DS - repetition - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	536	35	125	NA	NA	696
2	65	NA	430	95	106	696
3	306	NA	166	209	15	696
4	301	NA	231	65	37	634
5	225	NA	209	68	38	540
6	201	1	133	69	20	424
7	172	NA	99	26	19	316
8	87	NA	54	24	4	169
9	74	NA	2	NA	7	83

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.7701149	0.0502874	0.1795977	NA	NA	696
2	0.0933908	NA	0.6178161	0.1364943	0.1522989	696
3	0.4396552	NA	0.2385057	0.3002874	0.0215517	696
4	0.4747634	NA	0.3643533	0.1025237	0.0583596	634
5	0.4166667	NA	0.3870370	0.1259259	0.0703704	540
6	0.4740566	0.0023585	0.3136792	0.1627358	0.0471698	424

pos_factor	O	P	V	1	S	total
7	0.5443038	NA	0.3132911	0.0822785	0.0601266	316
8	0.5147929	NA	0.3195266	0.1420118	0.0236686	169
9	0.8915663	NA	0.0240964	NA	0.0843373	83

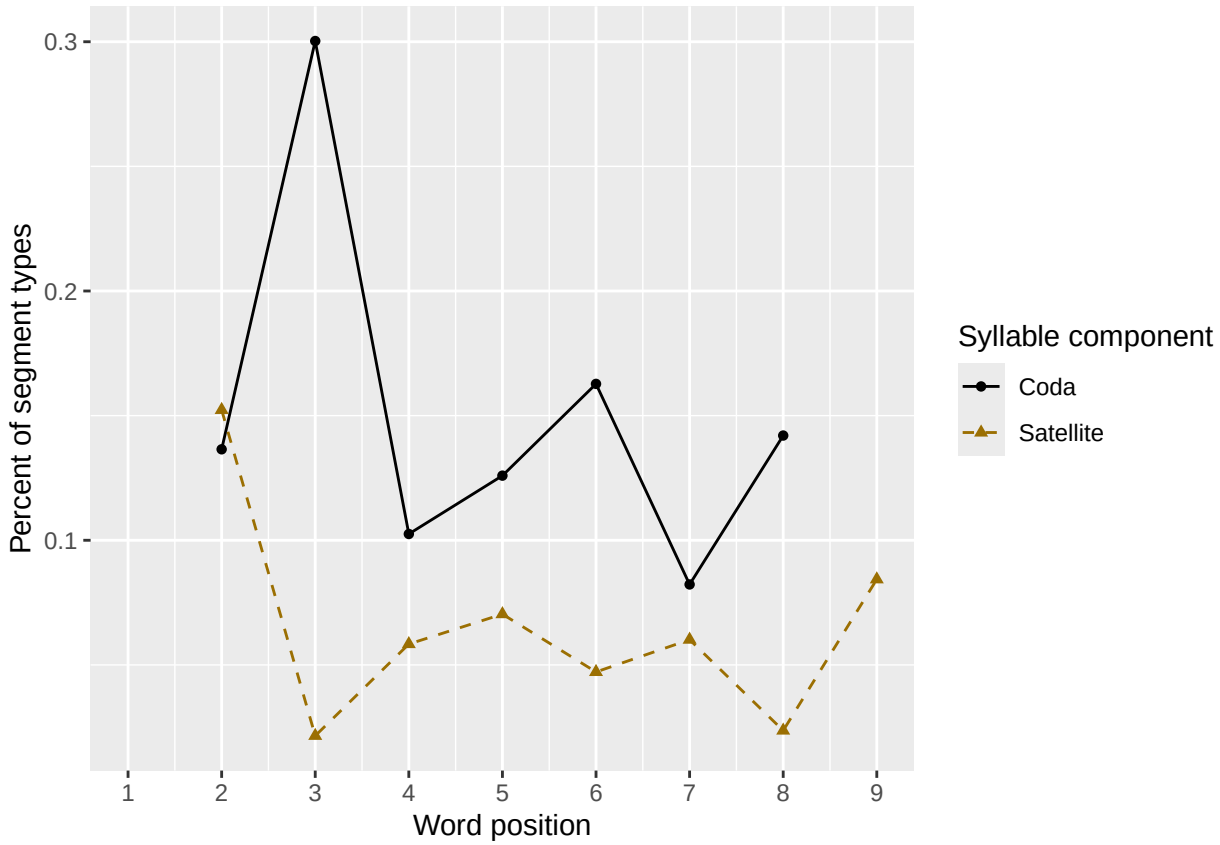
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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

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

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

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

```
syll_comp_plot
```

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 1     1     1     NA     NA     NA     NA     NA     NA
## 2       5 0.995 0.941 0.920  0.920 NA     NA     NA     NA     NA
## 3       6 0.974 0.932 0.899  0.876  0.829 NA     NA     NA     NA
## 4       7 1     0.991 0.903  0.894  0.875  0.815 NA     NA     NA
## 5       8 0.986 0.956 0.926  0.847  0.822  0.832  0.789 NA     NA
## 6       9 0.959 0.915 0.911  0.895  0.843  0.797  0.787  0.771 NA
## 7      10 0.964 0.938 0.896  0.900  0.847  0.745  0.729  0.715  0.701
```

```
# 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.
```

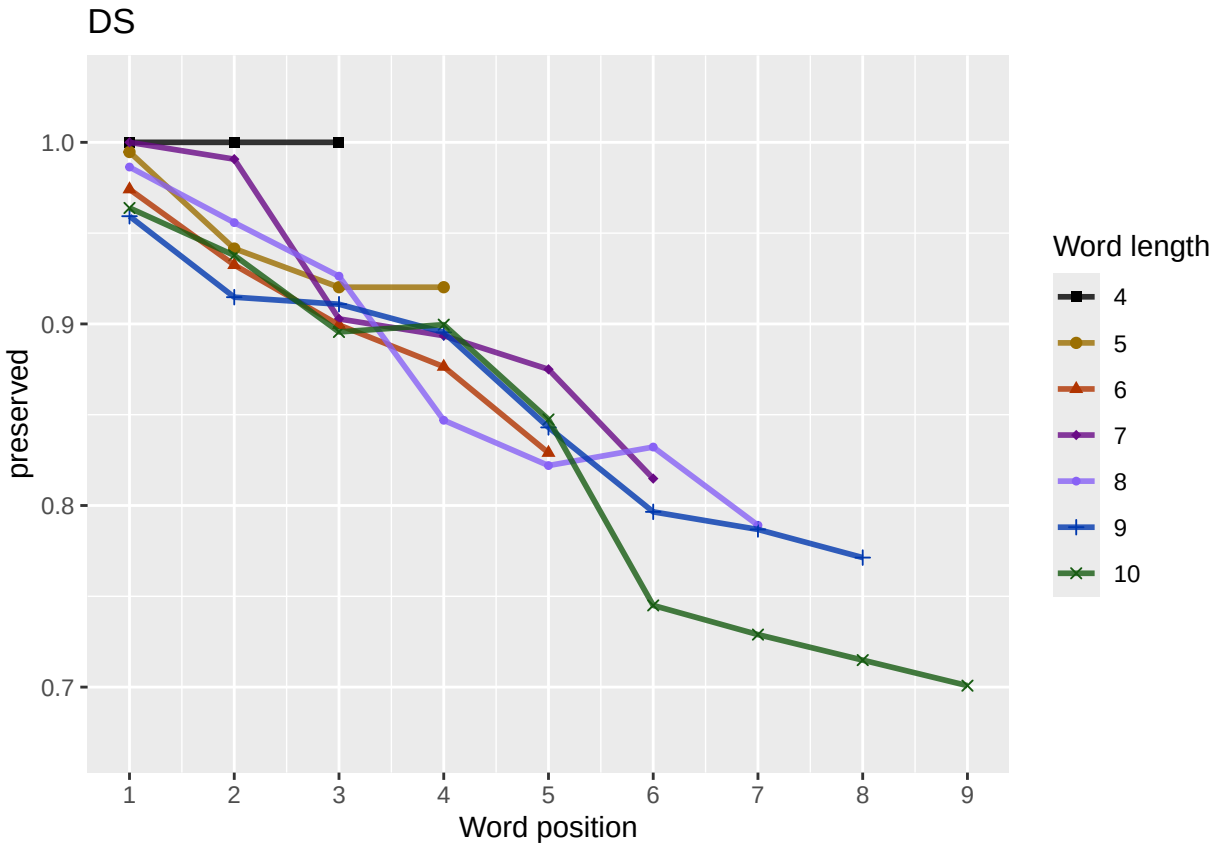
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    62    62    62    NA    NA    NA    NA    NA    NA
## 2       5    94    94    94    94    NA    NA    NA    NA    NA
## 3       6   116   116   116   116   116    NA    NA    NA    NA
## 4       7   108   108   108   108   108   108    NA    NA    NA
## 5       8   147   147   147   147   147   147   147    NA    NA
## 6       9    86    86    86    86    86    86    86    86    NA
## 7      10    83    83    83    83    83    83    83    83    83
```

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

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

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



Length and position

```
# length and position

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

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

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

```
## **************************
## model index:  8
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##           8.1928           -0.4490            0.2320           -2.3972           -0.0204
##      stimlen:pos
##           0.1824
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 2627  AIC: 2913
## log likelihood:  -1313.489
## Nagelkerke R2:  0.1222162
## % pres/err predicted correctly:  -794.4367
## % of predictable range [ (model-null)/(1-null) ]:  0.06246842
## **************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.15716     -0.09600      0.04902     -0.79236
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 2633  AIC: 2917
## log likelihood:  -1316.367
## Nagelkerke R2:  0.1196361
## % pres/err predicted correctly:  -794.9783
## % of predictable range [ (model-null)/(1-null) ]:  0.06183007
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.46340      0.04439     -0.78185
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2639  AIC: 2921
## log likelihood:  -1319.701
## Nagelkerke R2:  0.1166434
## % pres/err predicted correctly:  -796.0331
## % of predictable range [ (model-null)/(1-null) ]:  0.06058684
## **************************
## model index:  5
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.75186     -0.27502     -0.77092      0.05201
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 2644  AIC: 2931
## log likelihood:  -1322.205
## Nagelkerke R2:  0.114393
## % pres/err predicted correctly:  -795.969
## % of predictable range [ (model-null)/(1-null) ]:  0.06066243
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.97338     -0.06587     -0.31656
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2654  AIC: 2940
## log likelihood:  -1327.011
## Nagelkerke R2:  0.1100655
## % pres/err predicted correctly:  -795.7374
## % of predictable range [ (model-null)/(1-null) ]:  0.06093539
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.5615      -0.3408
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2657  AIC: 2940
## log likelihood:  -1328.65
## Nagelkerke R2:  0.1085872
## % pres/err predicted correctly:  -796.4665
## % of predictable range [ (model-null)/(1-null) ]:  0.06007599
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      4.0676      -0.2544
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2819  AIC: 3107
## log likelihood:  -1409.552
## Nagelkerke R2:  0.03419692
## % pres/err predicted correctly:  -833.3831
## % of predictable range [ (model-null)/(1-null) ]:  0.01656476
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.052
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4253 Residual
## Null Deviance:       2891
## Residual Deviance: 2891  AIC: 3173
## log likelihood:  -1445.734
## Nagelkerke R2:  2.250906e-16
## % pres/err predicted correctly:  -847.4372
## % 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)	2912.633	0.000000	1.000000	0.8918128	0.0122216	2.192754	-0.4489982	-3.972017	0.1824245	0.2320105	-0.0203969
preserved ~ stimlen + I(pos^2) + pos	2917.174	4.544121	0.103099	0.0919455	0.0119636	1.157163	-0.0960048	-7.923591	NA	0.0490195	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	2920.658	8.024182	0.0180955	0.0161378	0.1166434	4.463396	NA	-0.7818487	NA	0.0443873	NA
preserved ~ stimlen * pos	2930.782	18.14848	0.0001146	0.0001022	0.1143930	5.751865	-0.2750183	-0.7709210	0.0520118	NA	NA
preserved ~ stimlen + pos	2940.282	27.65514	0.0000010	0.0000009	0.1100635	5.973376	-0.0658710	-0.3165579	NA	NA	NA
preserved ~ pos	2940.462	27.83519	0.0000009	0.0000008	0.1085872	2.561465	NA	-0.3408272	NA	NA	NA
preserved ~ stimlen	3107.271	194.6409	0.0000000	0.0000000	0.0341964	1.067586	-0.2543557	NA	NA	NA	NA
preserved ~ 1	3172.644	260.0103	0.0000000	0.0000000	0.0000000	2.051969	NA	NA	NA	NA	NA

```r
print(BestLPModelFormula)
```

