2236432	NA	NA	NA	NA
preserved ~ 1	1421.377	53.899593	0.000000	0.000000	0.000000	1.0440066	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##      3.4650      -0.1423       0.0261      -0.4005
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1245  AIC: 1367
```

```
PosDat$LPFitted<-fitted(BestLPModel)
fitted_len_pos_plot <- plot_len_pos_obs_predicted(PosDat, PosDat$patient[1],
                          NULL,palette_values=palette_values)
```

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

```
# len/pos table
fitted_pos_len_summary <- PosDat %>% group_by(stimlen,pos) %>% summarise(fitted = mean(LPFitted))
```

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

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

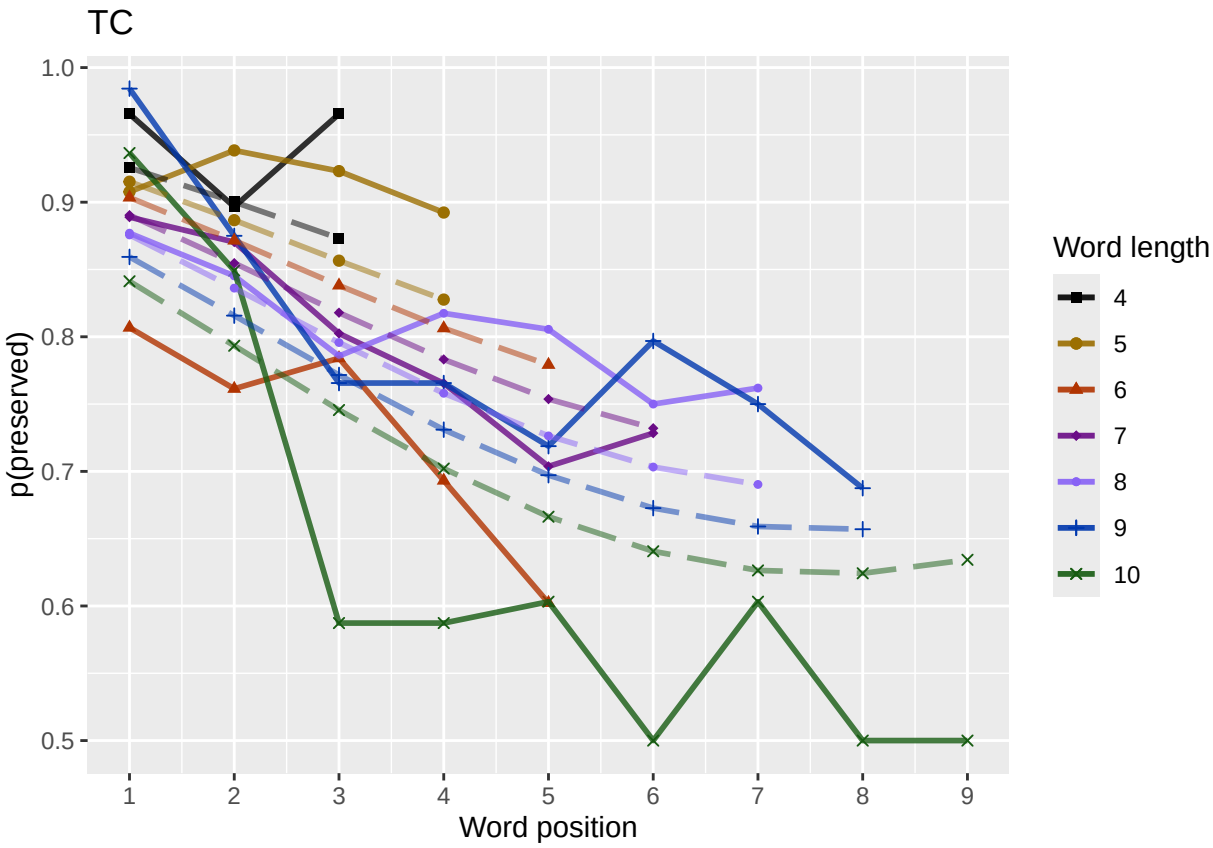
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.926 0.900 0.873 NA     NA     NA     NA     NA     NA
## 2       5 0.915 0.887 0.857  0.828 NA     NA     NA     NA     NA
## 3       6 0.903 0.872 0.838  0.806  0.779 NA     NA     NA     NA
## 4       7 0.890 0.855 0.818  0.783  0.754  0.732 NA     NA     NA
## 5       8 0.876 0.836 0.796  0.758  0.726  0.703  0.690 NA     NA
```

```
## 6       9 0.859 0.816 0.772  0.731  0.697  0.673  0.659  0.657 NA
## 7      10 0.841 0.793 0.746  0.702  0.666  0.641  0.626  0.624  0.634
```

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

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

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

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



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

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

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

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

```
## # A tibble: 1 x 2
##   frag_sum     N
##      <int> <int>
## 1       34   259
```

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

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

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

```
## [1] "The number of responses with fragments was 34 / 259 = 13.13 percent"
```

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

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

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

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

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

```
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.09491     -0.08855      0.04973     -0.37799
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1276 Residual
## Null Deviance:       923.2
## Residual Deviance: 917.4     AIC: 1018
## log likelihood:  -458.685
## Nagelkerke R2:  0.008878885
## % pres/err predicted correctly:  -274.2048
## % of predictable range [ (model-null)/(1-null) ]:  0.005854815
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##     2.39282     -0.06592
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1278 Residual
## Null Deviance:       923.2
## Residual Deviance: 921.7     AIC: 1019
## log likelihood:  -460.8534
## Nagelkerke R2:  0.002304555
## % pres/err predicted correctly:  -275.2258
## % of predictable range [ (model-null)/(1-null) ]:  0.002166419
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.65540     -0.23314     -0.44507      0.05743
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1276 Residual
## Null Deviance:       923.2
## Residual Deviance: 918.5     AIC: 1021
## log likelihood:  -459.2562
## Nagelkerke R2:  0.007149291
## % pres/err predicted correctly:  -274.6197
## % of predictable range [ (model-null)/(1-null) ]:  0.004355845
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
```

```
##       1.944
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1279 Residual
## Null Deviance:       923.2
## Residual Deviance: 923.2     AIC: 1021
## log likelihood:  -461.6117
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -275.8255
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##    2.394774    -0.063127    -0.006367
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1277 Residual
## Null Deviance:       923.2
## Residual Deviance: 921.7     AIC: 1021
## log likelihood:  -460.8449
## Nagelkerke R2:  0.002330403
## % pres/err predicted correctly:  -275.2369
## % of predictable range [ (model-null)/(1-null) ]:  0.0021264
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.49981      0.04222     -0.34856
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1277 Residual
## Null Deviance:       923.2
## Residual Deviance: 919.6     AIC: 1021
## log likelihood:  -459.8191
## Nagelkerke R2:  0.005443207
## % pres/err predicted correctly:  -275.0744
## % of predictable range [ (model-null)/(1-null) ]:  0.002713345
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##          2.56045          -0.03650          -0.07775           0.25166           0.01456
##      stimlen:pos
##         -0.06907
```

```
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1274 Residual
## Null Deviance:       923.2
## Residual Deviance: 916.2     AIC: 1022
## log likelihood:  -458.1101
## Nagelkerke R2:  0.01061836
## % pres/err predicted correctly:  -274.0756
## % of predictable range [ (model-null)/(1-null) ]:  0.006321326
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##     2.03171     -0.02678
##
## Degrees of Freedom: 1279 Total (i.e. Null);  1278 Residual
## Null Deviance:       923.2
## Residual Deviance: 922.9     AIC: 1023
## log likelihood:  -461.4408
## Nagelkerke R2:  0.0005197044
## % pres/err predicted correctly:  -275.7553
## % of predictable range [ (model-null)/(1-null) ]:  0.0002536354
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + I(pos^2) + pos	1018.200	0.000000	1.0000000	0.3590121	0.0088789	2.0094913	0.0885549	-0.3779868	NA	0.0497330	NA
preserved ~ stimlen	1019.211	1.013923	0.6023283	0.2162440	0.0023026	2.6392822	0.0659167	NA	NA	NA	NA
preserved ~ stimlen * pos	1020.832	2.633175	0.2680484	0.0962326	0.0071483	2.3655399	0.2331410	-0.4450670	0.0574337	NA	NA
preserved ~ 1	1020.951	1.751698	8.2526250	0.0906955	0.0000000	1.0943531	NA	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	1021.263	0.069474	1.2155123	0.3077371	0.0023302	4394774	-0.0631266	-0.0063675	NA	NA	NA
preserved ~ I(pos^2) + pos	1021.343	0.147463	0.2072703	0.0744126	0.0054432	2499805	NA	-0.3485641	NA	0.0422180	NA
preserved ~ stimlen * (I(pos^2) + pos)	1022.370	1.170183	0.1242958	0.0446237	0.0106184	4560454	-0.0365027	0.2516605	-0.0690698	-0.0777499	0.0145605
preserved ~ pos	1022.514	1.319631	0.1153464	0.0414108	0.0005127	7031709	NA	-0.0267761	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.
```

