VS - naming - Serial position analysis (v5)

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

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

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

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

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

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

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

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

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

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

kable(syll_comp_dist_N)
```

pos_factor	O	P	V	1	S	total
1	185	14	23	NA	NA	222
2	25	NA	153	12	32	222
3	81	NA	54	83	4	222
4	118	NA	52	18	10	198
5	58	NA	53	12	9	132
6	45	NA	14	19	5	83
7	31	NA	11	2	2	46
8	10	NA	2	2	NA	14
9	4	NA	1	NA	2	7

```
kable(syll_comp_dist_perc)
```

pos_factor	O	P	V	1	S	total
1	0.8333333	0.0630631	0.1036036	NA	NA	222
2	0.1126126	NA	0.6891892	0.0540541	0.1441441	222
3	0.3648649	NA	0.2432432	0.3738739	0.0180180	222
4	0.5959596	NA	0.2626263	0.0909091	0.0505051	198
5	0.4393939	NA	0.4015152	0.0909091	0.0681818	132
6	0.5421687	NA	0.1686747	0.2289157	0.0602410	83

pos_factor	O	P	V	1	S	total
7	0.6739130	NA	0.2391304	0.0434783	0.0434783	46
8	0.7142857	NA	0.1428571	0.1428571	NA	14
9	0.5714286	NA	0.1428571	NA	0.2857143	7

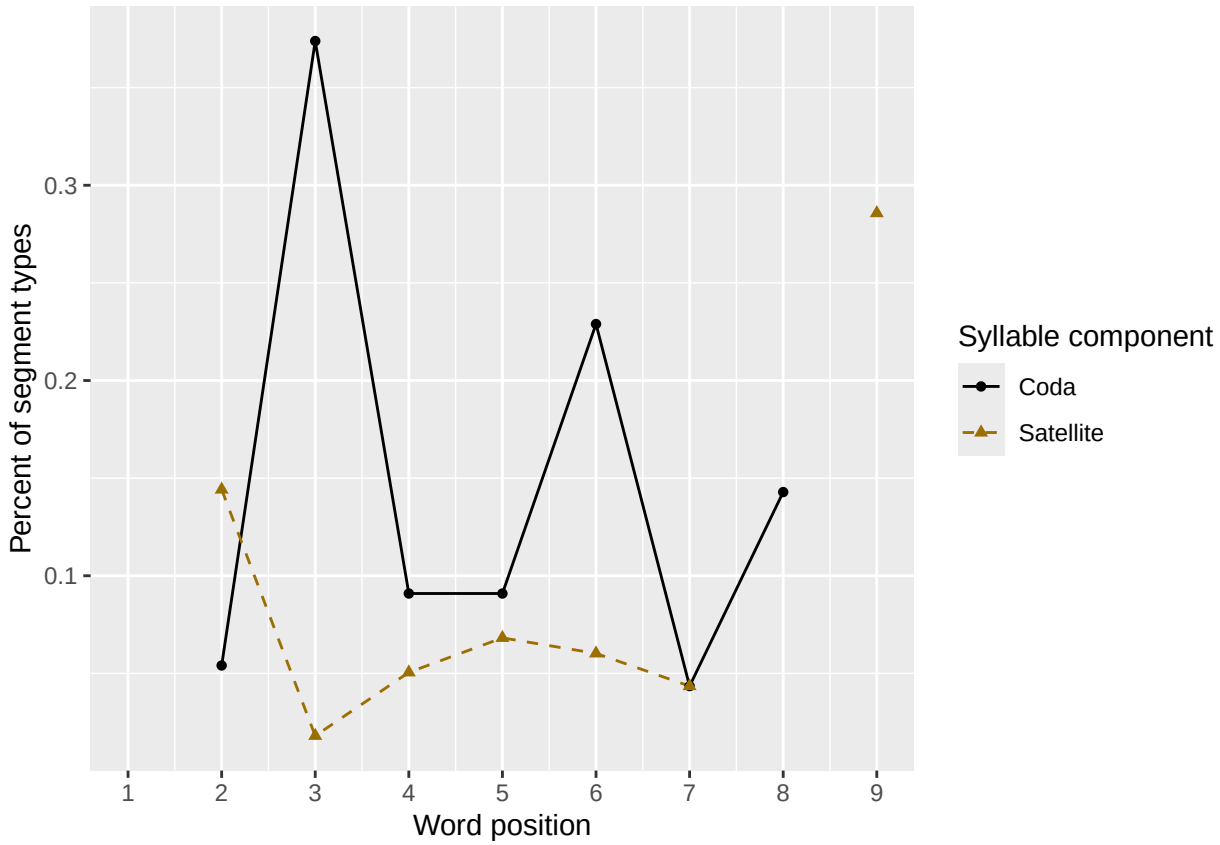
```
# get percentages only from summary table
syll_comp_perc <- syll_comp_dist_perc[,2:(ncol(syll_comp_dist_perc)-1)]
syll_comp_perc$pos <- as.integer(as.character(syll_comp_dist_perc$pos_factor))
syll_comp_perc <- syll_comp_perc %>% rename("Coda" ="1","Satellite"="S")