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

```r
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##           8.1928           -0.4490            0.2320           -2.3972           -0.0204
##      stimlen:pos
##           0.1824
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 2627  AIC: 2913
```

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

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

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

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

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

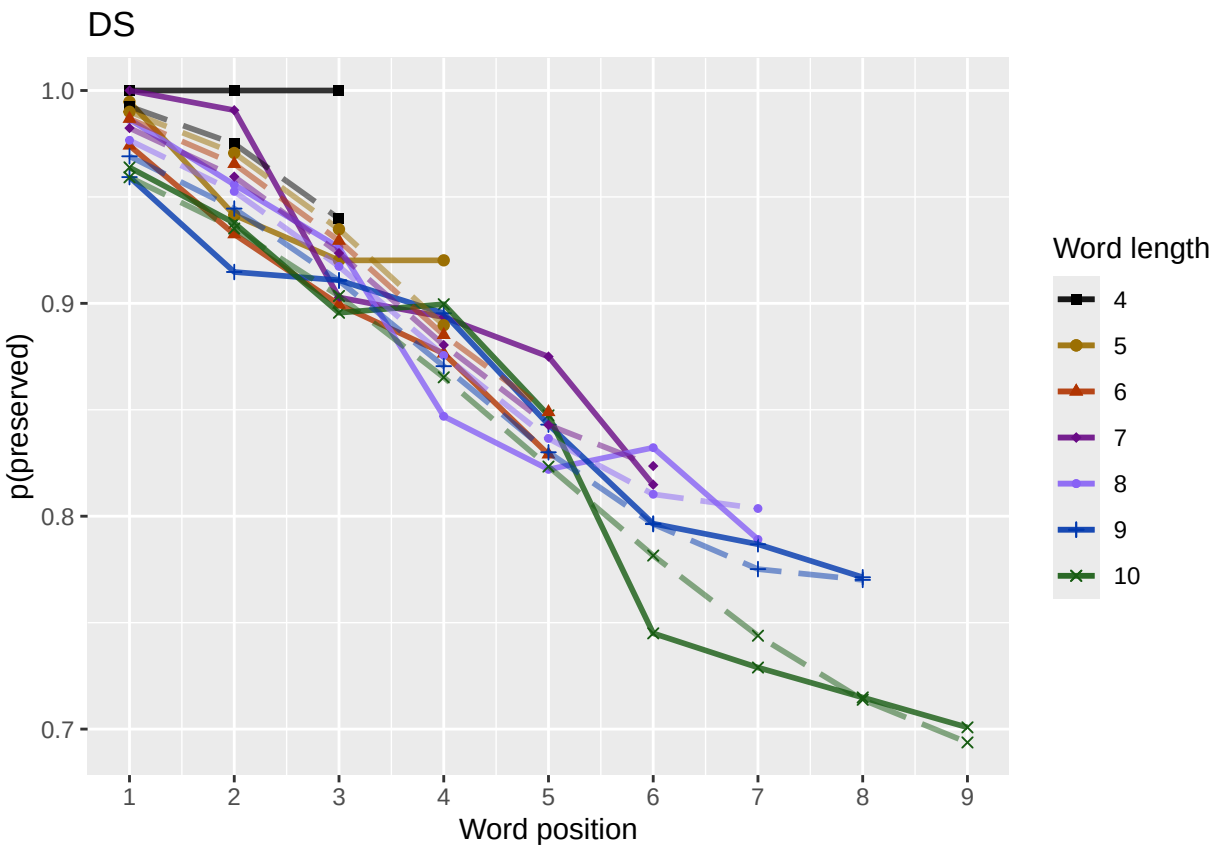
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`
##     <int> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1       4 0.992 0.975 0.940 NA     NA     NA     NA     NA     NA
## 2       5 0.990 0.971 0.935  0.890 NA     NA     NA     NA     NA
## 3       6 0.987 0.966 0.929  0.885  0.849 NA     NA     NA     NA
## 4       7 0.982 0.960 0.924  0.880  0.843  0.824 NA     NA     NA
```

```
## 5       8 0.977 0.953 0.917  0.876  0.837  0.810  0.804 NA     NA
## 6       9 0.969 0.945 0.911  0.870  0.830  0.796  0.775  0.770 NA
## 7      10 0.959 0.935 0.903  0.865  0.823  0.782  0.744  0.714  0.694
```

```
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       79   696
```

```
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 79 / 696 = 11.35 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:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##           8.1014           -0.4595            0.2577           -2.3702           -0.0222
##      stimlen:pos
##           0.1934
##
## Degrees of Freedom: 4004 Total (i.e. Null);  3999 Residual
## Null Deviance:       1756
## Residual Deviance: 1699  AIC: 1950
## log likelihood:  -849.7471
## Nagelkerke R2:  0.03928694
## % pres/err predicted correctly:  -450.5256
## % of predictable range [ (model-null)/(1-null) ]:  0.01354577
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##      5.1223      -0.1025       0.0645      -0.7216
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4001 Residual
## Null Deviance:       1756
## Residual Deviance: 1704  AIC: 1953
## log likelihood:  -851.7862
## Nagelkerke R2:  0.03645661
## % pres/err predicted correctly:  -450.828
## % of predictable range [ (model-null)/(1-null) ]:  0.01288509
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.37296      0.05916     -0.70650
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4002 Residual
## Null Deviance:       1756
## Residual Deviance: 1709  AIC: 1955
## log likelihood:  -854.3918
## Nagelkerke R2:  0.0328356
## % pres/err predicted correctly:  -451.3005
## % of predictable range [ (model-null)/(1-null) ]:  0.01185275
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.83203     -0.30602     -0.71523      0.06629
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4001 Residual
## Null Deviance:       1756
## Residual Deviance: 1714  AIC: 1965
## log likelihood:  -856.9016
## Nagelkerke R2:  0.02934326
## % pres/err predicted correctly:  -452.0926
## % of predictable range [ (model-null)/(1-null) ]:  0.0101222
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.3993      -0.1676
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4003 Residual
## Null Deviance:       1756
## Residual Deviance: 1725  AIC: 1975
## log likelihood:  -862.6687
## Nagelkerke R2:  0.02130193
## % pres/err predicted correctly:  -453.1296
## % of predictable range [ (model-null)/(1-null) ]:  0.007856667
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.84980     -0.07002     -0.14436
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4002 Residual
## Null Deviance:       1756
## Residual Deviance: 1723  AIC: 1975
## log likelihood:  -861.4249
## Nagelkerke R2:  0.02303819
## % pres/err predicted correctly:  -452.9028
## % of predictable range [ (model-null)/(1-null) ]:  0.00835222
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.8475      -0.1441
```