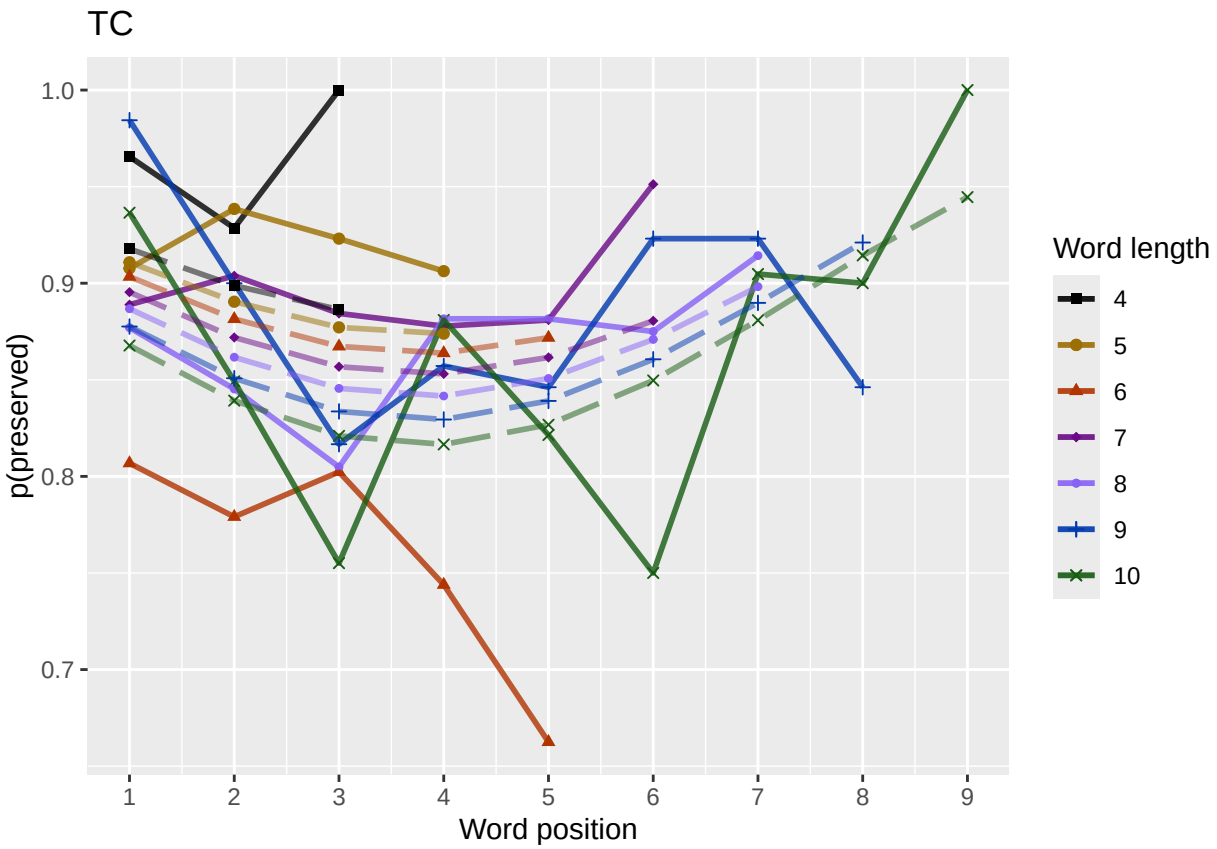
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.918 0.899 0.886 NA     NA     NA     NA     NA     NA
## 2       5 0.911 0.890 0.877  0.874 NA     NA     NA     NA     NA
## 3       6 0.903 0.881 0.867  0.864  0.872 NA     NA     NA     NA
## 4       7 0.895 0.872 0.857  0.853  0.862  0.881 NA     NA     NA
## 5       8 0.887 0.862 0.846  0.842  0.851  0.871  0.898 NA     NA
## 6       9 0.878 0.851 0.834  0.829  0.839  0.861  0.890  0.921 NA
## 7      10 0.868 0.839 0.821  0.817  0.827  0.850  0.881  0.914  0.945
```

```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: 0.45 - 1.03"
```

```
# 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] -0.02103576
```

```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] -0.02577169
```

```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] "Average upward change after U minimum"
## [1] 0.01005083
```

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

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

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

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

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

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

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

```
## [1] "differences from left max to min for each row: "
## [1] 0.05248156 0.08759338 0.12432303 0.15830407 0.18538775 0.20231510 0.21696373
## [1] "differences from min to right max for each row: "
## [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.01005083
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.2169637
## [1] 0.01005083
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.04632491
```

```
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",
```

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

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

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

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

```
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          2.48453           0.04908          -0.19636           0.03151          -0.46412
## stimlen:log_freq
##          0.09138
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
## Residual Deviance: 1151  AIC: 1269
```

```
## log likelihood:  -575.6639
## Nagelkerke R2:  0.1647977
## % pres/err predicted correctly:  -370.1816
## % of predictable range [ (model-null)/(1-null) ]:  0.123109
## **************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)               pos
##           2.71905            0.02585             0.04007            0.04797           -0.56091
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##           0.06835            0.01517           -0.09069
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1386 Residual
## Null Deviance:       1298
## Residual Deviance: 1149  AIC: 1271
## log likelihood:  -574.5203
## Nagelkerke R2:  0.1672334
## % pres/err predicted correctly:  -369.159
## % of predictable range [ (model-null)/(1-null) ]:  0.1255246
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          1.94449           0.06759          -0.16390          -0.20690           0.08687
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1389 Residual
## Null Deviance:       1298
## Residual Deviance: 1155  AIC: 1271
## log likelihood:  -577.2653
## Nagelkerke R2:  0.1613805
## % pres/err predicted correctly:  -371.1765
## % of predictable range [ (model-null)/(1-null) ]:  0.1207585
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          1.95262           0.05772          -0.16540          -0.19150           0.07737
##     log_freq:pos
##          0.01817
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
```

```
## Residual Deviance: 1154  AIC: 1273
## log likelihood:  -577.0529
## Nagelkerke R2:  0.1618342
## % pres/err predicted correctly:  -370.7254
## % of predictable range [ (model-null)/(1-null) ]:  0.1218241
## ***************************
## model index:  9
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)           I(pos^2)                pos           log_freq  I(pos^2):log_freq
##           2.82695            0.05137           -0.56896            0.51607            0.02113
##      pos:log_freq
##          -0.11650
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
## Residual Deviance: 1153  AIC: 1273
## log likelihood:  -576.499
## Nagelkerke R2:  0.1630166
## % pres/err predicted correctly:  -370.8521
## % of predictable range [ (model-null)/(1-null) ]:  0.1215249
## ****************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##           2.93787           -0.01672            0.05266           -0.57354            0.50988
## I(pos^2):log_freq       pos:log_freq
##           0.02123           -0.11645
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1387 Residual
## Null Deviance:       1298
## Residual Deviance: 1153  AIC: 1275
## log likelihood:  -576.4519
## Nagelkerke R2:  0.1631173
## % pres/err predicted correctly:  -370.6639
## % of predictable range [ (model-null)/(1-null) ]:  0.1219694
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.19221      -0.16190       0.27164       0.04556
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
```

```
## Null Deviance:       1298
## Residual Deviance: 1159  AIC: 1276
## log likelihood:  -579.6622
## Nagelkerke R2:  0.1562508
## % pres/err predicted correctly:  -372.5019
## % of predictable range [ (model-null)/(1-null) ]:  0.1176273
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      2.3036      -0.1946       0.4408
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1163  AIC: 1278
## log likelihood:  -581.3261
## Nagelkerke R2:  0.1526794
## % pres/err predicted correctly:  -374.6905
## % of predictable range [ (model-null)/(1-null) ]:  0.1124569
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##     2.146240      0.008158     -0.164807      0.274814      0.045287
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1389 Residual
## Null Deviance:       1298
## Residual Deviance: 1159  AIC: 1278
## log likelihood:  -579.6504
## Nagelkerke R2:  0.156276
## % pres/err predicted correctly:  -372.5696
## % of predictable range [ (model-null)/(1-null) ]:  0.1174673
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##    2.696066    -0.002579     0.027098    -0.422184     0.445259
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1389 Residual
## Null Deviance:       1298
## Residual Deviance: 1160  AIC: 1279
## log likelihood:  -580.0402
```

```
## Nagelkerke R2:  0.1554401
## % pres/err predicted correctly:  -374.1164
## % of predictable range [ (model-null)/(1-null) ]:  0.1138132
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.21445      0.01566     -0.19990      0.44495
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1163  AIC: 1280
## log likelihood:  -581.2825
## Nagelkerke R2:  0.152773
## % pres/err predicted correctly:  -374.7935
## % of predictable range [ (model-null)/(1-null) ]:  0.1122137
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          1.97986          -0.04699          -0.13392           0.08094
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1181  AIC: 1293
## log likelihood:  -590.3006
## Nagelkerke R2:  0.1332693
## % pres/err predicted correctly:  -379.9381
## % of predictable range [ (model-null)/(1-null) ]:  0.1000599
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##      2.2376      -0.0927       0.4351
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1188  AIC: 1300
## log likelihood:  -593.8912
## Nagelkerke R2:  0.1254332
## % pres/err predicted correctly:  -383.095
## % of predictable range [ (model-null)/(1-null) ]:  0.09260183
```