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




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

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

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

```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.868 0.889 0.743 NA     NA     NA     NA     NA     NA
## 2       5 0.919 0.816 0.75   0.780 NA     NA     NA     NA     NA
## 3       6 0.959 0.820 0.796  0.748  0.707 NA     NA     NA     NA
## 4       7 0.973 0.757 0.793  0.694  0.635  0.608 NA     NA     NA
## 5       8 0.930 0.805 0.695  0.617  0.562  0.573  0.552 NA     NA
## 6       9 1     0.714 0.690  0.214  0.381  0.786  0.643  0.571 NA
## 7      10 1     0.762 0.714  0.833  0.714  0.595  0.286  0.5    0.548
```

```
# 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.
```

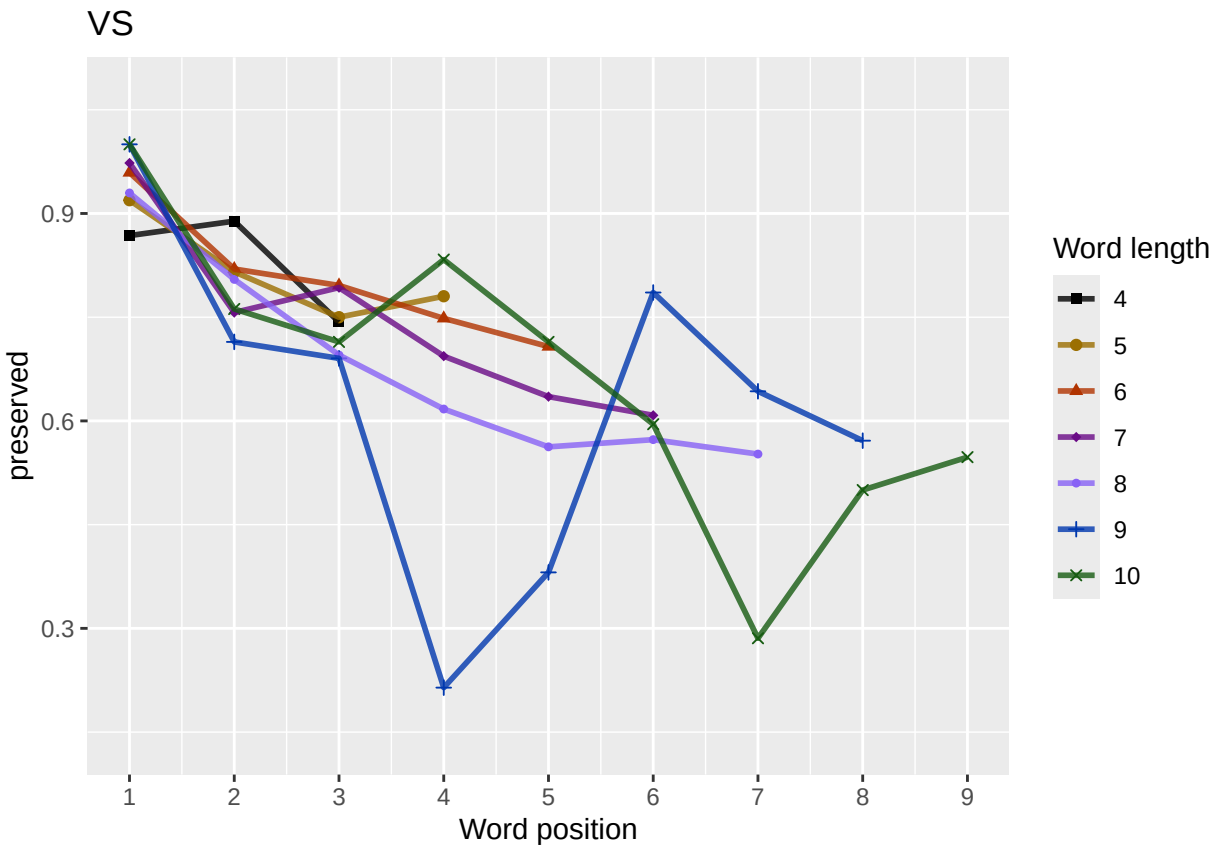
```
pos_len_N_table <- pos_len_N %>% pivot_wider(names_from = pos, values_from = N)
write.csv(pos_len_N_table,paste0(TablesDir,CurPat,"_",CurTask,"_N_by_len_position.csv"))
pos_len_N_table
```

```
## # 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    24    24    24    NA    NA    NA    NA    NA    NA
## 2       5    66    66    66    66    NA    NA    NA    NA    NA
## 3       6    49    49    49    49    49    NA    NA    NA    NA
## 4       7    37    37    37    37    37    37    NA    NA    NA
## 5       8    32    32    32    32    32    32    32    NA    NA
## 6       9     7     7     7     7     7     7     7     7    NA
## 7      10     7     7     7     7     7     7     7     7     7
```

```
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)
##        2.4580412         0.1324494         0.0961748        -0.5561625         0.0001089
##      stimlen:pos
##       -0.0704543
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1105  AIC: 1226
## log likelihood:  -552.3462
## Nagelkerke R2:  0.1202509
## % pres/err predicted correctly:  -374.0831
## % of predictable range [ (model-null)/(1-null) ]:  0.08566906
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.65275     -0.09606      0.06080     -0.79704
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1108  AIC: 1228
## log likelihood:  -554.1995
## Nagelkerke R2:  0.1156437
## % pres/err predicted correctly:  -375.6505
## % of predictable range [ (model-null)/(1-null) ]:  0.0818483
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.03432      0.05256     -0.76748
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1112  AIC: 1231
## log likelihood:  -555.7954
## Nagelkerke R2:  0.1116646
## % pres/err predicted correctly:  -377.7723
## % of predictable range [ (model-null)/(1-null) ]:  0.07667614
## ***************************
## model index:  4
##
```