```
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4003 Residual
## Null Deviance:       1756
## Residual Deviance: 1742  AIC: 1995
## log likelihood:  -871.1697
## Nagelkerke R2:  0.009406161
## % pres/err predicted correctly:  -455.2157
## % of predictable range [ (model-null)/(1-null) ]:  0.003299238
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.727
##
## Degrees of Freedom: 4004 Total (i.e. Null);  4004 Residual
## Null Deviance:       1756
## Residual Deviance: 1756  AIC: 2004
## log likelihood:  -877.8662
## Nagelkerke R2:  -6.25614e-16
## % pres/err predicted correctly:  -456.7258
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * (I(pos^2) + pos)	1950.463	0.000000	1.000000	0.7455540	0.0392869	2.9101406	-0.4594977	-7.3701954	0.1934100	0.2577420	-0.0222032
preserved ~ stimlen + I(pos^2) + pos	1953.217	2.753926	0.2523487	0.1881361	0.0103645	2.6122271	-0.1024967	-7.7215900	NA	0.0645041	NA
preserved ~ I(pos^2) + pos	1955.314	4.852079	0.0883862	0.0658968	0.0032835	2.6372964	NA	-0.7065030	NA	0.0591595	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen * pos	1965.498	15.034254	0.0005437	0.0004054	0.0293433	3.3832031	-0.3060236	-0.7152259	0.0662909	NA	NA
preserved ~ pos	1974.752	24.292405	0.0000053	0.0000040	0.0213019	3.399312	NA	-0.1676393	NA	NA	NA
preserved ~ stimlen + pos	1975.332	24.868939	0.0000040	0.0000030	0.0230382	3.2849804	-0.0700207	-0.1443602	NA	NA	NA
preserved ~ stimlen	1994.564	44.101724	0.0000000	0.0000000	0.0094063	2.847451	-0.1441467	NA	NA	NA	NA
preserved ~ 1	2003.923	53.459491	0.0000000	0.0000000	0.0000000	2.0726577	NA	NA	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.
```

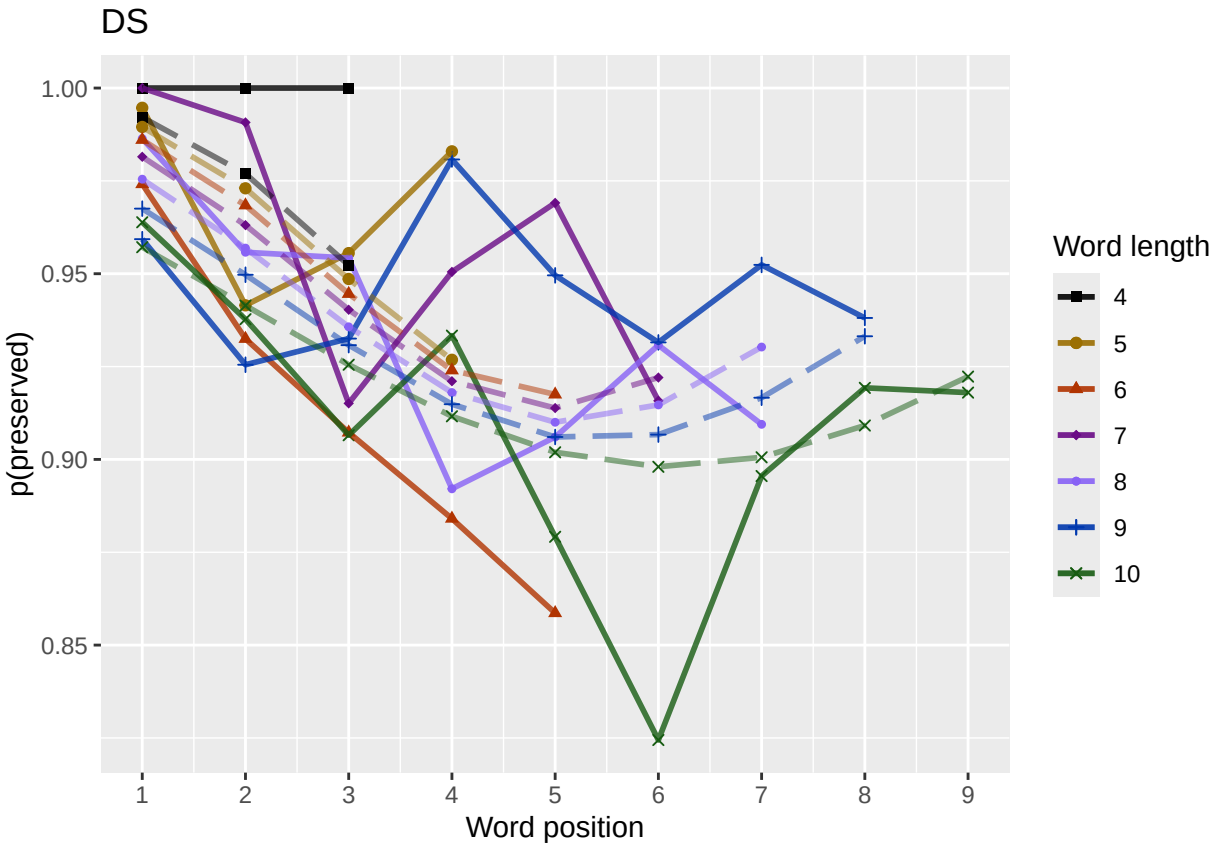
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.992 0.977 0.952 NA     NA     NA     NA     NA     NA
## 2       5 0.990 0.973 0.949  0.927 NA     NA     NA     NA     NA
## 3       6 0.986 0.968 0.945  0.924  0.917 NA     NA     NA     NA
## 4       7 0.981 0.963 0.940  0.921  0.914  0.922 NA     NA     NA
## 5       8 0.975 0.957 0.936  0.918  0.910  0.915  0.930 NA     NA
## 6       9 0.968 0.950 0.931  0.915  0.906  0.907  0.917  0.933 NA
## 7      10 0.957 0.941 0.925  0.912  0.902  0.898  0.901  0.909  0.922
```

```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: 0.67 - 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
```

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

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

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

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

```
print(OA_mean_len_diff)
```

```
## [1] -0.008276479
```

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

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

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

```
print(OA_mean_pos_diff)
```

```
## [1] -0.02845441
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen          I(pos^2)              pos  stimlen:I(pos^2)
##          8.1928          -0.4490            0.2320          -2.3972           -0.0204
##     stimlen:pos
##          0.1824
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 2627  AIC: 2913
## log likelihood:  -1313.489
## Nagelkerke R2:  0.1222162
## % pres/err predicted correctly:  -794.4367
## % of predictable range [ (model-null)/(1-null) ]:  0.06246842
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     5.15716     -0.09600      0.04902     -0.79236
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
```

```
## Residual Deviance: 2633  AIC: 2917
## log likelihood:  -1316.367
## Nagelkerke R2:  0.1196361
## % pres/err predicted correctly:  -794.9783
## % of predictable range [ (model-null)/(1-null) ]:  0.06183007
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     5.06212     -0.08375      0.04883     -0.79057      0.03885
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4249 Residual
## Null Deviance:       2891
## Residual Deviance: 2631  AIC: 2917
## log likelihood:  -1315.352
## Nagelkerke R2:  0.1205464
## % pres/err predicted correctly:  -794.5873
## % of predictable range [ (model-null)/(1-null) ]:  0.06229097
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          5.01582          -0.08143           0.18574           0.04810          -0.78321
## stimlen:log_freq
##         -0.01785
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 2630  AIC: 2918
## log likelihood:  -1314.825
## Nagelkerke R2:  0.1210186
## % pres/err predicted correctly:  -794.6283
## % of predictable range [ (model-null)/(1-null) ]:  0.06224258
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.46340      0.04439     -0.78185
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2639  AIC: 2921
```

```
## log likelihood:  -1319.701
## Nagelkerke R2:  0.1166434
## % pres/err predicted correctly:  -796.0331
## % of predictable range [ (model-null)/(1-null) ]:  0.06058684
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)               pos           log_freq
##         5.037e+00         -8.068e-02          4.818e-02         -7.874e-01          7.611e-02
## I(pos^2):log_freq       pos:log_freq
##         4.964e-05         -7.580e-03
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4247 Residual
## Null Deviance:       2891
## Residual Deviance: 2630  AIC: 2921
## log likelihood:  -1315.21
## Nagelkerke R2:  0.1206736
## % pres/err predicted correctly:  -794.6215
## % of predictable range [ (model-null)/(1-null) ]:  0.06225056
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##         5.0190400         -0.0812261          0.2077465          0.0483269         -0.7854969
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##        -0.0177281          0.0008893         -0.0097510
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4246 Residual
## Null Deviance:       2891
## Residual Deviance: 2630  AIC: 2922
## log likelihood:  -1314.811
## Nagelkerke R2:  0.1210311
## % pres/err predicted correctly:  -794.6285
## % of predictable range [ (model-null)/(1-null) ]:  0.06224231
## ***************************
## 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
##         4.4476227          0.0440324         -0.7755612          0.1068759         -0.0001201
##      pos:log_freq
##        -0.0100859
##
```

```
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 2635  AIC: 2923
## log likelihood:  -1317.38
## Nagelkerke R2:  0.1187272
## % pres/err predicted correctly:  -795.2995
## % of predictable range [ (model-null)/(1-null) ]:  0.06145155
## **************************
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     5.75186     -0.27502     -0.77092      0.05201
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 2644  AIC: 2931
## log likelihood:  -1322.205
## Nagelkerke R2:  0.114393
## % pres/err predicted correctly:  -795.969
## % of predictable range [ (model-null)/(1-null) ]:  0.06066243
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##     3.54250     -0.33514      0.04963
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2654  AIC: 2939
## log likelihood:  -1326.906
## Nagelkerke R2:  0.1101596
## % pres/err predicted correctly:  -795.8061
## % of predictable range [ (model-null)/(1-null) ]:  0.06085437
## **************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     3.87885     -0.05311     -0.31675      0.04068
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 2652  AIC: 2940
```

```
## log likelihood:  -1325.901
## Nagelkerke R2:  0.1110657
## % pres/err predicted correctly:  -795.3489
## % of predictable range [ (model-null)/(1-null) ]:  0.06139326
## **************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     3.97338     -0.06587     -0.31656
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2654  AIC: 2940
## log likelihood:  -1327.011
## Nagelkerke R2:  0.1100655
## % pres/err predicted correctly:  -795.7374
## % of predictable range [ (model-null)/(1-null) ]:  0.06093539
## **************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.5615      -0.3408
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2657  AIC: 2940
## log likelihood:  -1328.65
## Nagelkerke R2:  0.1085872
## % pres/err predicted correctly:  -796.4665
## % of predictable range [ (model-null)/(1-null) ]:  0.06007599
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          3.84194          -0.05049           0.22497          -0.31665          -0.02248
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4249 Residual
## Null Deviance:       2891
## Residual Deviance: 2650  AIC: 2941
## log likelihood:  -1325.02
## Nagelkerke R2:  0.1118594
## % pres/err predicted correctly:  -795.4141
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.06131641
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##       3.5590       -0.3397        0.1237       -0.0149
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 2652  AIC: 2941
## log likelihood:  -1326.15
## Nagelkerke R2:  0.1108409
## % pres/err predicted correctly:  -795.8378
## % of predictable range [ (model-null)/(1-null) ]:  0.06081699
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      3.85821      -0.04766      -0.32251       0.10469      -0.01270
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4249 Residual
## Null Deviance:       2891
## Residual Deviance: 2651  AIC: 2942
## log likelihood:  -1325.358
## Nagelkerke R2:  0.1115551
## % pres/err predicted correctly:  -795.4366
## % of predictable range [ (model-null)/(1-null) ]:  0.06128992
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         3.840528         -0.048625          0.221558         -0.319337         -0.018191
##     log_freq:pos
##        -0.006295
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 2650  AIC: 2943
## log likelihood:  -1324.919
## Nagelkerke R2:  0.1119499
## % pres/err predicted correctly:  -795.4407
## % of predictable range [ (model-null)/(1-null) ]:  0.06128509
```

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     3.97767     -0.24232      0.03878
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2817  AIC: 3107
## log likelihood:  -1408.491
## Nagelkerke R2:  0.03519026
## % pres/err predicted correctly:  -833.0688
## % of predictable range [ (model-null)/(1-null) ]:  0.01693523
## ***************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.0676      -0.2544
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2819  AIC: 3107
## log likelihood:  -1409.552
## Nagelkerke R2:  0.03419692
## % pres/err predicted correctly:  -833.3831
## % of predictable range [ (model-null)/(1-null) ]:  0.01656476
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          3.94136          -0.23969           0.22274          -0.02232
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 2815  AIC: 3108
## log likelihood:  -1407.586
## Nagelkerke R2:  0.03603767
## % pres/err predicted correctly:  -833.1291
## % of predictable range [ (model-null)/(1-null) ]:  0.01686407
## ***************************
## model index:  14
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.052
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4253 Residual
## Null Deviance:       2891
## Residual Deviance: 2891  AIC: 3173
## log likelihood:  -1445.734
## Nagelkerke R2:  2.250906e-16
## % pres/err predicted correctly:  -847.4372
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