```
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##      3.4650      -0.1423       0.0261      -0.4005
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1245  AIC: 1367
## log likelihood:  -622.544
## Nagelkerke R2:  0.06143366
## % pres/err predicted correctly:  -404.4203
## % of predictable range [ (model-null)/(1-null) ]:  0.04222235
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##      2.9989      -0.1247      -0.1859
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1248  AIC: 1369
## log likelihood:  -623.7784
## Nagelkerke R2:  0.05861699
## % pres/err predicted correctly:  -404.9615
## % of predictable range [ (model-null)/(1-null) ]:  0.04094385
## ***************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.52112     -0.19317     -0.34440      0.02002
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 1247  AIC: 1370
## log likelihood:  -623.4173
## Nagelkerke R2:  0.0594415
## % pres/err predicted correctly:  -405.0324
## % of predictable range [ (model-null)/(1-null) ]:  0.04077624
## ***************************
## 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)
##        3.3262518        -0.1179943         0.0300570        -0.3706408        -0.0000415
##      stimlen:pos
##       -0.0075552
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
## Residual Deviance: 1245  AIC: 1371
## log likelihood:  -622.5133
## Nagelkerke R2:  0.06150363
## % pres/err predicted correctly:  -404.3236
## % of predictable range [ (model-null)/(1-null) ]:  0.04245082
## **************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2957      -0.2325
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1254  AIC: 1378
## log likelihood:  -626.9605
## Nagelkerke R2:  0.05133276
## % pres/err predicted correctly:  -407.7824
## % of predictable range [ (model-null)/(1-null) ]:  0.03427957
## **************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.51751      0.01579     -0.36625
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1253  AIC: 1379
## log likelihood:  -626.486
## Nagelkerke R2:  0.05242125
## % pres/err predicted correctly:  -407.7274
## % of predictable range [ (model-null)/(1-null) ]:  0.03440942
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.0110      -0.2236
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1271  AIC: 1387
## log likelihood:  -635.5173
## Nagelkerke R2:  0.03157967
## % pres/err predicted correctly:  -412.2753
## % of predictable range [ (model-null)/(1-null) ]:  0.02366538
## ***************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.44
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1393 Residual
## Null Deviance:       1298
## Residual Deviance: 1298  AIC: 1421
## log likelihood:  -648.9827
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -422.2927
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	Delta_AIC	AICexp	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 ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1963633	0.0913834	-0.4641239	NA	NA	0.0315134	NA	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.5609114	NA	-0.0906855	0.0479652	[illegible]	0.015166	NA	NA
preserved ~ stimlen * log_freq + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1639004	0.0868692	-0.2068980	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.1653992	0.0773726	-0.1914959	NA	0.0181660	NA	NA	NA	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.5160657	[illegible]	[illegible]	NA	0.0515070	[illegible]	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0167178	0.5098765	[illegible]	NA	0.0526604	[illegible]	NA	NA	NA	NA
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.2716419	NA	-0.1619038	0.0455605	NA	NA	NA	NA	NA	NA
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	NA	0.4407796	NA	-0.1946057	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.1648065	0.0452875	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0025788	0.4452587	-0.4221843	NA	NA	0.0270980	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.1998975	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0469883	0.0809377	[illegible]	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0927050	0.4350862	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAIC AICexp 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 ~ stimlen + I(pos^2) + pos	1367.948.96103318000000000614365048	-0.1423146	NA	NA	-0.4005349	NA	NA	0.0260954	NA	NA	NA
preserved ~ stimlen + pos	1369.0113.14659000000000582198930	-0.1247496	NA	NA	-0.1859006	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	1370.1719.3102932000000005944515122	-0.1931728	NA	NA	-0.3444016	NA	NA	NA	NA	0.0200172	NA
preserved ~ stimlen * (I(pos^2) + pos)	1371.3225.458910000000006153326252	-0.1179943	NA	NA	-0.3706408	NA	NA	0.0300570	NA	-0.0074552	-4.55e-05
preserved ~ pos	1378.2193.3465030000000051232857	NA	NA	NA	-0.2324707	NA	NA	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1379.1029.1626400000000522517350	NA	NA	NA	-0.3662451	NA	NA	0.0157920	NA	NA	NA
preserved ~ stimlen	1387.4198.6281300000000313579710	-0.2236432	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	1421.3577.510910000000000010044006	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          2.48453           0.04908          -0.19636           0.03151          -0.46412
## stimlen:log_freq
##          0.09138
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
## Residual Deviance: 1151  AIC: 1269
```

```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.
```

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

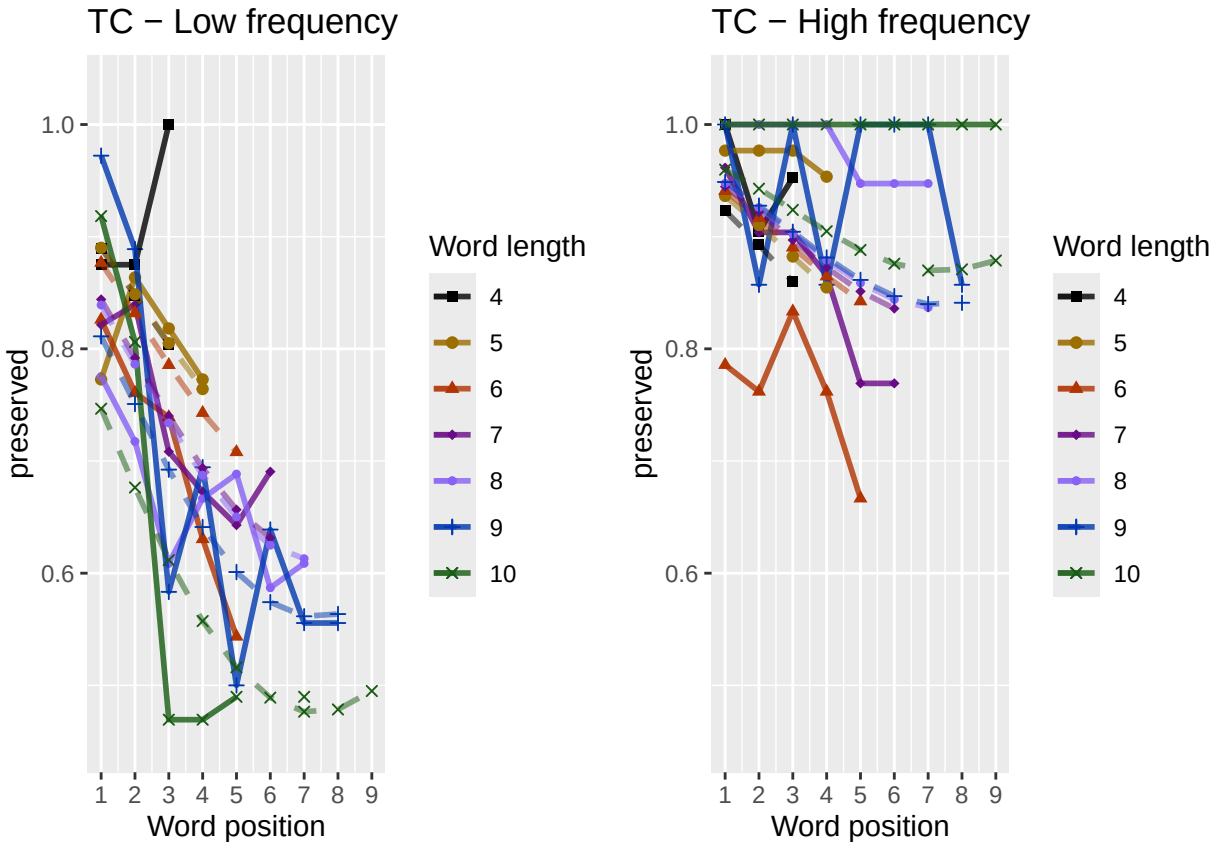
```
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## `summarise()` has grouped output by 'stimlen'. You can override using the `.groups` argument.
## Scale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

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

```
## Warning: Removed 3 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
##       2.436       -1.796
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 795.6     AIC: 855.4
## log likelihood:  -397.7858
## Nagelkerke R2:  0.4994372
## % pres/err predicted correctly:  -233.1762
## % of predictable range [ (model-null)/(1-null) ]:  0.4467748
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.8994       0.3130
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1247  AIC: 1367
## log likelihood:  -623.4325
## Nagelkerke R2:  0.05940678
## % pres/err predicted correctly:  -407.3372
## % of predictable range [ (model-null)/(1-null) ]:  0.03533132
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.2957      -0.2325
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1254  AIC: 1378
## log likelihood:  -626.9605
## Nagelkerke R2:  0.05133276
## % pres/err predicted correctly:  -407.7824
## % of predictable range [ (model-null)/(1-null) ]:  0.03427957
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.51751      0.01579     -0.36625
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1253  AIC: 1379
## log likelihood:  -626.486
## Nagelkerke R2:  0.05242125
## % pres/err predicted correctly:  -407.7274
## % of predictable range [ (model-null)/(1-null) ]:  0.03440942
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.0110      -0.2236
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1271  AIC: 1387
## log likelihood:  -635.5173
## Nagelkerke R2:  0.03157967
## % pres/err predicted correctly:  -412.2753
## % of predictable range [ (model-null)/(1-null) ]:  0.02366538
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.44
```

```
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1393 Residual
## Null Deviance:       1298
## Residual Deviance: 1298  AIC: 1421
## log likelihood:  -648.9827
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -422.2927
## % 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	855.445	0.0000	1	1	0.4994372	2.4364904	NA	-1.795521	NA	NA	NA
preserved ~ CumPres	1367.347	511.9017	0	0	0.0594068	8.8993523	0.3130078	NA	NA	NA	NA
preserved ~ pos	1378.213	522.7676	0	0	0.0513328	8.2957272	NA	NA	NA	-0.2324707	NA
preserved ~ (I(pos^2) + pos)	1379.029	523.5838	0	0	0.0524213	8.5175094	NA	NA	0.015792	-0.3662451	NA
preserved ~ stimlen	1387.494	532.0492	0	0	0.0315797	8.0110009	NA	NA	NA	NA	-0.2236432
preserved ~ 1	1421.377	565.9320	0	0	0.0000000	1.4400659	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	0.7767696	185
O	0.8142398	674
P	1.0000000	12
S	0.8260870	69
V	0.8050661	454