```
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.57603     -0.05162     -0.30950
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1119  AIC: 1237
## log likelihood:  -559.6582
## Nagelkerke R2:  0.1019872
## % pres/err predicted correctly:  -378.9161
## % of predictable range [ (model-null)/(1-null) ]:  0.07388783
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3047      -0.3303
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1120  AIC: 1237
## log likelihood:  -560.1503
## Nagelkerke R2:  0.1007497
## % pres/err predicted correctly:  -379.715
## % of predictable range [ (model-null)/(1-null) ]:  0.07194047
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.17876     -0.13451     -0.48829      0.02356
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1118  AIC: 1239
## log likelihood:  -559.2009
## Nagelkerke R2:  0.1031363
## % pres/err predicted correctly:  -378.7755
## % of predictable range [ (model-null)/(1-null) ]:  0.07423062
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen
##      2.5773      -0.2152
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1174  AIC: 1290
## log likelihood:  -586.886
## Nagelkerke R2:  0.03189073
## % pres/err predicted correctly:  -398.7117
## % of predictable range [ (model-null)/(1-null) ]:  0.02563249
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.13
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1145 Residual
## Null Deviance:       1198
## Residual Deviance: 1198  AIC: 1317
## log likelihood:  -598.8578
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -409.2269
## % 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)	1225.724	0.000001	1.000000	0.7237078	0.1202529	2.945804	0.1324494	-0.5561625	-5.070454	0.0961748	0.0001089
preserved ~ stimlen + I(pos^2) + pos	1228.083	2.359991	0.307292	0.2222390	0.1156437	3.652745	-0.0960609	-1.797043	NA	0.0607962	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ I(pos^2) + pos	1231.175	8.454560	0.0653969	0.0047328	0.3111664	2.6034318	NA	-0.7674843	NA	0.0525631	NA
preserved ~ stimlen + pos	1236.795	1.068858	0.0039485	0.0002857	0.1019872	2.2576029	-0.0516204	-0.3095010	NA	NA	NA
preserved ~ pos	1236.890	1.165688	0.0037610	0.0002722	0.1007492	1.7304664	NA	-0.3302605	NA	NA	NA
preserved ~ stimlen * pos	1238.905	3.181000	0.0013784	0.0000998	0.0910313	2.3178757	-0.1345149	-0.4882851	0.0235621	NA	NA
preserved ~ stimlen	1289.536	53.811660	0.0000000	0.0000000	0.0031892	1.7577300	-0.2152113	NA	NA	NA	NA
preserved ~ 1	1317.068	81.344570	0.0000000	0.0000000	0.0000000	1.0129883	NA	NA	NA	NA	NA