Model	AIC DeltaAIC AICexp AICwt NagR2 (Intercept)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	-0.4489983	NA	NA	-2.3972017	NA	NA	0.2320105	NA	NA	0.1824245	-0.0203969
preserved ~ stimlen + I(pos^2) + pos	[illegible]	-0.0960048	NA	NA	-0.7923591	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	-0.0837510	[illegible]	NA	-0.7905701	NA	NA	[illegible]	NA	NA	NA	NA
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	-0.0814316	0.1857393	[illegible]	-0.7832076	NA	NA	[illegible]	NA	NA	NA	NA

Model	AIC Delta_AIC 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 ~ I(pos^2) + pos	2920.8582410821809553891664463396	NA	NA	NA	-0.7818487	NA	NA	0.0443873	NA	NA	NA	NA
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	2921.4826670120104793012705737354	-0.0806763	0.0761412	NA	-0.7873709	-0.0975798	NA	0.0480183	0.0004996	NA	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	2922.4759104307479657442153119040	-0.0812261	0.2077465	-0.0177287	-0.7854969	NA	-0.0097510	0.0483269	NA	0.0008893	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	2923.2053692715068938912784727621	NA	0.1068759	NA	-0.7755612	-0.0100859	NA	0.0440324	-0.0001201	NA	NA	NA
preserved ~ stimlen * pos	2930.7821484810010460088045931865	-0.2750183	NA	NA	-0.7709210	NA	NA	NA	NA	NA	0.0520118	NA
preserved ~ pos + log_freq	2939.2986650490001600112035942504	NA	0.0496250	NA	-0.3351436	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos + log_freq	2940.2124950911000110001813678848	-0.0531055	0.0406784	NA	-0.3167530	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	2940.2789551480001000108036573376	-0.0658718	NA	NA	-0.3165579	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	2940.2763351094000090010785872461	NA	NA	NA	-0.3408272	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos	2940.2841070940000800116135941937	-0.0504863	0.2249703	-0.0224753	-0.3166462	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos * log_freq	2940.3870237060000700116034590504	NA	0.1236984	NA	-0.3397031	-0.0149009	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos * log_freq	2941.3972407570000400113135558211	-0.0476619	0.1046905	NA	-0.3225006	-0.0127026	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos * log_freq	2942.3021690090000300112934940528	-0.0486251	0.2215583	-0.0181906	-0.3193371	NA	-0.0062946	NA	NA	NA	NA	NA
preserved ~ stimlen + log_freq	3107.1924578062000000003539737666	-0.2423175	0.0387812	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	3107.2746409924000000003449697586	-0.2543557	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Model	AIC DeltaAICAICexpAICwNagR2(Intercept)stimlenlog_freqstimlen:log_freqposlog_freq:posI(pos^2)log_freq:I(pos^2)stimlen:posstimlen:I(pos^2)										
preserved ~ stimlen * log_freq	3107.6820.0000.0000000.0000.0363371359	-0.2396881	0.2227409	-0.0223237	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	3172.26464.0000.0000000.0000000.0002.051969	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##     (Intercept)          stimlen         I(pos^2)              pos  stimlen:I(pos^2)
##          8.1928          -0.4490           0.2320          -2.3972           -0.0204
##     stimlen:pos
##          0.1824
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:      2891
## Residual Deviance: 2627  AIC: 2913
```

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

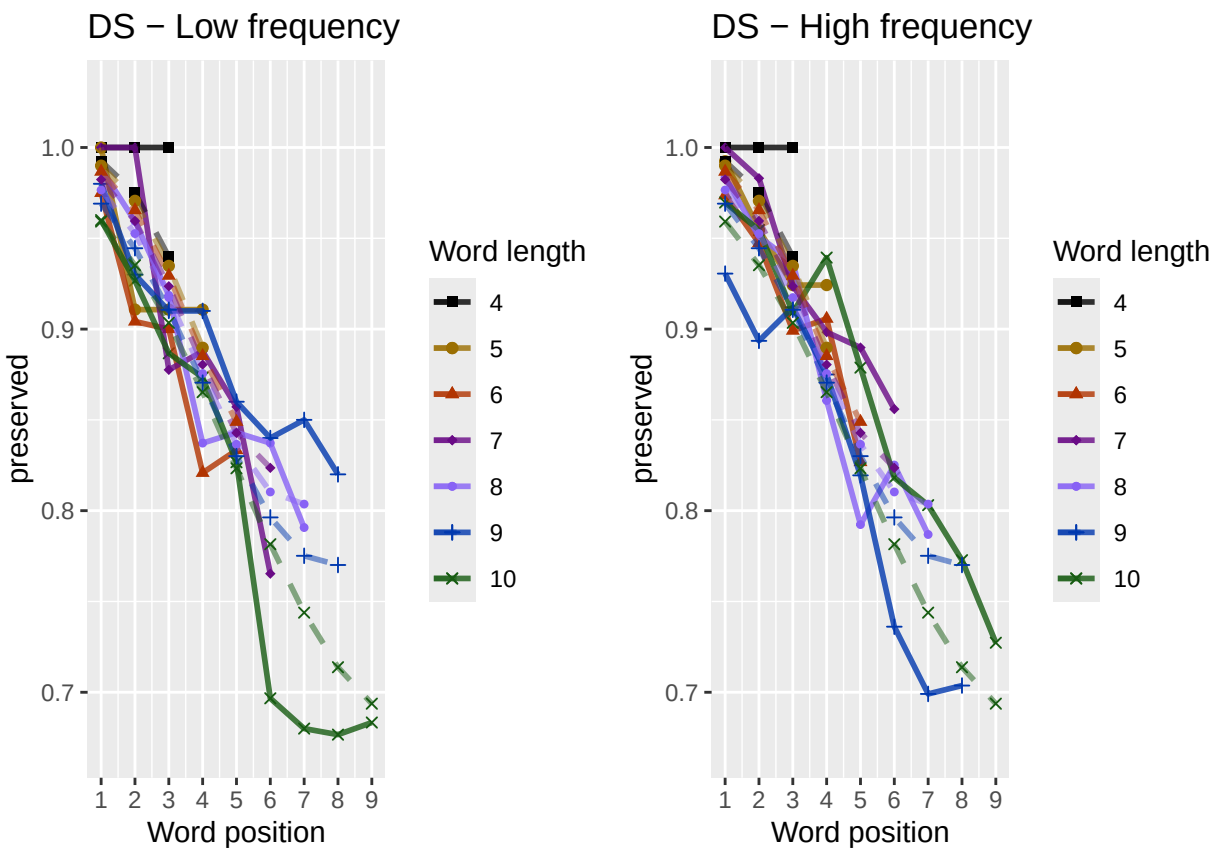
```r
PosDat$FLPFitted<-fitted(BestFLPModel)

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

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

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

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

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

```r
library(ggpubr)
Both_Plots <- ggarrange(LF_Plot,HF_Plot) # labels=c("LF","HF",ncol=2)
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
##       2.848       -1.860
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 1923  AIC: 2108
## log likelihood:  -961.3428
## Nagelkerke R2:  0.4129163
## % pres/err predicted correctly:  -525.0216
## % of predictable range [ (model-null)/(1-null) ]:  0.3800112
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.46340      0.04439     -0.78185
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2639  AIC: 2921
## log likelihood:  -1319.701
## Nagelkerke R2:  0.1166434
## % pres/err predicted correctly:  -796.0331
## % of predictable range [ (model-null)/(1-null) ]:  0.06058684
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.5615      -0.3408
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2657  AIC: 2940
## log likelihood:  -1328.65
## Nagelkerke R2:  0.1085872
## % pres/err predicted correctly:  -796.4665
## % of predictable range [ (model-null)/(1-null) ]:  0.06007599
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.0676      -0.2544
##
```

```
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2819  AIC: 3107
## log likelihood:  -1409.552
## Nagelkerke R2:  0.03419692
## % pres/err predicted correctly:  -833.3831
## % of predictable range [ (model-null)/(1-null) ]:  0.01656476
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.17006     -0.04431
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2888  AIC: 3172
## log likelihood:  -1443.996
## Nagelkerke R2:  0.001655774
## % pres/err predicted correctly:  -846.9272
## % of predictable range [ (model-null)/(1-null) ]:  0.0006011965
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.052
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4253 Residual
## Null Deviance:       2891
## Residual Deviance: 2891  AIC: 3173
## log likelihood:  -1445.734
## Nagelkerke R2:  2.250906e-16
## % pres/err predicted correctly:  -847.4372
## % 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	2107.594	0.0000	1	1	0.4129163	2.847976	NA	-1.860352	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2920.658	813.0635	0	0	0.1166434	4.463396	NA	NA	0.0443873	-0.7818487	NA
preserved ~ pos	2940.469	832.8746	0	0	0.1085873	3.561465	NA	NA	NA	-0.3408272	NA
preserved ~ stimlen	3107.274	999.6803	0	0	0.0341969	4.067586	NA	NA	NA	NA	-0.2543557
preserved ~ CumPres	3172.387	1064.7926	0	0	0.0016558	2.170057	-0.0443078	NA	NA	NA	NA
preserved ~ 1	3172.644	1065.0497	0	0	0.0000000	2.051969	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	0.8812950	556
O	0.8788341	1967
P	1.0000000	36
S	0.8638211	246
V	0.8988958	1449

```
#  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
##       2.851       -2.036
##
## Degrees of Freedom: 3971 Total (i.e. Null);  3970 Residual
## Null Deviance:       2690
## Residual Deviance: 1769  AIC: 1950
## log likelihood:  -884.4547
## Nagelkerke R2:  0.4206783
## % pres/err predicted correctly:  -483.2881
## % of predictable range [ (model-null)/(1-null) ]:  0.3867853
## ***************************
```

```
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.43236      0.04231     -0.76097
##
## Degrees of Freedom: 3971 Total (i.e. Null);  3969 Residual
## Null Deviance:       2690
## Residual Deviance: 2459  AIC: 2732
## log likelihood:  -1229.334
## Nagelkerke R2:  0.1149831
## % pres/err predicted correctly:  -741.7695
## % of predictable range [ (model-null)/(1-null) ]:  0.05949124
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.5689      -0.3399
##
## Degrees of Freedom: 3971 Total (i.e. Null);  3970 Residual
## Null Deviance:       2690
## Residual Deviance: 2474  AIC: 2748
## log likelihood:  -1236.892
## Nagelkerke R2:  0.1076719
## % pres/err predicted correctly:  -741.8556
## % of predictable range [ (model-null)/(1-null) ]:  0.05938217
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.0151      -0.2474
##
## Degrees of Freedom: 3971 Total (i.e. Null);  3970 Residual
## Null Deviance:       2690
## Residual Deviance: 2626  AIC: 2903
## log likelihood:  -1312.946
## Nagelkerke R2:  0.03252638
## % pres/err predicted correctly:  -776.2152
## % of predictable range [ (model-null)/(1-null) ]:  0.01587538
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.055
##
## Degrees of Freedom: 3971 Total (i.e. Null);  3971 Residual
## Null Deviance:       2690
## Residual Deviance: 2690  AIC: 2961
## log likelihood:  -1344.984
## Nagelkerke R2:  2.256647e-16
## % pres/err predicted correctly:  -788.7529
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.15382     -0.03961
##
## Degrees of Freedom: 3971 Total (i.e. Null);  3970 Residual
## Null Deviance:       2690
## Residual Deviance: 2688  AIC: 2962
## log likelihood:  -1343.801
## Nagelkerke R2:  0.001209715
## % pres/err predicted correctly:  -788.4087
## % of predictable range [ (model-null)/(1-null) ]:  0.0004358314
## ***************************
```