```
#  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
##       2.397       -1.811
##
## Degrees of Freedom: 1312 Total (i.e. Null);  1311 Residual
## Null Deviance:       1229
## Residual Deviance: 764   AIC: 825.6
## log likelihood:  -381.9967
## Nagelkerke R2:  0.4906012
## % pres/err predicted correctly:  -224.1786
## % of predictable range [ (model-null)/(1-null) ]:  0.4404203
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.8557       0.3569
##
## Degrees of Freedom: 1312 Total (i.e. Null);  1311 Residual
## Null Deviance:       1229
## Residual Deviance: 1172  AIC: 1291
## log likelihood:  -585.8296
## Nagelkerke R2:  0.07018623
## % pres/err predicted correctly:  -384.4349
## % of predictable range [ (model-null)/(1-null) ]:  0.04217579
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2920      -0.2351
##
## Degrees of Freedom: 1312 Total (i.e. Null);  1311 Residual
## Null Deviance:       1229
## Residual Deviance: 1186  AIC: 1310
## log likelihood:  -592.9824
## Nagelkerke R2:  0.05293062
## % pres/err predicted correctly:  -387.1168
## % of predictable range [ (model-null)/(1-null) ]:  0.0355112
## ***************************
```

```
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.50125      0.01478     -0.36073
##
## Degrees of Freedom: 1312 Total (i.e. Null);  1310 Residual
## Null Deviance:       1229
## Residual Deviance: 1185  AIC: 1311
## log likelihood:  -592.579
## Nagelkerke R2:  0.05390895
## % pres/err predicted correctly:  -387.0584
## % of predictable range [ (model-null)/(1-null) ]:  0.03565627
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       3.086       -0.236
##
## Degrees of Freedom: 1312 Total (i.e. Null);  1311 Residual
## Null Deviance:       1229
## Residual Deviance: 1200  AIC: 1316
## log likelihood:  -600.1116
## Nagelkerke R2:  0.03554419
## % pres/err predicted correctly:  -390.6939
## % of predictable range [ (model-null)/(1-null) ]:  0.02662172
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.423
##
## Degrees of Freedom: 1312 Total (i.e. Null);  1312 Residual
## Null Deviance:       1229
## Residual Deviance: 1229  AIC: 1352
## log likelihood:  -614.4495
## Nagelkerke R2:  -3.653336e-16
## % pres/err predicted correctly:  -401.4067
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

```
                "_OV1_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	825.6284	0.0000	1	1	0.4906012	2.3974765	NA	-1.810908	NA	NA	NA
preserved ~ CumPres	1291.3066	465.6776	0	0	0.0701862	0.8556903	0.3568525	NA	NA	NA	NA
preserved ~ pos	1309.7054	484.0768	0	0	0.0529306	2.2920052	NA	NA	NA	-0.2351169	NA
preserved ~ (I(pos^2) + pos)	1310.6837	485.0553	0	0	0.0539090	2.5012546	NA	NA	0.0147797	-0.3607258	NA
preserved ~ stimlen	1315.6966	490.0682	0	0	0.0355443	3.0857641	NA	NA	NA	NA	-0.2360316
preserved ~ 1	1351.6236	525.9946	0	0	0.0000000	0.4228721	NA	NA	NA	NA	NA

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.320       -1.922
##
## Degrees of Freedom: 1127 Total (i.e. Null);  1126 Residual
## Null Deviance:       1040
## Residual Deviance: 693.9     AIC: 757.8
## log likelihood:  -346.9625
## Nagelkerke R2:  0.4389645
```

```
## % pres/err predicted correctly:  -204.5465
## % of predictable range [ (model-null)/(1-null) ]:  0.3938825
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2712      -0.2254
##
## Degrees of Freedom: 1127 Total (i.e. Null);  1126 Residual
## Null Deviance:       1040
## Residual Deviance: 1005  AIC: 1113
## log likelihood:  -502.4396
## Nagelkerke R2:  0.05137904
## % pres/err predicted correctly:  -326.5834
## % of predictable range [ (model-null)/(1-null) ]:  0.03401884
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.49698      0.01668     -0.36571
##
## Degrees of Freedom: 1127 Total (i.e. Null);  1125 Residual
## Null Deviance:       1040
## Residual Deviance: 1004  AIC: 1114
## log likelihood:  -501.9691
## Nagelkerke R2:  0.05272035
## % pres/err predicted correctly:  -326.4303
## % of predictable range [ (model-null)/(1-null) ]:  0.03447051
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.0017       0.3332
##
## Degrees of Freedom: 1127 Total (i.e. Null);  1126 Residual
## Null Deviance:       1040
## Residual Deviance: 1008  AIC: 1114
## log likelihood:  -504.0995
## Nagelkerke R2:  0.04663743
## % pres/err predicted correctly:  -328.6099
## % of predictable range [ (model-null)/(1-null) ]:  0.02804328
## ***************************
```