```
print(BestLPModelFormula)
```

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

```
print(BestLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.4580412         0.1324494         0.0961748        -0.5561625         0.0001089
##      stimlen:pos
##       -0.0704543
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1105  AIC: 1226
```

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

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

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

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

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

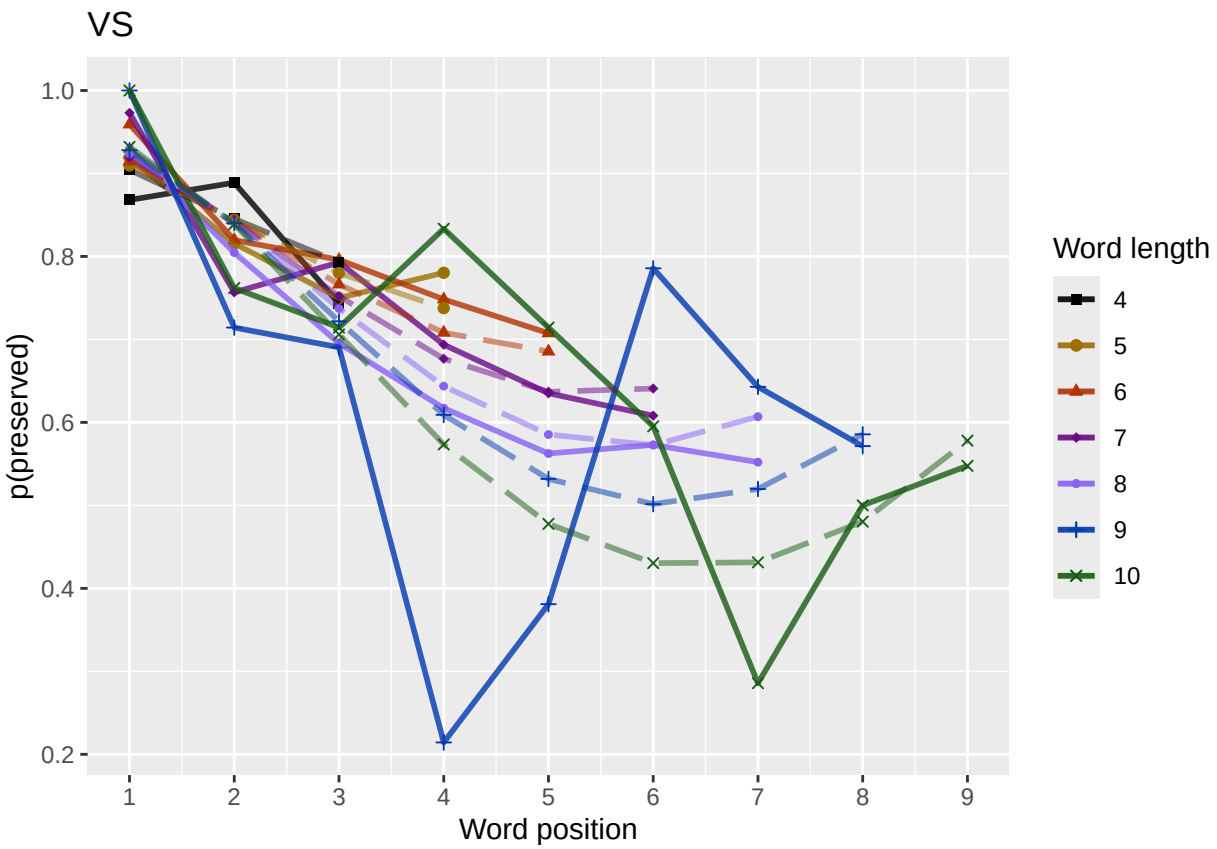
```
## # A tibble: 7 x 10
## # Groups:   stimlen [7]
##   stimlen   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       4 0.904 0.845 0.793 NA     NA     NA     NA     NA     NA
## 2       5 0.910 0.844 0.780  0.738 NA     NA     NA     NA     NA
## 3       6 0.915 0.843 0.766  0.708  0.685 NA     NA     NA     NA
## 4       7 0.919 0.842 0.752  0.677  0.637  0.641 NA     NA     NA
```