```
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	1949.778	0.0000	1	1	0.4206782	3.850608	NA	-2.036019	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2731.938	782.1595	0	0	0.1149834	1.432364	NA	NA	0.0423072	-0.7609657	NA
preserved ~ pos	2748.380	798.6016	0	0	0.1076719	1.568858	NA	NA	NA	-0.3399367	NA
preserved ~ stimlen	2902.621	952.8424	0	0	0.0325264	4.015126	NA	NA	NA	NA	-0.2474052
preserved ~ 1	2961.080	1011.3023	0	0	0.0000000	2.055450	NA	NA	NA	NA	NA
preserved ~ CumPres	2961.714	1011.9359	0	0	0.0012097	2.153818	-0.0396066	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)
```

```r
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
##       2.791       -2.258
##
## Degrees of Freedom: 3415 Total (i.e. Null);  3414 Residual
## Null Deviance:       2302
## Residual Deviance: 1573  AIC: 1734
## log likelihood:  -786.5056
## Nagelkerke R2:  0.3920847
## % pres/err predicted correctly:  -430.0833
## % of predictable range [ (model-null)/(1-null) ]:  0.3611785
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.36424      0.03987     -0.73179
##
## Degrees of Freedom: 3415 Total (i.e. Null);  3413 Residual
## Null Deviance:       2302
## Residual Deviance: 2100  AIC: 2330
## log likelihood:  -1049.857
## Nagelkerke R2:  0.1174526
## % pres/err predicted correctly:  -632.9051
## % of predictable range [ (model-null)/(1-null) ]:  0.06061714
## ***************************
## model index:  4
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      3.5641      -0.3361
##
## Degrees of Freedom: 3415 Total (i.e. Null);  3414 Residual
## Null Deviance:       2302
## Residual Deviance: 2112  AIC: 2344
## log likelihood:  -1056.16
## Nagelkerke R2:  0.110347
## % pres/err predicted correctly:  -632.9025
## % of predictable range [ (model-null)/(1-null) ]:  0.06062105
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      3.9217      -0.2353
##
## Degrees of Freedom: 3415 Total (i.e. Null);  3414 Residual
## Null Deviance:       2302
## Residual Deviance: 2252  AIC: 2486
## log likelihood:  -1125.918
## Nagelkerke R2:  0.02992771
## % pres/err predicted correctly:  -664.1556
## % of predictable range [ (model-null)/(1-null) ]:  0.0143071
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.24091     -0.08284
##
## Degrees of Freedom: 3415 Total (i.e. Null);  3414 Residual
## Null Deviance:       2302
## Residual Deviance: 2295  AIC: 2528
## log likelihood:  -1147.688
## Nagelkerke R2:  0.004150422
## % pres/err predicted correctly:  -672.6369
## % of predictable range [ (model-null)/(1-null) ]:  0.001738668
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)
##       2.064
##
## Degrees of Freedom: 3415 Total (i.e. Null);  3415 Residual
## Null Deviance:       2302
## Residual Deviance: 2302  AIC: 2532
## log likelihood:  -1151.168
## Nagelkerke R2:  4.528502e-16
## % pres/err predicted correctly:  -673.8102
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

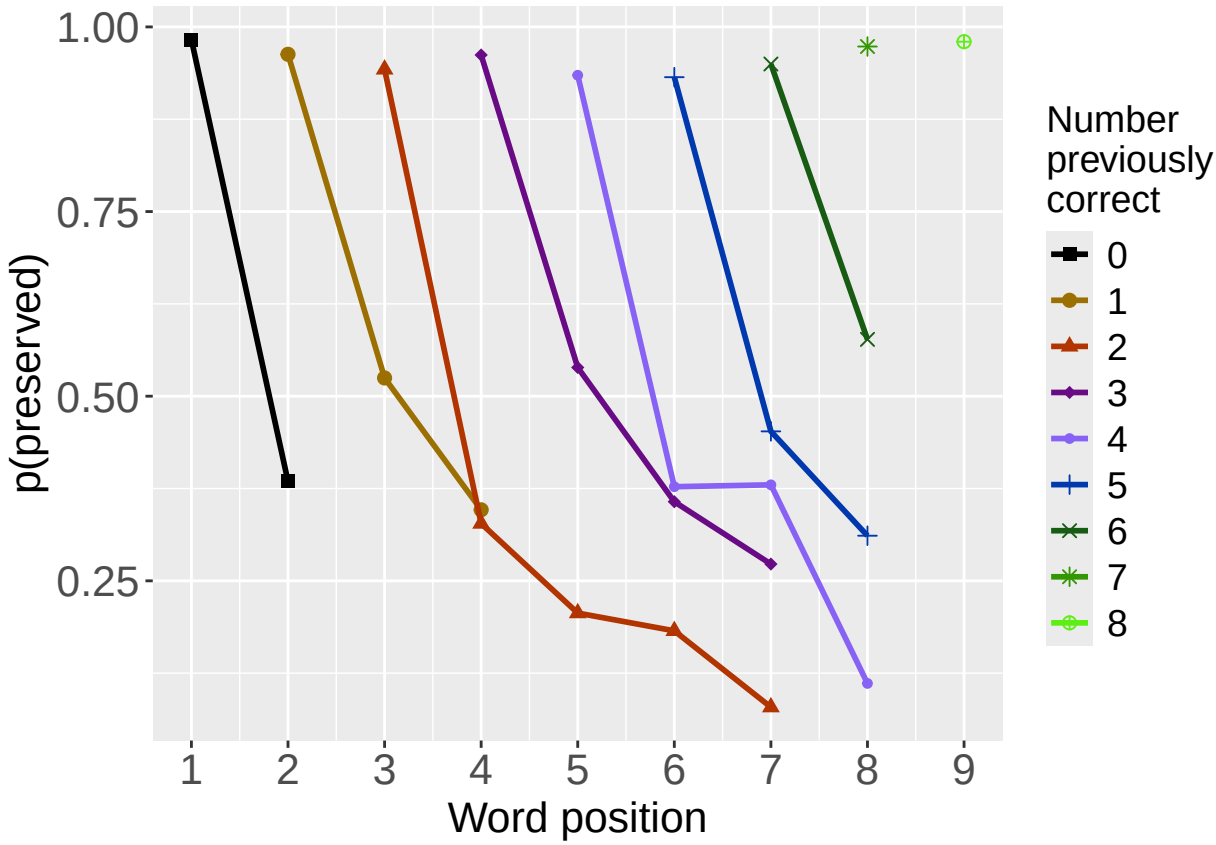
```
write.csv(SimpSyllMEAICSummary2,
          paste0(TablesDir,CurPat,"_",CurTask,
                 "_OV_main_effects_model_summary.csv"),row.names = FALSE)
kable(SimpSyllMEAICSummary2)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1734.367	0.0000	1	1	0.3920842	2.790558	NA	-2.257658	NA	NA	NA
preserved ~ (I(pos^2) + pos)	2330.334	595.9669	0	0	0.1174526	4.364238	NA	NA	0.0398715	-0.7317941	NA
preserved ~ pos	2344.377	610.0097	0	0	0.1103470	3.564081	NA	NA	NA	-0.3360662	NA
preserved ~ stimlen	2486.484	752.1170	0	0	0.0299273	3.921706	NA	NA	NA	NA	-0.2353128
preserved ~ CumPres	2527.616	793.2483	0	0	0.0041502	4.240914	-0.0828428	NA	NA	NA	NA
preserved ~ 1	2531.610	797.2425	0	0	0.0000000	2.063893	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.
```

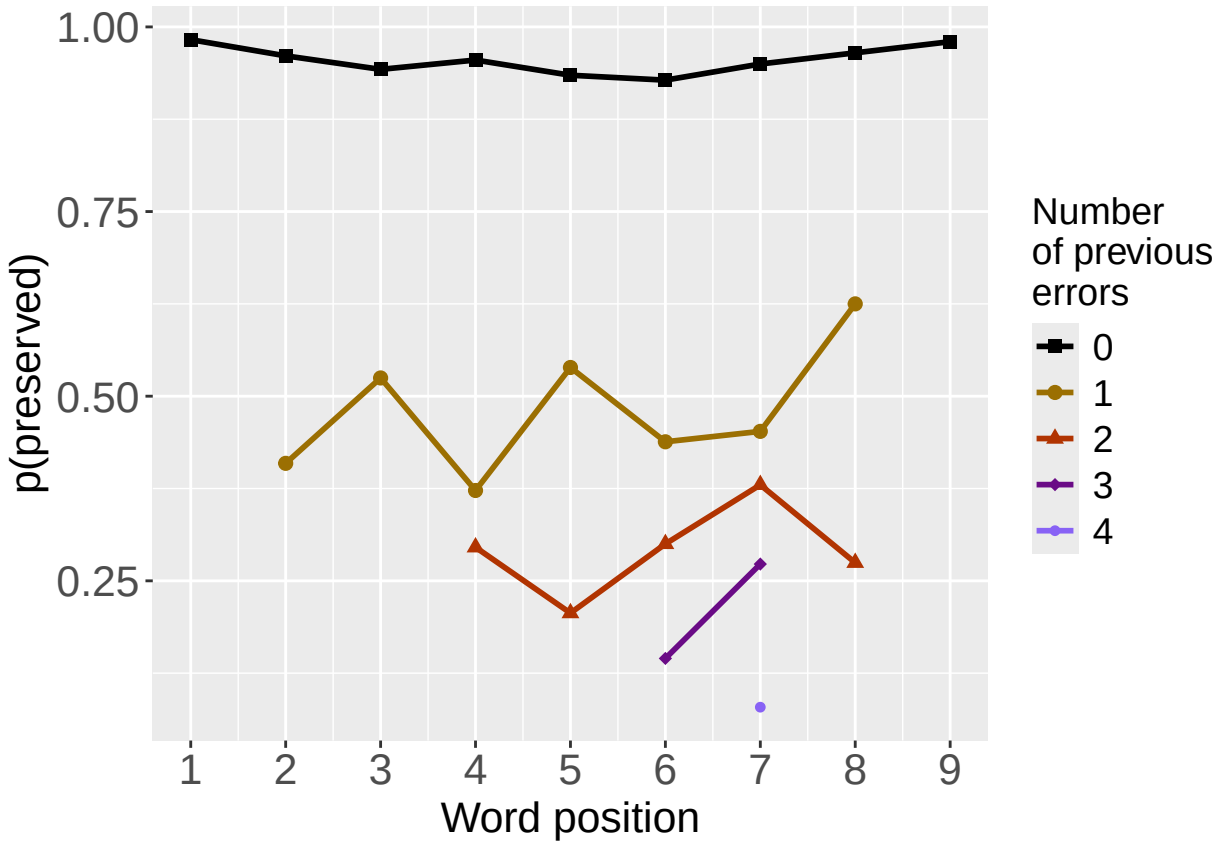
```
print(PrevCorPlot)
```




```
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
```