```
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.9509      -0.2143
##
## Degrees of Freedom: 1127 Total (i.e. Null);  1126 Residual
## Null Deviance:       1040
## Residual Deviance: 1020  AIC: 1122
## log likelihood:  -510.0001
## Nagelkerke R2:  0.02966939
## % pres/err predicted correctly:  -330.6628
## % of predictable range [ (model-null)/(1-null) ]:  0.02198951
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.454
##
## Degrees of Freedom: 1127 Total (i.e. Null);  1127 Residual
## Null Deviance:       1040
## Residual Deviance: 1040  AIC: 1147
## log likelihood:  -520.1714
## Nagelkerke R2:  -3.686046e-16
## % pres/err predicted correctly:  -338.1199
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

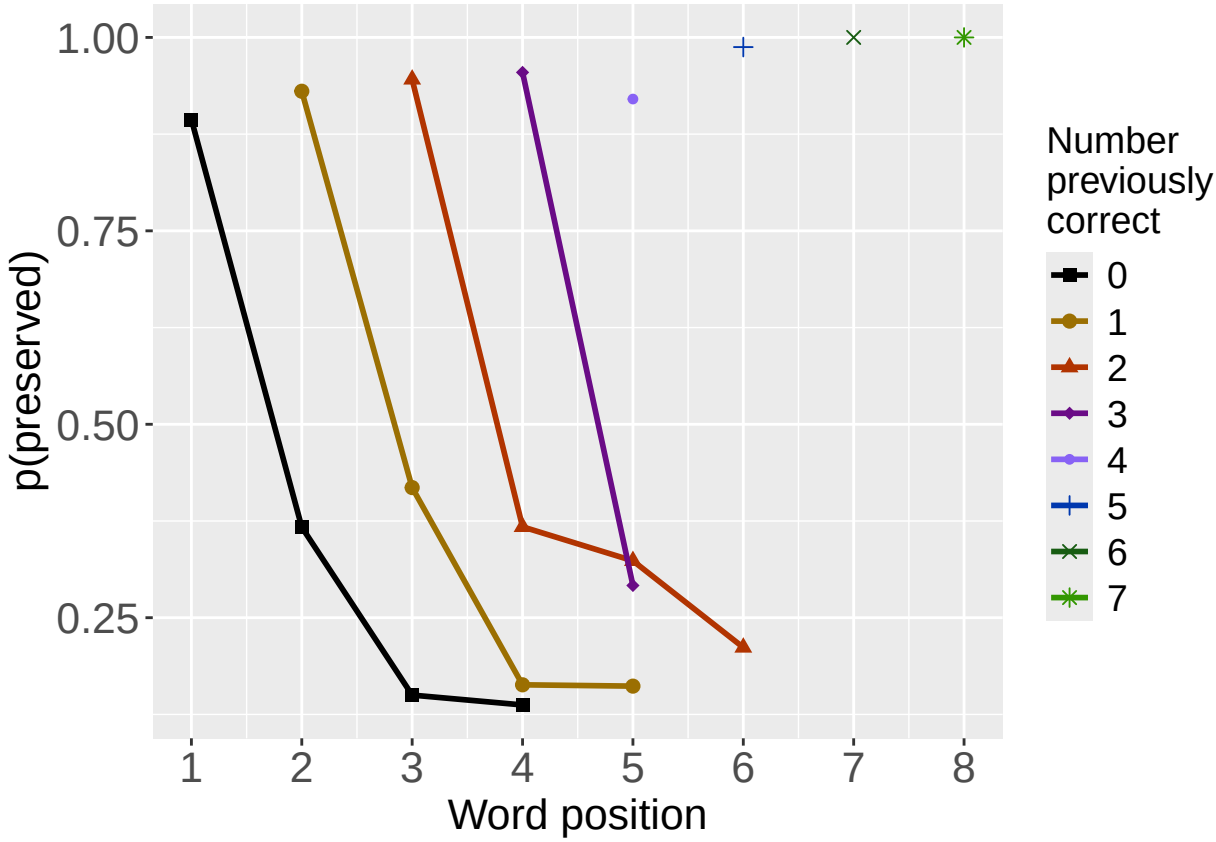
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	757.7869	0.0000	1	1	0.4389643	3.320475	NA	-1.921611	NA	NA	NA
preserved ~ pos	1113.3507	355.5639	0	0	0.0513790	2.271217	NA	NA	NA	-0.2253674	NA
preserved ~ (I(pos^2) + pos)	1114.0263	356.2394	0	0	0.0527203	2.496979	NA	NA	0.0166812	-0.3657070	NA
preserved ~ CumPres	1114.1283	356.3417	0	0	0.0466374	1.001709	0.3332493	NA	NA	NA	NA
preserved ~ stimlen	1122.1979	364.4110	0	0	0.0296692	2.950899	NA	NA	NA	NA	-0.2142998
preserved ~ 1	1147.4123	389.6255	0	0	0.0000000	0.453572	NA	NA	NA	NA	NA

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

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

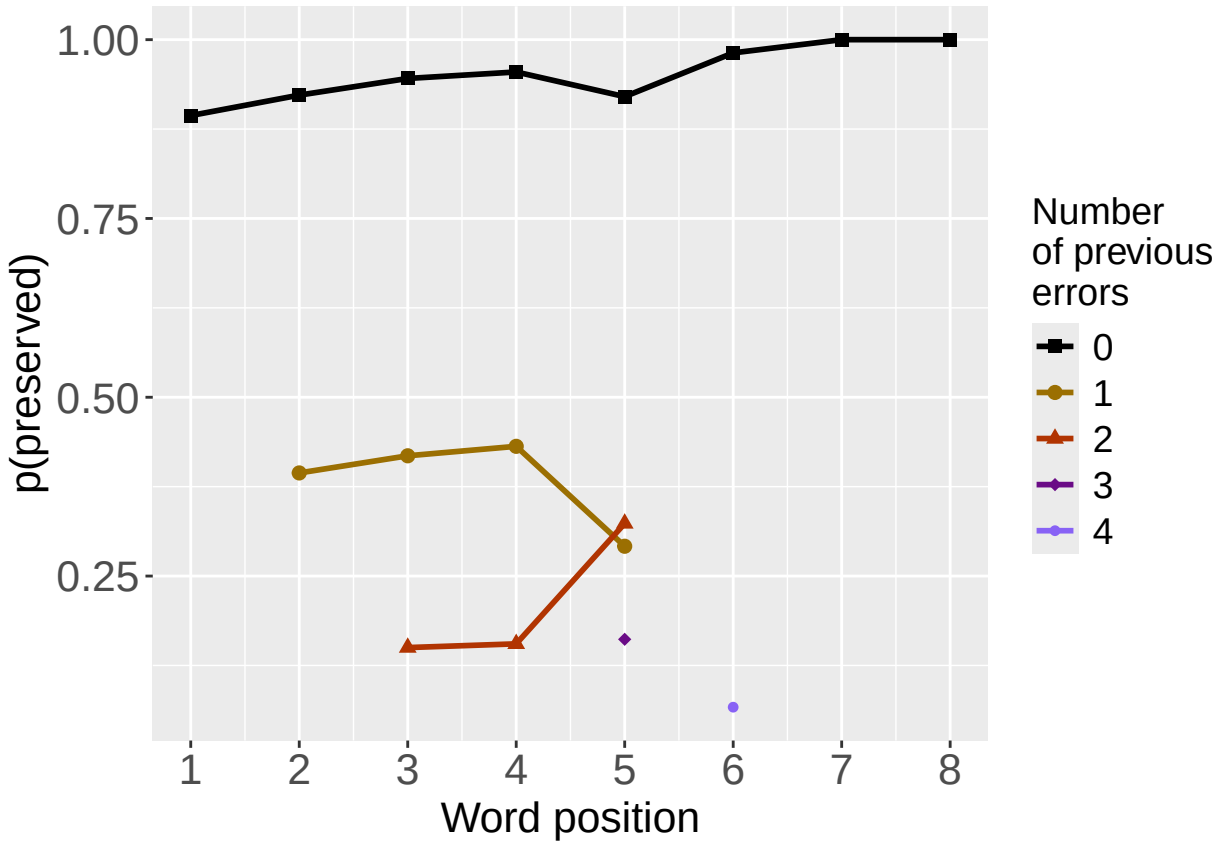
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

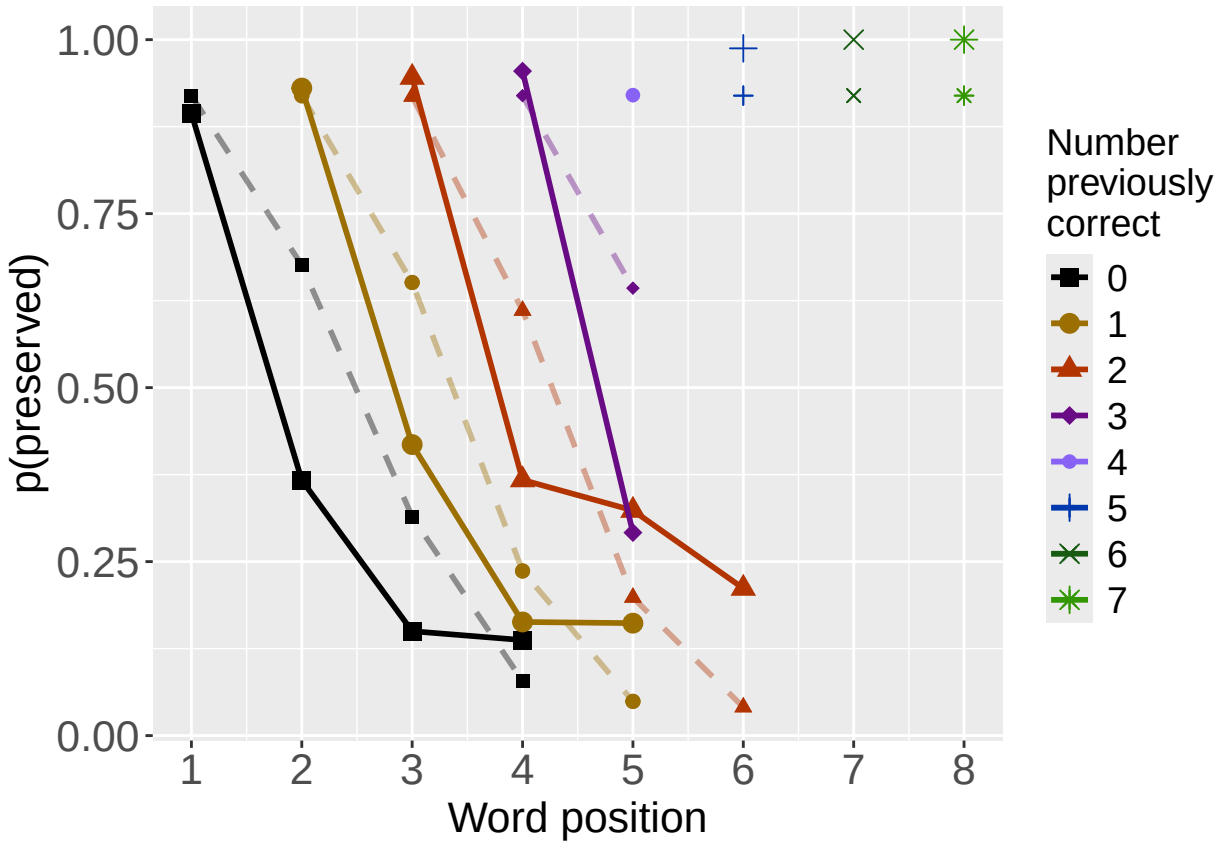
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

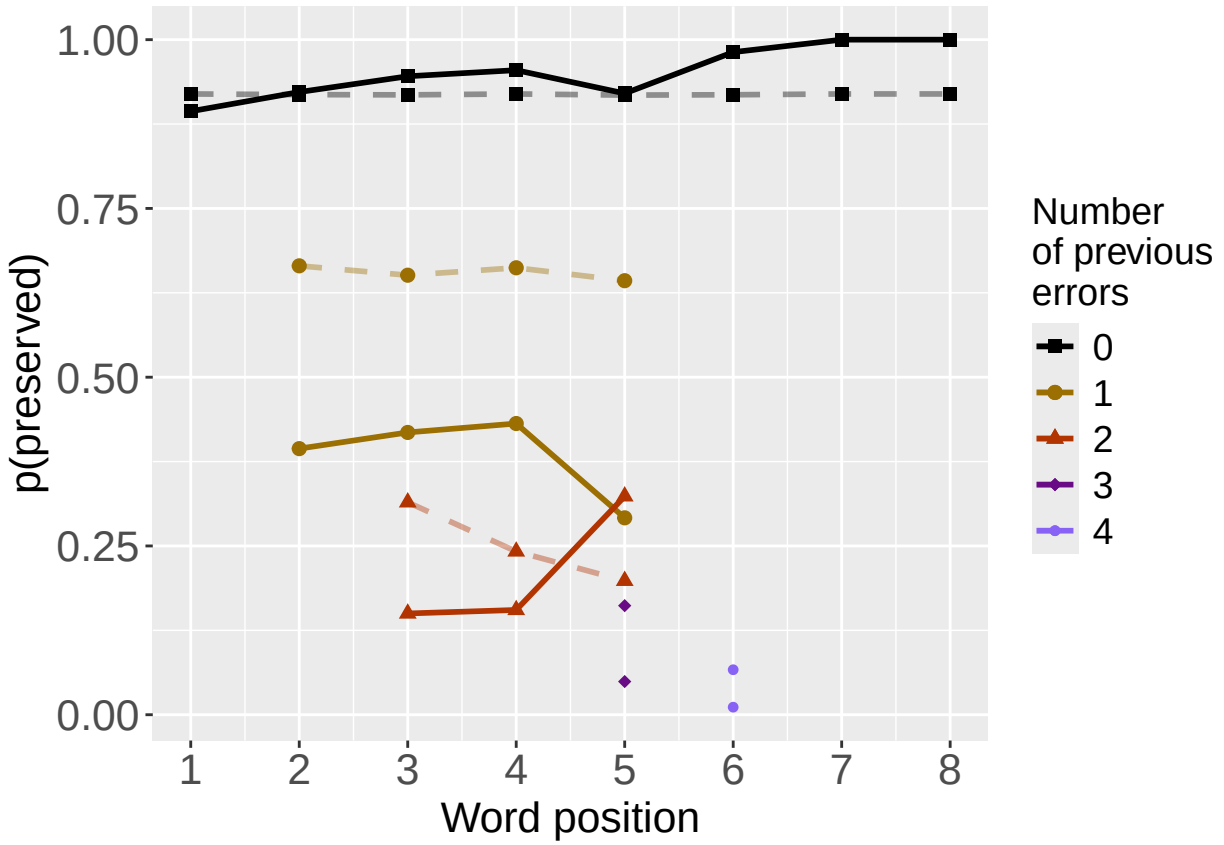
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.21522     -2.00658      0.04824     -0.09832
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 773.9     AIC: 835.9
## log likelihood:  -386.9528
## Nagelkerke R2:  0.5171889
## % pres/err predicted correctly:  -228.6815
## % of predictable range [ (model-null)/(1-null) ]:  0.4573931
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.796
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 795.6     AIC: 855.4
## log likelihood:  -397.7858
## Nagelkerke R2:  0.4994372
## % pres/err predicted correctly:  -233.1762
## % of predictable range [ (model-null)/(1-null) ]:  0.4467748
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.51751      0.01579     -0.36625
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 1253  AIC: 1379
## log likelihood:  -626.486
## Nagelkerke R2:  0.05242125
## % pres/err predicted correctly:  -407.7274
## % of predictable range [ (model-null)/(1-null) ]:  0.03440942
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	835.9081	0.00000	1.00e+00	0.9999428	0.5171889	2.215219	-2.006576	0.0482412	-0.0983183

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr	855.4450	19.53686	5.72e-05	0.0000572	0.4994372	2.436490	-1.795521	NA	NA
preserved ~ I(pos^2) + pos	1379.0287	543.12062	0.00e+00	0.0000000	0.0524213	2.517509	NA	0.0157920	-0.3662451

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.796
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 795.6     AIC: 855.4
## log likelihood:  -397.7858
## Nagelkerke R2:  0.4994372
## % pres/err predicted correctly:  -233.1762
## % of predictable range [ (model-null)/(1-null) ]:  0.4467748
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.63082     -1.78791     -0.02877
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 795.3     AIC: 856.2
## log likelihood:  -397.6659
## Nagelkerke R2:  0.4996353
## % pres/err predicted correctly:  -233.0066
## % of predictable range [ (model-null)/(1-null) ]:  0.4471753
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.0110      -0.2236
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1271  AIC: 1387
## log likelihood:  -635.5173
## Nagelkerke R2:  0.03157967
## % pres/err predicted correctly:  -412.2753
## % of predictable range [ (model-null)/(1-null) ]:  0.02366538
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	855.4450	0.000000	1.000000	0.5920236	0.4994372	2.436490	-1.795521	NA
preserved ~ CumErr + stimlen	856.1897	0.744674	0.689122	0.4079764	0.4996353	2.630815	-1.787907	-0.0287708
preserved ~ stimlen	1387.4942	532.049243	0.000000	0.0000000	0.0315797	3.011001	NA	-0.2236432