```
## 5       8 0.924 0.841 0.737  0.644  0.585  0.573  0.607 NA     NA
## 6       9 0.928 0.840 0.722  0.609  0.532  0.502  0.520  0.586 NA
## 7      10 0.932 0.839 0.706  0.573  0.478  0.430  0.431  0.480  0.578
```

```
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       24   222
```

```
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 24 / 222 = 10.81 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)
##        2.3140081         0.1622674         0.1158504        -0.4801846         0.0007881
##      stimlen:pos
##       -0.0925862
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1090 Residual
## Null Deviance:       1054
## Residual Deviance: 995.2     AIC: 1121
## log likelihood:  -497.612
## Nagelkerke R2:  0.08486512
## % pres/err predicted correctly:  -331.6394
## % of predictable range [ (model-null)/(1-null) ]:  0.0595554
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.85431     -0.12356      0.07884     -0.83727
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1092 Residual
## Null Deviance:       1054
## Residual Deviance: 1000  AIC: 1125
## log likelihood:  -500.1773
## Nagelkerke R2:  0.07766884
## % pres/err predicted correctly:  -333.852
## % of predictable range [ (model-null)/(1-null) ]:  0.05329974
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.05115      0.06791     -0.79659
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1093 Residual
## Null Deviance:       1054
## Residual Deviance: 1005  AIC: 1130
## log likelihood:  -502.6491
## Nagelkerke R2:  0.07070304
## % pres/err predicted correctly:  -336.6025
## % of predictable range [ (model-null)/(1-null) ]:  0.04552359
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)      stimlen          pos
##     2.56119     -0.07237     -0.22179
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1093 Residual
## Null Deviance:       1054
## Residual Deviance: 1015  AIC: 1137
## log likelihood:  -507.5905
## Nagelkerke R2:  0.05668269
## % pres/err predicted correctly:  -338.1284
## % of predictable range [ (model-null)/(1-null) ]:  0.04120952
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.1740      -0.2497
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1094 Residual
## Null Deviance:       1054
## Residual Deviance: 1017  AIC: 1138
## log likelihood:  -508.4859
## Nagelkerke R2:  0.05412843
## % pres/err predicted correctly:  -339.2814
## % of predictable range [ (model-null)/(1-null) ]:  0.03794971
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.17933     -0.15741     -0.41250      0.02517
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1092 Residual
## Null Deviance:       1054
## Residual Deviance: 1014  AIC: 1139
## log likelihood:  -507.133
## Nagelkerke R2:  0.05798609
## % pres/err predicted correctly:  -338.1095
## % of predictable range [ (model-null)/(1-null) ]:  0.04126307
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5307      -0.1809
```