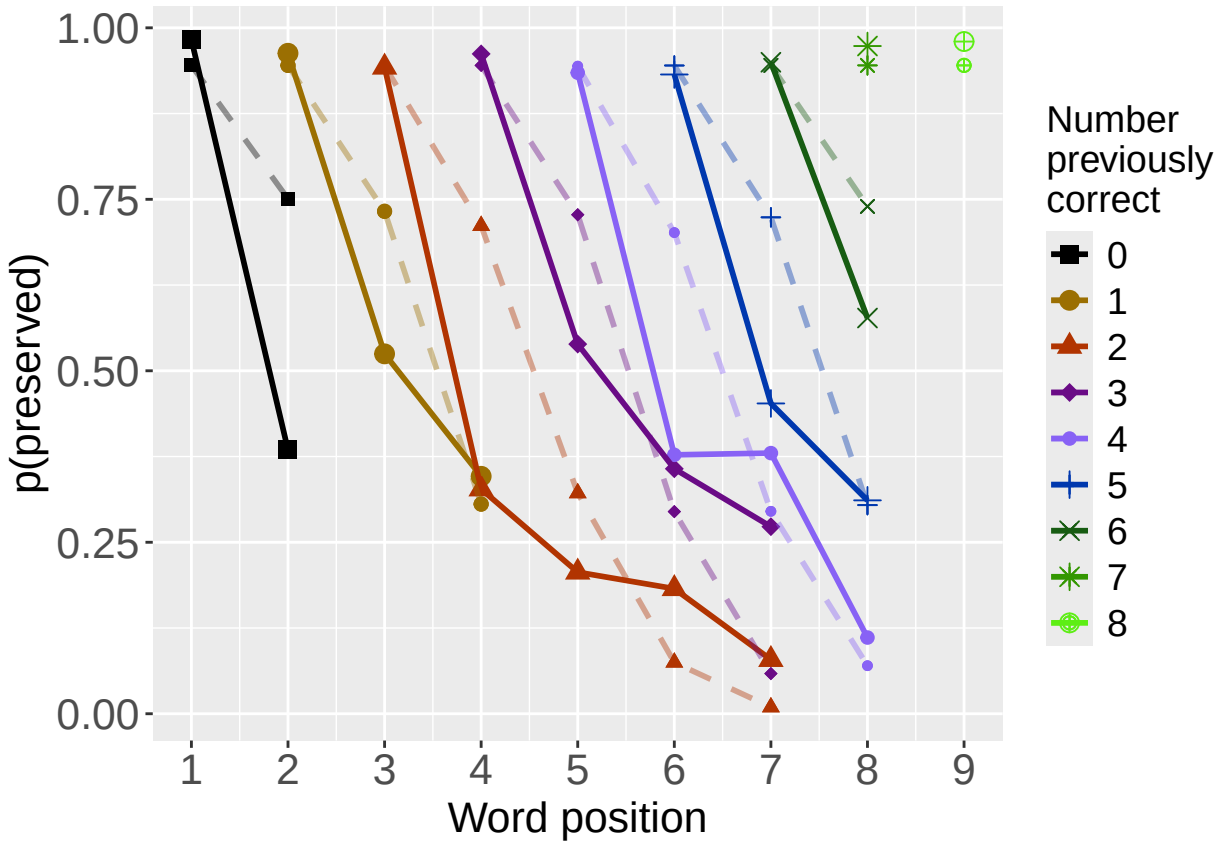
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

```
## 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.
```

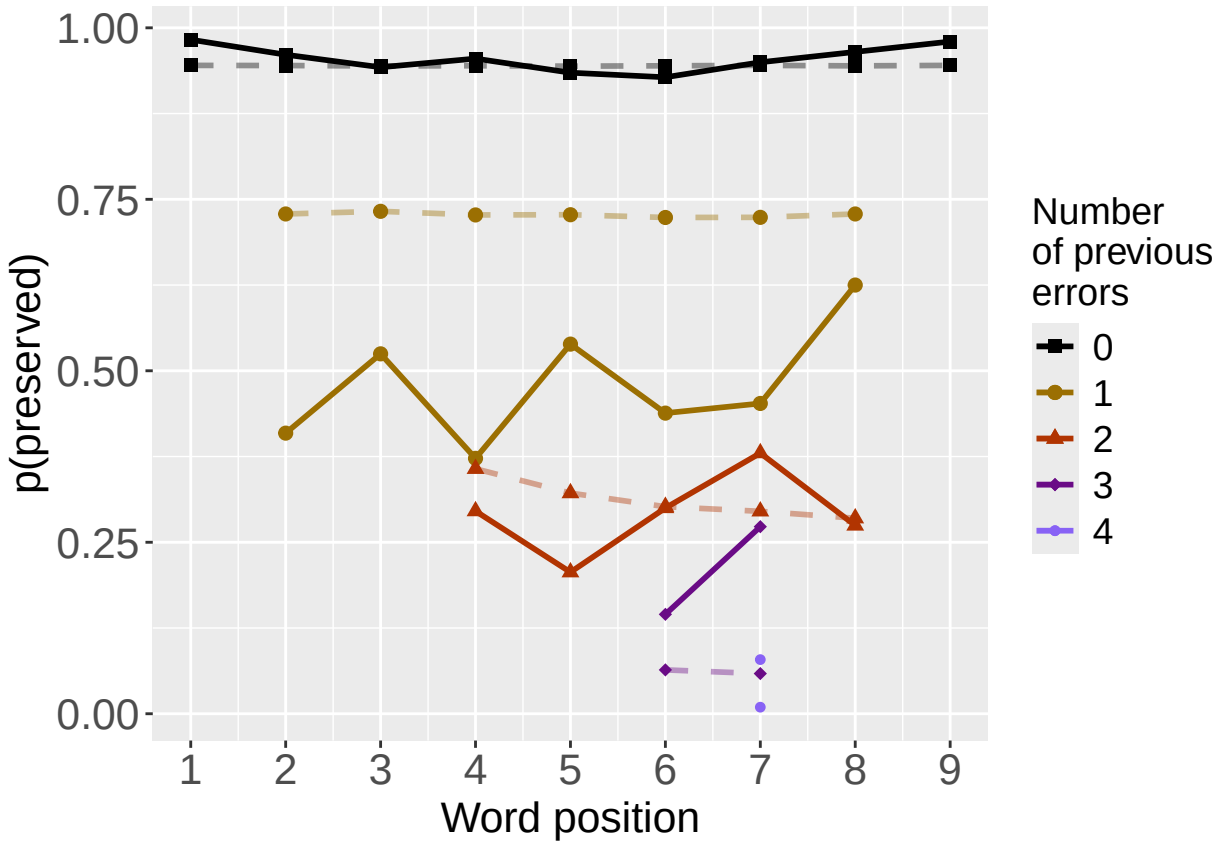
```
print(PrevCorPlot)
```




```
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
```

```
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     4.45155     -1.78144      0.07815     -0.79159
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 1883  AIC: 2068
## log likelihood:  -941.5021
## Nagelkerke R2:  0.4279065
## % pres/err predicted correctly:  -519.265
## % of predictable range [ (model-null)/(1-null) ]:  0.3867961
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.848       -1.860
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 1923  AIC: 2108
## log likelihood:  -961.3428
## Nagelkerke R2:  0.4129163
## % pres/err predicted correctly:  -525.0216
## % of predictable range [ (model-null)/(1-null) ]:  0.3800112
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     4.46340      0.04439     -0.78185
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 2639  AIC: 2921
## log likelihood:  -1319.701
## Nagelkerke R2:  0.1166434
## % pres/err predicted correctly:  -796.0331
## % of predictable range [ (model-null)/(1-null) ]:  0.06058684
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	2067.868	0.0000	1	1	0.4279065	4.451554	-1.781444	0.0781500	-0.7915898

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	2107.594	39.7257	0	0	0.4129163	2.847976	-1.860352	NA	NA
preserved ~ I(posˆ2) + pos	2920.658	852.7892	0	0	0.1166434	4.463396	NA	0.0443873	-0.7818487

```
########
# 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
##       2.848       -1.860
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 1923  AIC: 2108
## log likelihood:  -961.3428
## Nagelkerke R2:  0.4129163
## % pres/err predicted correctly:  -525.0216
## % of predictable range [ (model-null)/(1-null) ]:  0.3800112
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     3.42363     -1.83179     -0.07508
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 1919  AIC: 2108
## log likelihood:  -959.3005
## Nagelkerke R2:  0.4144658
## % pres/err predicted correctly:  -524.8389
## % of predictable range [ (model-null)/(1-null) ]:  0.3802265
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      4.0676      -0.2544
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2819  AIC: 3107
## log likelihood:  -1409.552
## Nagelkerke R2:  0.03419692
## % pres/err predicted correctly:  -833.3831
## % of predictable range [ (model-null)/(1-null) ]:  0.01656476
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr	2107.594	0.0000000	1.0000000	0.5457628	0.4129163	2.847976	-1.860352	NA
preserved ~ CumErr + stimlen	2107.961	0.3671298	0.8322979	0.4542372	0.4144658	3.423633	-1.831787	-0.0750761
preserved ~ stimlen	3107.274	999.6802895	0.0000000	0.0000000	0.0341969	4.067586	NA	-0.2543557

```
########
# 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
##     3.06553     -1.85613     -0.07879
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
```

```
## Null Deviance:       2891
## Residual Deviance: 1916  AIC: 2106
## log likelihood:  -958.0286
## Nagelkerke R2:  0.41543
## % pres/err predicted correctly:  -525.0522
## % of predictable range [ (model-null)/(1-null) ]:  0.3799752
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.848       -1.860
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 1923  AIC: 2108
## log likelihood:  -961.3428
## Nagelkerke R2:  0.4129163
## % pres/err predicted correctly:  -525.0216
## % of predictable range [ (model-null)/(1-null) ]:  0.3800112
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##     2.17006     -0.04431
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2888  AIC: 3172
## log likelihood:  -1443.996
## Nagelkerke R2:  0.001655774
## % pres/err predicted correctly:  -846.9272
## % of predictable range [ (model-null)/(1-null) ]:  0.0006011965
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	2106.061	0.000000	1.0000000	0.6827605	0.4154300	3.065533	-1.856127	-0.0787938
preserved ~ CumErr	2107.594	1.532974	0.4646425	0.3172395	0.4129163	2.847976	-1.860352	NA
preserved ~ CumPres	3172.387	1066.3255520.0000000		0.0000000	0.0016558	2.170057	NA	-0.0443078