```
########
# 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
##      1.9883      -1.7588       0.2503
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
```

```
## Null Deviance:       1298
## Residual Deviance: 777.1     AIC: 836.5
## log likelihood:  -388.5257
## Nagelkerke R2:  0.5146285
## % pres/err predicted correctly:  -229.5703
## % of predictable range [ (model-null)/(1-null) ]:  0.4552933
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.796
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 795.6     AIC: 855.4
## log likelihood:  -397.7858
## Nagelkerke R2:  0.4994372
## % pres/err predicted correctly:  -233.1762
## % of predictable range [ (model-null)/(1-null) ]:  0.4467748
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      0.8994       0.3130
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1247  AIC: 1367
## log likelihood:  -623.4325
## Nagelkerke R2:  0.05940678
## % pres/err predicted correctly:  -407.3372
## % of predictable range [ (model-null)/(1-null) ]:  0.03533132
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	836.5063	0.00000	1.00e+00	0.9999228	0.5146285	1.9883135	-1.758781	0.2502792
preserved ~ CumErr	855.4450	18.93863	7.72e-05	0.0000772	0.4994372	2.4364904	-1.795521	NA
preserved ~ CumPres	1367.3467	530.84033	0.00e+00	0.0000000	0.0594068	0.8993523	NA	0.3130078

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

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      1.7380      -2.0091       0.2503
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 777.1     AIC: 836.5
## log likelihood:  -388.5257
## Nagelkerke R2:  0.5146285
## % pres/err predicted correctly:  -229.5703
## % of predictable range [ (model-null)/(1-null) ]:  0.4552933
## ****************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.796
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 795.6     AIC: 855.4
## log likelihood:  -397.7858
## Nagelkerke R2:  0.4994372
## % pres/err predicted correctly:  -233.1762
## % of predictable range [ (model-null)/(1-null) ]:  0.4467748
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.2957      -0.2325
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 1254  AIC: 1378
## log likelihood:  -626.9605
## Nagelkerke R2:  0.05133276
## % pres/err predicted correctly:  -407.7824
## % of predictable range [ (model-null)/(1-null) ]:  0.03427957
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	836.5063	0.00000	1.00e+00	0.9999228	0.5146285	1.738034	-2.009060	0.2502792
preserved ~ CumErr	855.4450	18.93863	7.72e-05	0.0000772	0.4994372	2.436490	-1.795521	NA
preserved ~ pos	1378.2126	541.70624	0.00e+00	0.0000000	0.0513328	2.295727	NA	-0.2324707