```
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1094 Residual
## Null Deviance:       1054
## Residual Deviance: 1040  AIC: 1161
## log likelihood:  -519.8433
## Nagelkerke R2:  0.0213673
## % pres/err predicted correctly:  -346.5045
## % of predictable range [ (model-null)/(1-null) ]:  0.01752853
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.322
##
## Degrees of Freedom: 1095 Total (i.e. Null);  1095 Residual
## Null Deviance:       1054
## Residual Deviance: 1054  AIC: 1178
## log likelihood:  -527.1259
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -352.7044
## % 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)	1120.548	0.000000	1.000000	0.8839191	0.0848652	1.3140080	0.1622674	-0.4801846	-0.6092586	0.1158504	0.0007881
preserved ~ stimlen + I(pos^2) + pos	1124.769	4.220607	0.121200	0.1071321	0.0776688	1.3854311	-0.1235639	-0.8372669	NA	0.0788438	NA
preserved ~ I(pos^2) + pos	1129.845	9.296806	0.009576	0.0008465	0.0207070	1.3051149	NA	-0.7965902	NA	0.0679110	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	stimlen	pos	stimlen:pos	I(pos^2)	stimlen:I(pos^2)
preserved ~ stimlen + pos	1136.843	36.2946407	0.0002895	0.0002559	0.0566827	2.7561185	-0.0723710	-0.2217922	NA	NA	NA
preserved ~ pos	1138.025	37.4768850	0.0001603	0.0001417	0.0541284	2.4174023	NA	-0.2496649	NA	NA	NA
preserved ~ stimlen * pos	1139.024	38.4754780	0.0000973	0.0000860	0.0579863	3.1179330	-0.1574124	-1.4124985	0.0251657	NA	NA
preserved ~ stimlen	1161.054	60.5105330	0.0000000	0.0000000	0.0213672	2.3530652	-0.1808858	NA	NA	NA	NA
preserved ~ 1	1178.395	77.8478650	0.0000000	0.0000000	0.0000000	1.0321684	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.
```

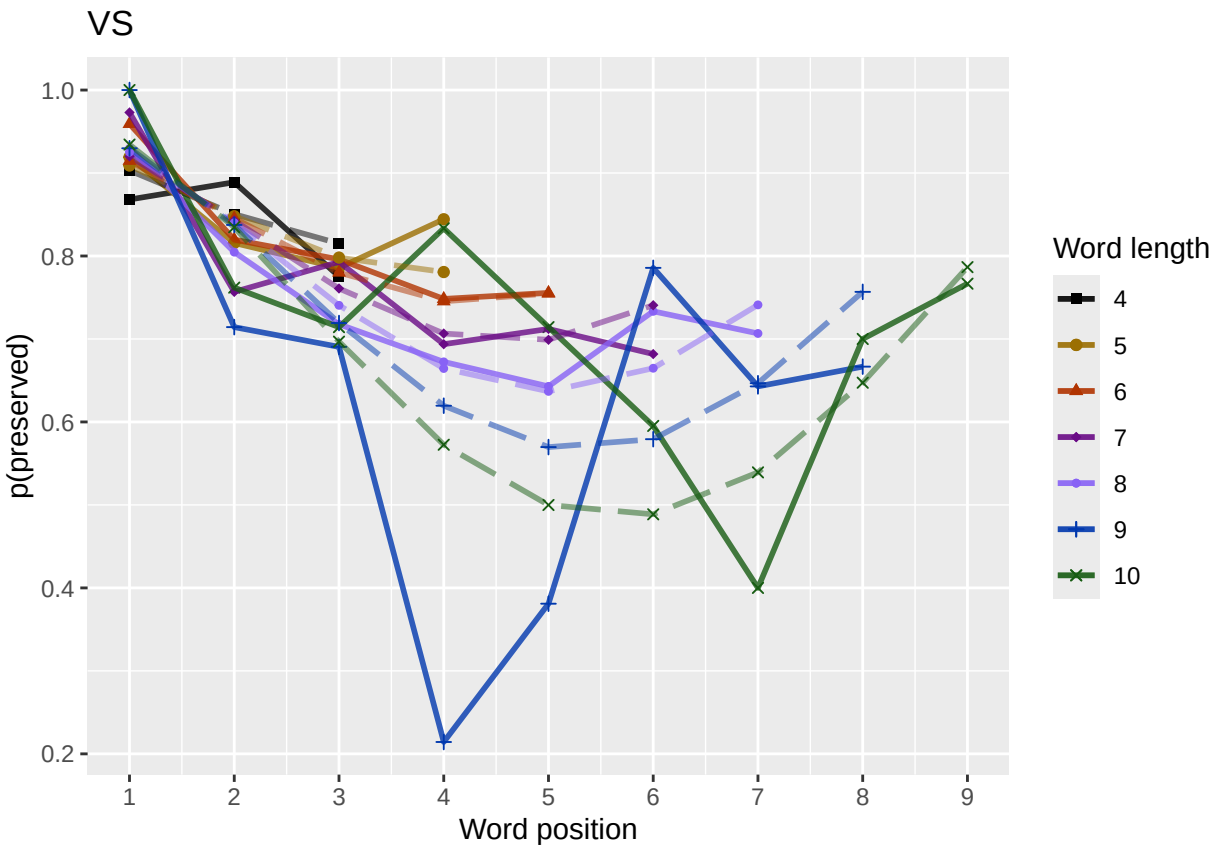
```r
nofrag_fitted_pos_len_table <- nofrag_fitted_pos_len_summary %>% pivot_wider(names_from = pos, values_f
nofrag_fitted_pos_len_table
```