```
########
# 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
##     3.14433     -1.77733     -0.07879
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 1916  AIC: 2106
## log likelihood:  -958.0286
## Nagelkerke R2:  0.41543
## % pres/err predicted correctly:  -525.0522
## % of predictable range [ (model-null)/(1-null) ]:  0.3799752
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.848       -1.860
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 1923  AIC: 2108
## log likelihood:  -961.3428
## Nagelkerke R2:  0.4129163
## % pres/err predicted correctly:  -525.0216
## % of predictable range [ (model-null)/(1-null) ]:  0.3800112
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      3.5615      -0.3408
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 2657  AIC: 2940
## log likelihood:  -1328.65
## Nagelkerke R2:  0.1085872
## % pres/err predicted correctly:  -796.4665
## % of predictable range [ (model-null)/(1-null) ]:  0.06007599
## **************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	2106.061	0.000000	1.0000000	0.6827605	0.4154300	3.144327	-1.777333	-0.0787938
preserved ~ CumErr	2107.594	1.532974	0.4646425	0.3172395	0.4129163	2.847976	-1.860352	NA
preserved ~ pos	2940.469	834.407533	0.0000000	0.0000000	0.1085872	3.561465	NA	-0.3408272

```
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	2067.868	0.0000000	1.0000000	1.0000000	0.4279065	3.5451554	-1.781444	0.0781500	-0.7915898	NA	NA
preserved ~ CumErr + CumPres	2106.061	0.0000000	1.0000000	0.6827605	0.4154300	3.0065533	-1.856127	NA	NA	NA	-0.0787938
preserved ~ CumErr + pos	2106.061	0.0000000	1.0000000	0.6827605	0.4154300	3.0144327	-1.777333	NA	-0.0787938	NA	NA
preserved ~ CumErr	2107.594	19.7256994	4.0000000	0.0000000	0.4129163	2.3847976	-1.860352	NA	NA	NA	NA
preserved ~ CumErr	2107.594	0.0000000	1.0000000	0.5457628	0.4129163	2.3847976	-1.860352	NA	NA	NA	NA
preserved ~ CumErr	2107.594	1.5329739	0.4646425	0.3172395	0.4129163	2.3847976	-1.860352	NA	NA	NA	NA
preserved ~ CumErr	2107.594	1.5329739	0.4646425	0.3172395	0.4129163	2.3847976	-1.860352	NA	NA	NA	NA
preserved ~ CumErr + stimlen	2107.960	1.3671298	0.8322970	0.7454237	0.4241446	2.8423633	-1.831787	NA	NA	-0.0750761	NA
preserved ~ I(pos^2) + pos	2920.658	852.7892468	0.0000000	0.0000000	0.1166434	3.4463396	NA	0.0443873	-0.7818487	NA	NA
preserved ~ pos	2940.469	834.4075328	0.0000000	0.0000000	0.1085872	3.2561465	NA	NA	-0.3408272	NA	NA
preserved ~ stimlen	3107.279	999.6802895	0.0000000	0.0000000	0.0034196	4.0067586	NA	NA	NA	-0.2543557	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	stimlen	CumPres
preserved ~ CumPres	3172.387	1066.32555	20000000	0.0000000	0.0016552	2.8170057	NA	NA	NA	NA	-0.0443078

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     5.10296     -1.77889      0.08305     -0.80689     -0.08869
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4249 Residual
## Null Deviance:       2891
## Residual Deviance: 1879  AIC: 2067
## log likelihood:  -939.2668
## Nagelkerke R2:  0.4295866
## % pres/err predicted correctly:  -518.3625
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.3878599
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     5.02320     -1.77780      0.08282     -0.80484     -0.07855      0.03335
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4248 Residual
## Null Deviance:       2891
## Residual Deviance: 1878  AIC: 2068
## log likelihood:  -938.7826
## Nagelkerke R2:  0.4299503
## % pres/err predicted correctly:  -518.3229
## % of predictable range [ (model-null)/(1-null) ]:  0.3879065
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     4.45155     -1.78144      0.07815     -0.79159
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 1883  AIC: 2068
## log likelihood:  -941.5021
## Nagelkerke R2:  0.4279065
## % pres/err predicted correctly:  -519.265
## % of predictable range [ (model-null)/(1-null) ]:  0.3867961
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     4.44261     -1.77949      0.07862     -0.79085      0.04822
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4249 Residual
## Null Deviance:       2891
## Residual Deviance: 1881  AIC: 2068
## log likelihood:  -940.4285
## Nagelkerke R2:  0.4287137
## % pres/err predicted correctly:  -519.0296
## % of predictable range [ (model-null)/(1-null) ]:  0.3870736
## ***************************
## model index:  2
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       2.052
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4253 Residual
## Null Deviance:       2891
## Residual Deviance: 2891  AIC: 3173
## log likelihood:  -1445.734
## Nagelkerke R2:  2.250906e-16
## % pres/err predicted correctly:  -847.4372
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen	2066.691	0.000000	1.0000000	0.3834309	0.0429585	6.102959	-1.778890	0.0830513	-0.8068896	NA	-0.0886931
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	2067.851	1.160176	0.5598400	0.2146685	0.0429955	3.023196	-1.777800	0.0828177	-0.8048371	0.0333520	-0.0785475
preserved ~ CumErr + I(pos^2) + pos	2067.861	1.177072	0.5551394	0.2128626	0.0427906	4.451554	-1.781444	0.0781500	-0.7915898	NA	NA
preserved ~ CumErr + I(pos^2) + pos + log_freq	2068.101	1.414565	0.4929821	0.1890290	0.0428713	7.442609	-1.779492	0.0786177	-0.7908538	0.0482196	NA
preserved ~ 1	3172.644	1105.952470	0.0000000	0.0000000	0.0000000	2.051969	NA	NA	NA	NA	NA

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

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

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

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

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

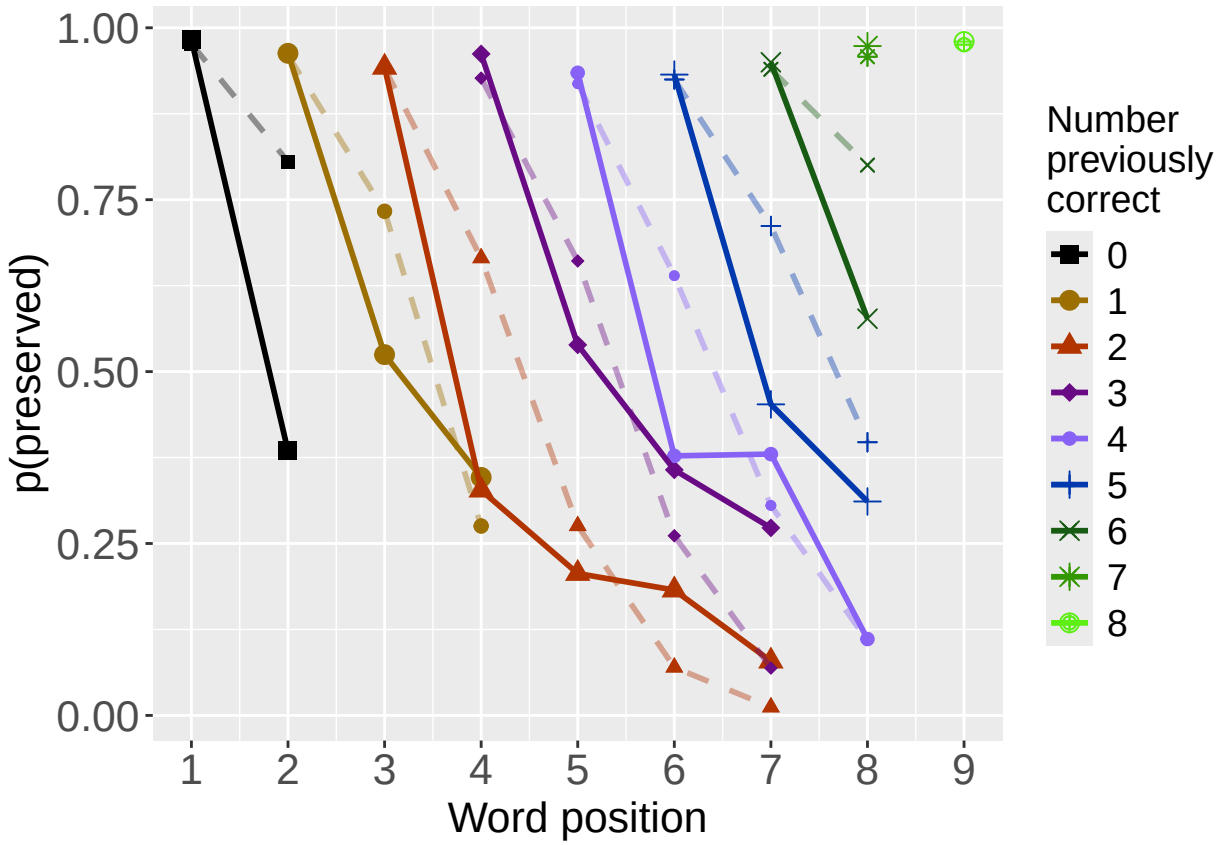
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen
##          Df Deviance    AIC
## CumErr    1   2632.7 2818.9
## pos       1   1918.5 2104.7
## I(pos^2)  1   1914.9 2101.0
## stimlen   1   1883.0 2069.2
## <none>        1878.5 2066.7
```