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumErr + I(pos^2) + pos	835.9081	0.000000	1.000000	0.9999942	0.5171882	2.152194	-2.006576	0.0482412	-0.0983183	NA	NA
preserved ~ CumErr + pos	836.5063	0.000000	1.000000	0.9999228	0.5146285	1.738034	-2.009060	NA	0.2502792	NA	NA
preserved ~ CumErr + CumPres	836.5063	0.000000	1.000000	0.9999228	0.5146285	1.598831	-1.758781	NA	NA	NA	0.2502792
preserved ~ CumErr	855.4450	19.536862	2.000057e-05	0.0000077	0.4994372	2.436490	-1.795521	NA	NA	NA	NA
preserved ~ CumErr	855.4450	0.000000	1.000000e+00	0.5920236	0.4994372	2.436490	-1.795521	NA	NA	NA	NA
preserved ~ CumErr	855.4450	18.938629	7.000077e-05	0.0000077	0.4994372	2.436490	-1.795521	NA	NA	NA	NA
preserved ~ CumErr	855.4450	18.938629	7.000077e-05	0.0000077	0.4994372	2.436490	-1.795521	NA	NA	NA	NA
preserved ~ CumErr + stimlen	856.1897	0.744674	0.6891220	0.4079764	0.4499635	2.363082	-1.787907	NA	NA	-0.0287708	NA
preserved ~ CumPres	1367.3467	530.840361	1.000000e+00	0.0000000	0.0059406	1.889935	NA	NA	NA	NA	0.3130078
preserved ~ pos	1378.2126	541.706245	1.000000e+00	0.0000000	0.0513328	2.295727	NA	NA	-0.2324707	NA	NA
preserved ~ I(pos^2) + pos	1379.0285	543.120601	1.000000e+00	0.0000000	0.0524212	2.351751	NA	0.0157920	-0.3662451	NA	NA
preserved ~ stimlen	1387.4945	552.049243	1.000000e+00	0.0000000	0.0031579	2.701100	NA	NA	NA	-0.2236432	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     I(pos^2)          pos      stimlen     log_freq
##     2.63630     -1.91476      0.05892     -0.14883     -0.05153      0.28728
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1388 Residual
## Null Deviance:       1298
## Residual Deviance: 747.9     AIC: 808.3
## log likelihood:  -373.9546
## Nagelkerke R2:  0.5381276
## % pres/err predicted correctly:  -222.7119
## % of predictable range [ (model-null)/(1-null) ]:  0.471496
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.29464     -1.91394      0.05575     -0.13839      0.30217
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1389 Residual
## Null Deviance:       1298
## Residual Deviance: 748.5     AIC: 808.6
## log likelihood:  -374.2606
## Nagelkerke R2:  0.5376391
## % pres/err predicted correctly:  -223.1016
## % of predictable range [ (model-null)/(1-null) ]:  0.4705752
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##      3.1651      -1.9963       0.0578      -0.1293      -0.1427
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1389 Residual
## Null Deviance:       1298
## Residual Deviance: 768.6     AIC: 827.5
## log likelihood:  -384.3152
## Nagelkerke R2:  0.5214694
## % pres/err predicted correctly:  -226.8604
## % of predictable range [ (model-null)/(1-null) ]:  0.4616954
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.21522     -2.00658      0.04824     -0.09832
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 773.9     AIC: 835.9
## log likelihood:  -386.9528
## Nagelkerke R2:  0.5171889
## % pres/err predicted correctly:  -228.6815
## % of predictable range [ (model-null)/(1-null) ]:  0.4573931
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##        1.44
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1393 Residual
## Null Deviance:       1298
## Residual Deviance: 1298  AIC: 1421
## log likelihood:  -648.9827
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -422.2927
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	808.3368	0.0000000	1.0000000	0.5360956	0.5381276	1.636298	-1.914758	0.0589197	-0.1488302	0.2872835	-0.0515341
preserved ~ CumErr + I(pos^2) + pos + log_freq	808.6262	0.2894283	0.8652696	0.4638677	0.5376392	1.294635	-1.913938	0.0557511	-0.1383942	0.3021670	NA
preserved ~ CumErr + I(pos^2) + pos + stimlen	827.5181	19.1812680	0.0000684	0.0000367	0.5214694	1.165074	-1.996306	0.0578008	-0.1293206	NA	-0.1426750
preserved ~ CumErr + I(pos^2) + pos	835.9082	27.5713130	0.0000010	0.0000005	0.5171822	1.215219	-2.006576	0.0482412	-0.0983183	NA	NA
preserved ~ 1	1421.3760	613.0401980	0.0000000	0.0000000	0.0000000	0.440066	NA	NA	NA	NA	NA

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

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

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

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

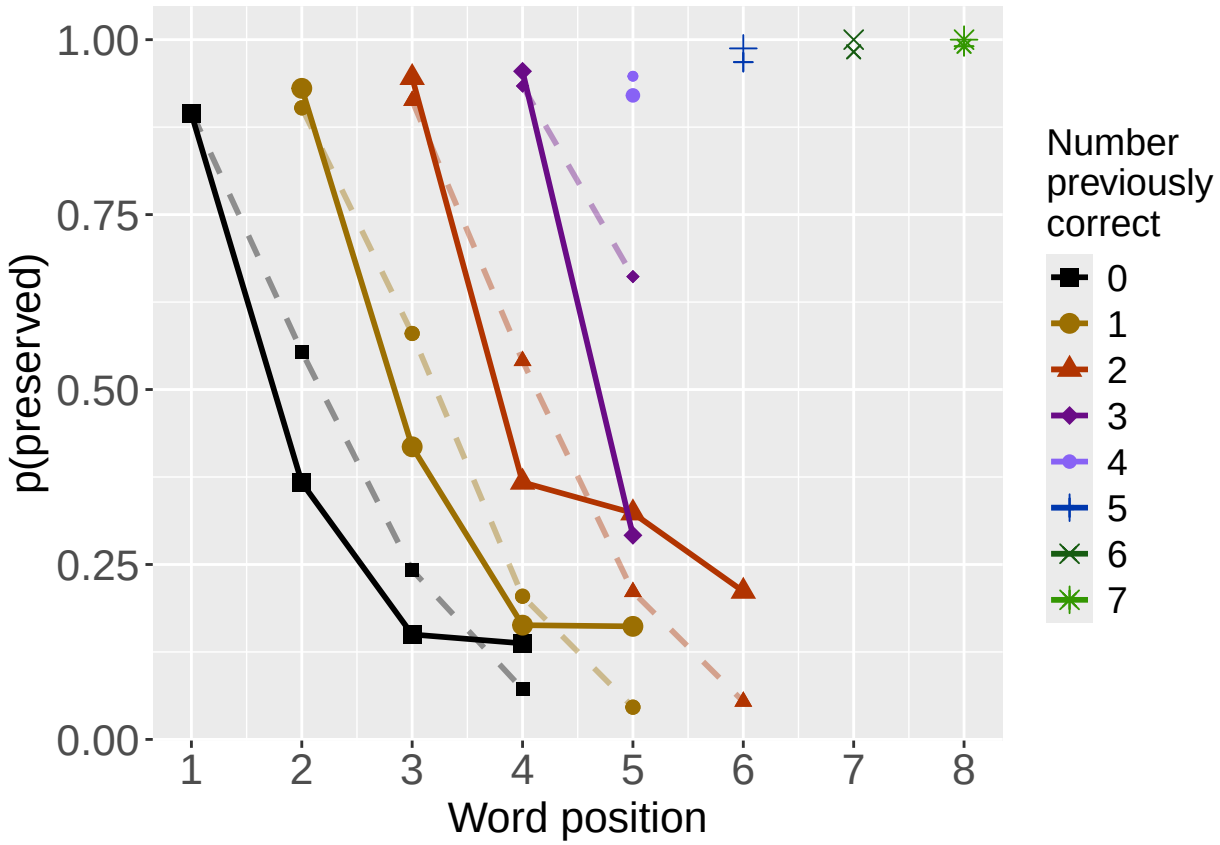
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance     AIC
## CumErr    1  1160.08 1218.51
## log_freq  1   768.63  827.06
## I(pos^2)  1   752.49  810.92
## <none>        747.91  808.34
## stimlen   1   748.52  806.95
## pos       1   748.42  806.84
```

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

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

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

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

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

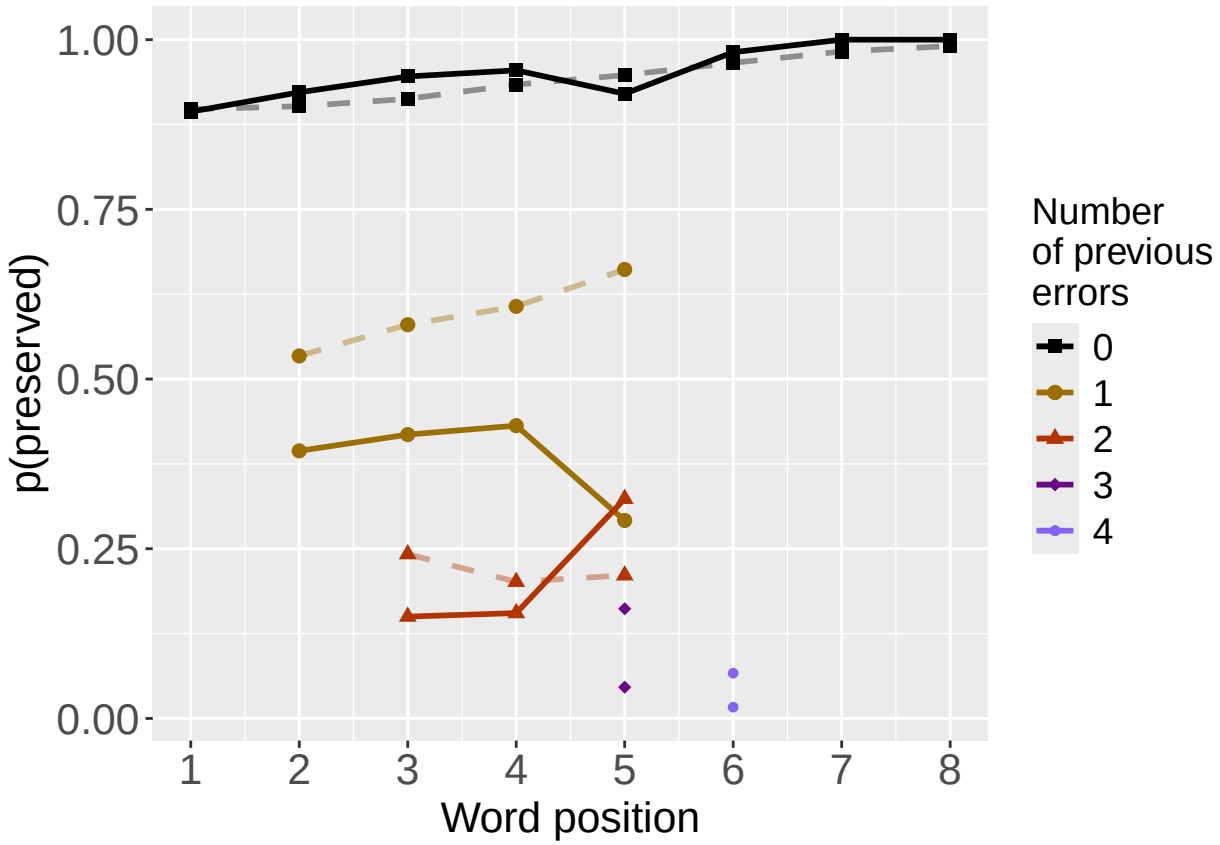
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

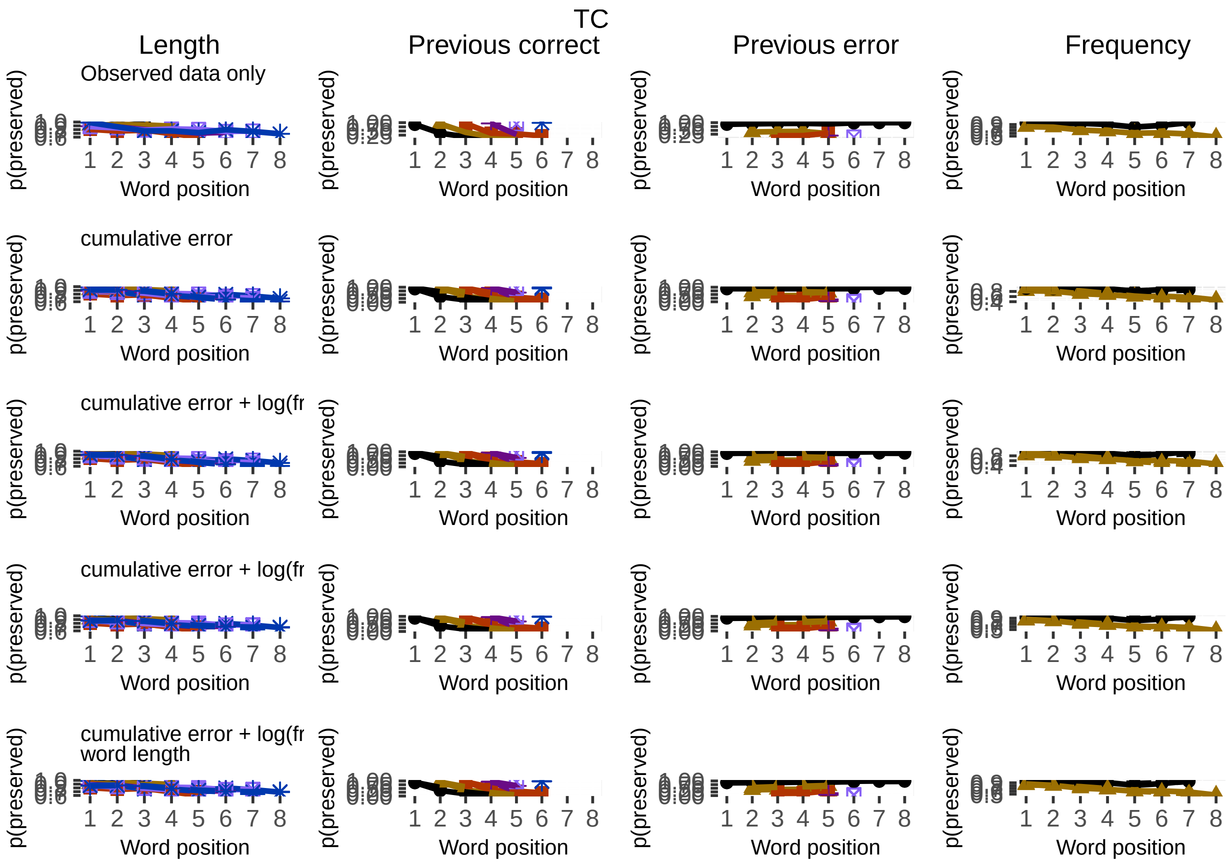
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.436       -1.796
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1392 Residual
## Null Deviance:       1298
## Residual Deviance: 795.6     AIC: 855.4
## log likelihood:  -397.7858
```

```
## Nagelkerke R2:  0.4994372
## % pres/err predicted correctly:  -233.1762
## % of predictable range [ (model-null)/(1-null) ]:  0.4467748
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq
##      2.4738      -1.6970       0.2804
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1391 Residual
## Null Deviance:       1298
## Residual Deviance: 773.2     AIC: 831.4
## log likelihood:  -386.5869
## Nagelkerke R2:  0.5177836
## % pres/err predicted correctly:  -228.0491
## % of predictable range [ (model-null)/(1-null) ]:  0.4588871
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq     I(pos^2)
##     2.08780     -1.92617      0.30028      0.03834
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1390 Residual
## Null Deviance:       1298
## Residual Deviance: 749   AIC: 806.8
## log likelihood:  -374.4805
## Nagelkerke R2:  0.5372879
## % pres/err predicted correctly:  -223.3098
## % of predictable range [ (model-null)/(1-null) ]:  0.4700835
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     log_freq     I(pos^2)      stimlen
##     2.39479     -1.92793      0.28608      0.04009     -0.04852
##
## Degrees of Freedom: 1393 Total (i.e. Null);  1389 Residual
## Null Deviance:       1298
## Residual Deviance: 748.4     AIC: 806.7
## log likelihood:  -374.2076
## Nagelkerke R2:  0.5377237
## % pres/err predicted correctly:  -222.9654
## % of predictable range [ (model-null)/(1-null) ]:  0.470897
```