```
## # 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 0.903 0.850 0.815 NA     NA     NA     NA     NA     NA
## 2       5 0.909 0.848 0.798  0.781 NA     NA     NA     NA     NA
## 3       6 0.915 0.845 0.780  0.745  0.755 NA     NA     NA     NA
## 4       7 0.920 0.843 0.761  0.707  0.699  0.741 NA     NA     NA
## 5       8 0.925 0.840 0.741  0.664  0.637  0.665  0.741 NA     NA
## 6       9 0.930 0.837 0.719  0.620  0.570  0.579  0.647  0.757 NA
## 7      10 0.934 0.835 0.697  0.573  0.500  0.489  0.539  0.647  0.787
```

```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: 0.14 - 1.08"
```

```
# 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] -0.02352326
```

```
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] -0.06850799
```

```
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] "Average upward change after U minimum"
## [1] 0.04419634
```

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

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

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

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

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

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

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

```
## [1] "differences from left max to min for each row: "
## [1] 0.1112980 0.1718818 0.2290983 0.2823999 0.3511703 0.4265012 0.5016740
## [1] "differences from min to right max for each row: "
## [1] 0.000000000 0.000000000 0.000000000 0.003914613 0.034292583 0.084107194 0.147593997
## [1] " "
## [1] "row with biggest return upward"
## [1] 7
## [1] " "
## [1] "downward distance for row with the largest upward value"
## [1] 0.501674
## [1] 0.147594
## [1] " "
## [1] "Return upward as a proportion of the downward distance:"
## [1] 0.294203
```

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

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

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

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

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

```
## ***************************
## model index:  12
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          3.715788          -0.088948           0.074310          -0.879807          -0.069920
## I(pos^2):log_freq       pos:log_freq
##          0.009419          -0.005064
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1139 Residual
## Null Deviance:       1198
## Residual Deviance: 1097  AIC: 1223
```

```
## log likelihood:  -548.7448
## Nagelkerke R2:  0.1291611
## % pres/err predicted correctly:  -371.1667
## % of predictable range [ (model-null)/(1-null) ]:  0.09277824
## ***************************
## 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
##          3.136629           0.066374          -0.849312          -0.048782           0.008859
##      pos:log_freq
##         -0.003672
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1100  AIC: 1225
## log likelihood:  -550.0299
## Nagelkerke R2:  0.1259882
## % pres/err predicted correctly:  -372.8448
## % of predictable range [ (model-null)/(1-null) ]:  0.08868763
## ***************************
## model index:  10
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           log_freq           I(pos^2)                pos
##          3.726525          -0.091724           0.015049           0.074800          -0.881466
##  stimlen:log_freq  log_freq:I(pos^2)       log_freq:pos
##         -0.013508           0.010546          -0.008547
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1138 Residual
## Null Deviance:       1198
## Residual Deviance: 1097  AIC: 1225
## log likelihood:  -548.665
## Nagelkerke R2:  0.1293579
## % pres/err predicted correctly:  -371.1232
## % of predictable range [ (model-null)/(1-null) ]:  0.09288428
## ***************************
## model index:  21
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          I(pos^2)               pos  stimlen:I(pos^2)
##        2.4580412         0.1324494         0.0961748        -0.5561625         0.0001089
##      stimlen:pos
##       -0.0704543
##
```

```
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1105  AIC: 1226
## log likelihood:  -552.3462
## Nagelkerke R2:  0.1202509
## % pres/err predicted correctly:  -374.0831
## % of predictable range [ (model-null)/(1-null) ]:  0.08566906
## ***************************
## model index:  11
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq          I(pos^2)               pos
##          3.52472          -0.06931          -0.13352           0.06233          -0.81129
## stimlen:log_freq
##          0.03101
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1104  AIC: 1227
## log likelihood:  -552.232
## Nagelkerke R2:  0.1205344
## % pres/err predicted correctly:  -373.8746
## % of predictable range [ (model-null)/(1-null) ]:  0.08617736
## ***************************
## model index:  13
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos     log_freq
##     3.52758     -0.07594      0.06091     -0.79869      0.07795
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1141 Residual
## Null Deviance:       1198
## Residual Deviance: 1106  AIC: 1228
## log likelihood:  -552.8269
## Nagelkerke R2:  0.1190573
## % pres/err predicted correctly:  -374.3231
## % of predictable range [ (model-null)/(1-null) ]:  0.08508409
## ***************************
## model index:  20
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     I(pos^2)          pos
##     3.65275     -0.09606      0.06080     -0.79704
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
```