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

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

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

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

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

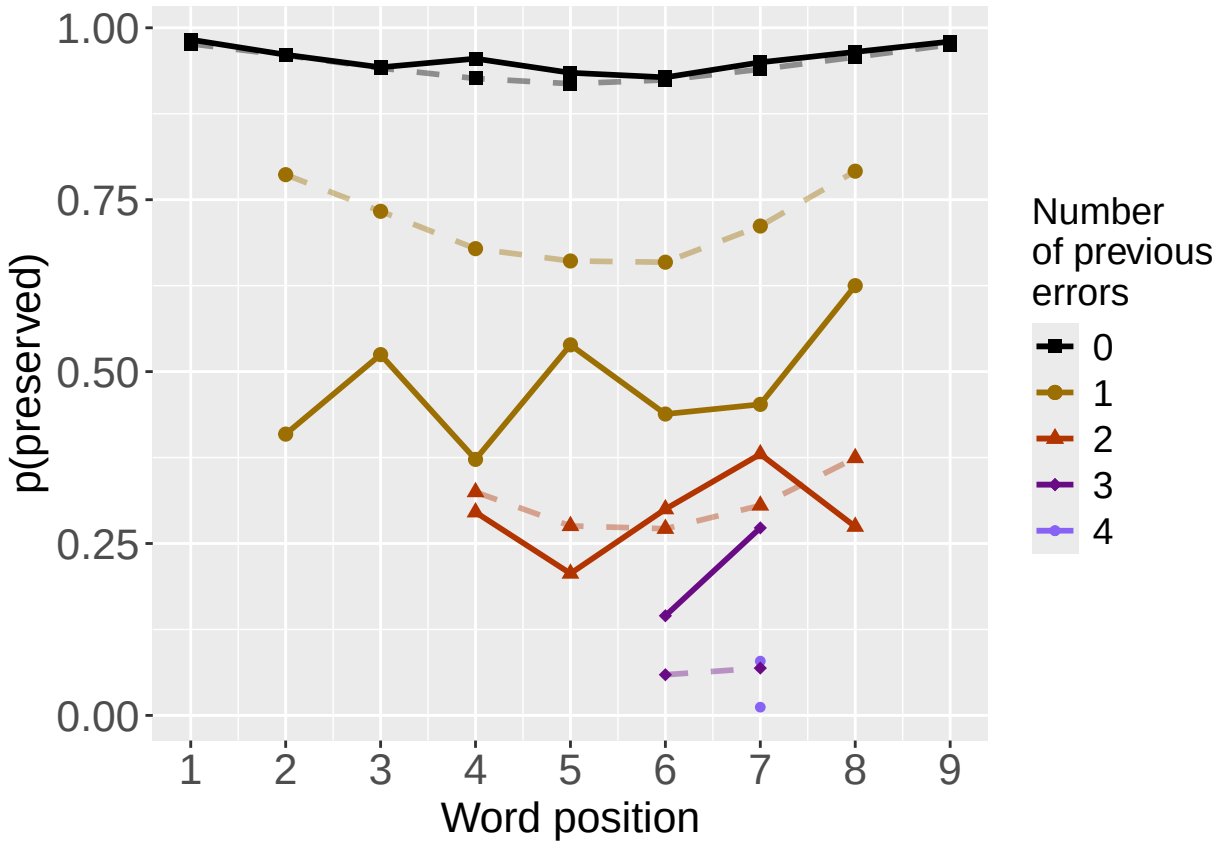
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

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

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

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

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

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

FinalModelSet<-ModelSetFromDeletions(BestModelFormulaL3,
                                     BestModelDeletions,
                                     last_model_num)

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

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

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

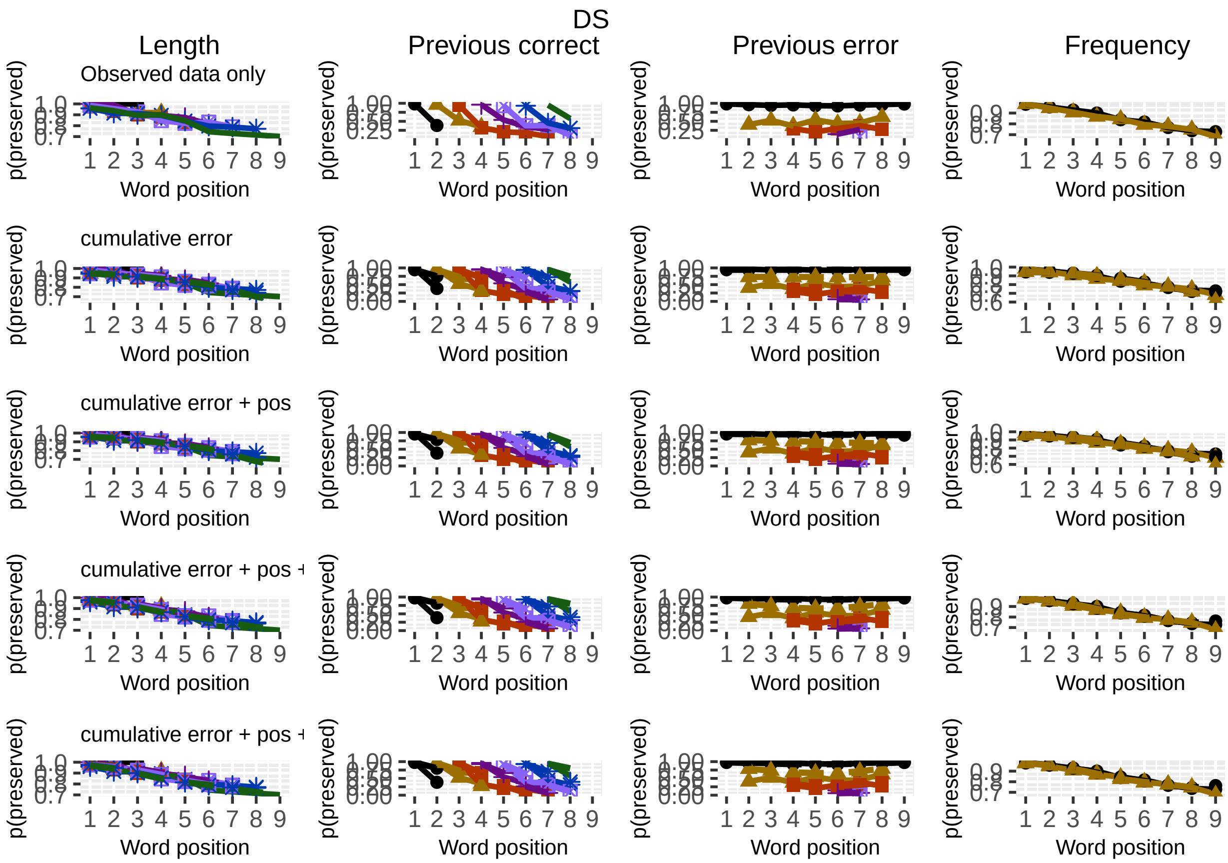
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##       2.848       -1.860
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4252 Residual
## Null Deviance:       2891
## Residual Deviance: 1923  AIC: 2108
## log likelihood:  -961.3428
```

```
## Nagelkerke R2:  0.4129163
## % pres/err predicted correctly:  -525.0216
## % of predictable range [ (model-null)/(1-null) ]:  0.3800112
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##     3.14433     -1.77733     -0.07879
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4251 Residual
## Null Deviance:       2891
## Residual Deviance: 1916  AIC: 2106
## log likelihood:  -958.0286
## Nagelkerke R2:  0.41543
## % pres/err predicted correctly:  -525.0522
## % of predictable range [ (model-null)/(1-null) ]:  0.3799752
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)
##     4.45155     -1.78144     -0.79159      0.07815
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4250 Residual
## Null Deviance:       2891
## Residual Deviance: 1883  AIC: 2068
## log likelihood:  -941.5021
## Nagelkerke R2:  0.4279065
## % pres/err predicted correctly:  -519.265
## % of predictable range [ (model-null)/(1-null) ]:  0.3867961
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)      stimlen
##     5.10296     -1.77889     -0.80689      0.08305     -0.08869
##
## Degrees of Freedom: 4253 Total (i.e. Null);  4249 Residual
## Null Deviance:       2891
## Residual Deviance: 1879  AIC: 2067
## log likelihood:  -939.2668
## Nagelkerke R2:  0.4295866
## % pres/err predicted correctly:  -518.3625
## % of predictable range [ (model-null)/(1-null) ]:  0.3878599
```

```
## ***************************

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

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

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

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

## Warning: The shape palette can deal with a maximum of 6 discrete values because more than 6 becomes
## difficult to discriminate
## i you have requested 9 values. Consider specifying shapes manually if you need that many have
##   them.

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

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




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

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

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

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

	CumErr	I(pos^2)	pos	stimlen
McFadden	0.2899606	0.0242090	0.0311935	0.0059701
SquaredCorrelation	0.1873134	0.0169155	0.0218058	0.0043466
Nagelkerke	0.1873134	0.0169155	0.0218058	0.0043466
Estrella	0.2288049	0.0185551	0.0238864	0.0044950

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

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

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

```
##                                                          model deviance
## CumErr + pos + I(pos^2) + stimlen CumErr + pos + I(pos^2) + stimlen 1878.534
## CumErr + pos + I(pos^2)                    CumErr + pos + I(pos^2) 1883.004
## CumErr + pos                                         CumErr + pos 1916.057
## CumErr                                                     CumErr 1922.686
## null                                                         null 2891.468
##                                  deviance_explained percent_explained
## CumErr + pos + I(pos^2) + stimlen          1012.9344          35.03184
## CumErr + pos + I(pos^2)                    1008.4637          34.87722
## CumErr + pos                                975.4108          33.73410
## CumErr                                      968.7823          33.50486
## null                                          0.0000           0.00000
##                                  percent_of_explained_deviance increment_in_explained
## CumErr + pos + I(pos^2) + stimlen                     100.00000              0.4413548
## CumErr + pos + I(pos^2)                                99.55865              3.2630870
## CumErr + pos                                           96.29556              0.6543847
## CumErr                                                 95.64117             95.6411736
## null                                                         NA              0.0000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + I(pos^2) + stimlen	1878.534	1012.9344
CumErr + pos + I(pos^2)	1883.004	1008.4637
CumErr + pos	1916.057	975.4108
CumErr	1922.686	968.7823
null	2891.468	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + I(pos^2) + stimlen	35.03184	100.00000	0.4413548
CumErr + pos + I(pos^2)	34.87722	99.55865	3.2630870
CumErr + pos	33.73410	96.29556	0.6543847
CumErr	33.50486	95.64117	95.6411736
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   0.81305831
## I(pos^2) 0.07342401
## pos      0.09465073
## stimlen  0.01886695
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr+pos	0.9387917	1916.057
preserved ~ CumErr	0.9396413	1922.686
preserved ~ CumErr+pos+I(pos^2)	0.9494054	1883.004
preserved ~ CumErr+pos+I(pos^2)+stimlen	0.9515889	1878.534

```
## 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+pos
## 1                 preserved ~ CumErr+pos       0.9387917       1916.057    0.0000000000
## 2                     preserved ~ CumErr       0.9396413       1922.686    0.0008495804
## 3         preserved ~ CumErr+pos+I(pos^2)       0.9494054       1883.004    0.0106136246
## 4 preserved ~ CumErr+pos+I(pos^2)+stimlen       0.9515889       1878.534    0.0127971226
##     diff_CumErr diff_CumErr+pos+I(pos^2) diff_CumErr+pos+I(pos^2)+stimlen
## 1 -0.0008495804             -0.010613625                     -0.012797123
## 2  0.0000000000             -0.009764044                     -0.011947542
## 3  0.0097640442              0.000000000                     -0.002183498
## 4  0.0119475422              0.002183498                      0.000000000
```

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

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

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

model	diff_CumErr+pos	diff_CumErr	diff_CumErr+pos+I(pos^2)
preserved ~ CumErr+pos	0.0000000	-0.0008496	-0.0106136
preserved ~ CumErr	0.0008496	0.0000000	-0.0097640
preserved ~ CumErr+pos+I(pos^2)	0.0106136	0.0097640	0.0000000
preserved ~ CumErr+pos+I(pos^2)+stimlen	0.0127971	0.0119475	0.0021835