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

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

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

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

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

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

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

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

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

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

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




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

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

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

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

	CumErr	I(pos^2)	pos	stimlen	log_freq
McFadden	0.3617012	0.0117731	0.0125892	0.0076760	0.0452137
SquaredCorrelation	0.2904899	0.0102369	0.0114494	0.0072375	0.0410066
Nagelkerke	0.4547738	0.0160263	0.0179245	0.0113307	0.0641975

	CumErr	I(posˆ2)	pos	stimlen	log_freq
Estrella	0.3663898	0.0119397	0.0127828	0.0078005	0.0459124

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

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

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

```
##                                                                    model  deviance
## CumErr + log_freq + I(pos^2) + stimlen CumErr + log_freq + I(pos^2) + stimlen  748.4153
## CumErr + log_freq + I(pos^2)                     CumErr + log_freq + I(pos^2)  748.9611
## CumErr + log_freq                                          CumErr + log_freq  773.1739
## CumErr                                                            CumErr  795.5716
## null                                                                null 1297.9653
##                                        deviance_explained percent_explained
## CumErr + log_freq + I(pos^2) + stimlen          549.5501          42.33935
## CumErr + log_freq + I(pos^2)                    549.0042          42.29730
## CumErr + log_freq                               524.7914          40.43185
## CumErr                                          502.3937          38.70625
## null                                              0.0000           0.00000
##                                        percent_of_explained_deviance increment_in_explained
## CumErr + log_freq + I(pos^2) + stimlen                     100.00000             0.09932025
## CumErr + log_freq + I(pos^2)                                99.90068             4.40593355
## CumErr + log_freq                                           95.49475             4.07564592
## CumErr                                                      91.41910            91.41910029
## null                                                              NA             0.00000000
```

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

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

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

	deviance	deviance_explained
CumErr + log_freq + I(pos^2) + stimlen	748.4153	549.5501
CumErr + log_freq + I(pos^2)	748.9611	549.0042
CumErr + log_freq	773.1739	524.7914
CumErr	795.5716	502.3937
null	1297.9653	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + log_freq + I(pos^2) + stimlen	42.33935	100.00000	0.0993203
CumErr + log_freq + I(pos^2)	42.29730	99.90068	4.4059335
CumErr + log_freq	40.43185	95.49475	4.0756459
CumErr	38.70625	91.41910	91.4191003
null	0.00000	NA	0.0000000

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

```
##          Nagelkerke
## CumErr   0.80597528
## I(pos^2) 0.02840276
## pos      0.03176682
## stimlen  0.02008083
## log_freq 0.11377431
```

```
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.
```

model	p_accounted_for	model_deviance
preserved ~ CumErr	0.9331940	795.5716
preserved ~ CumErr+log_freq	0.9436654	773.1739
preserved ~ CumErr+log_freq+I(pos^2)+stimlen	0.9612986	748.4153
preserved ~ CumErr+log_freq+I(pos^2)	0.9616078	748.9611

```
## `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
```

```
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       0.9331940       795.5716  0.00000000
## 2                    preserved ~ CumErr+log_freq       0.9436654       773.1739  0.01047149
## 3 preserved ~ CumErr+log_freq+I(pos^2)+stimlen       0.9612986       748.4153  0.02810464
## 4         preserved ~ CumErr+log_freq+I(pos^2)       0.9616078       748.9611  0.02841379
##   diff_CumErr+log_freq diff_CumErr+log_freq+I(pos^2)+stimlen diff_CumErr+log_freq+I(pos^2)
## 1          -0.01047149                         -0.0281046418                -0.0284137935
## 2           0.00000000                         -0.0176331504                -0.0179423022
## 3           0.01763315                          0.0000000000                -0.0003091517
## 4           0.01794230                          0.0003091517                 0.0000000000
```

```
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)
```

model	diff_CumErr	diff_CumErr+log_freq	diff_CumErr+log_freq+I(pos^2)+stimlen
preserved ~ CumErr	0.0000000	-0.0104715	-0.0281046
preserved ~ CumErr+log_freq	0.0104715	0.0000000	-0.0176332
preserved ~ CumErr+log_freq+I(pos^2)+stimlen	0.0281046	0.0176332	0.0000000
preserved ~ CumErr+log_freq+I(pos^2)	0.0284138	0.0179423	0.0003092

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