```
## Null Deviance:       1198
## Residual Deviance: 1108  AIC: 1228
## log likelihood:  -554.1995
## Nagelkerke R2:  0.1156437
## % pres/err predicted correctly:  -375.6505
## % of predictable range [ (model-null)/(1-null) ]:  0.0818483
## ***************************
## model index:  19
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.03432      0.05256     -0.76748
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1112  AIC: 1231
## log likelihood:  -555.7954
## Nagelkerke R2:  0.1116646
## % pres/err predicted correctly:  -377.7723
## % of predictable range [ (model-null)/(1-null) ]:  0.07667614
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)           pos      log_freq  pos:log_freq
##      2.25525      -0.31039      -0.11096       0.05271
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1113  AIC: 1232
## log likelihood:  -556.261
## Nagelkerke R2:  0.1105015
## % pres/err predicted correctly:  -376.0663
## % of predictable range [ (model-null)/(1-null) ]:  0.08083469
## ***************************
## model index:  7
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##  (Intercept)       stimlen           pos      log_freq  pos:log_freq
##      2.44272      -0.03526      -0.29686      -0.11943       0.05319
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1141 Residual
## Null Deviance:       1198
## Residual Deviance: 1112  AIC: 1233
## log likelihood:  -556.0448
```

```
## Nagelkerke R2:  0.1110418
## % pres/err predicted correctly:  -375.6696
## % of predictable range [ (model-null)/(1-null) ]:  0.08180166
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##         2.444704         -0.036113         -0.098190         -0.296343         -0.004097
##     log_freq:pos
##         0.054904
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1112  AIC: 1235
## log likelihood:  -556.037
## Nagelkerke R2:  0.1110613
## % pres/err predicted correctly:  -375.6719
## % of predictable range [ (model-null)/(1-null) ]:  0.08179602
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos     log_freq
##      2.2822      -0.3224       0.0839
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1117  AIC: 1235
## log likelihood:  -558.4636
## Nagelkerke R2:  0.104987
## % pres/err predicted correctly:  -377.9505
## % of predictable range [ (model-null)/(1-null) ]:  0.07624154
## ***************************
## model index:  8
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos     log_freq
##     2.44952     -0.03147     -0.31033      0.07765
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1117  AIC: 1236
## log likelihood:  -558.29
## Nagelkerke R2:  0.1054224
```

```
## % pres/err predicted correctly:  -377.5906
## % of predictable range [ (model-null)/(1-null) ]:  0.07711892
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq               pos  stimlen:log_freq
##          2.42747          -0.02528          -0.09167          -0.31149           0.02502
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1141 Residual
## Null Deviance:       1198
## Residual Deviance: 1116  AIC: 1237
## log likelihood:  -557.898
## Nagelkerke R2:  0.106405
## % pres/err predicted correctly:  -377.212
## % of predictable range [ (model-null)/(1-null) ]:  0.07804178
## ***************************
## model index:  17
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos
##     2.57603     -0.05162     -0.30950
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1119  AIC: 1237
## log likelihood:  -559.6582
## Nagelkerke R2:  0.1019872
## % pres/err predicted correctly:  -378.9161
## % of predictable range [ (model-null)/(1-null) ]:  0.07388783
## ***************************
## model index:  16
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.3047      -0.3303
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1120  AIC: 1237
## log likelihood:  -560.1503
## Nagelkerke R2:  0.1007497
## % pres/err predicted correctly:  -379.715
## % of predictable range [ (model-null)/(1-null) ]:  0.07194047
## ***************************
```

```
## model index:  18
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen          pos  stimlen:pos
##     3.17876     -0.13451     -0.48829      0.02356
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1118  AIC: 1239
## log likelihood:  -559.2009
## Nagelkerke R2:  0.1031363
## % pres/err predicted correctly:  -378.7755
## % of predictable range [ (model-null)/(1-null) ]:  0.07423062
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen     log_freq
##     2.45691     -0.19647      0.07361
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1171  AIC: 1289
## log likelihood:  -585.5887
## Nagelkerke R2:  0.03530678
## % pres/err predicted correctly:  -397.4675
## % of predictable range [ (model-null)/(1-null) ]:  0.0286655
## **************************
## model index:  15
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5773      -0.2152
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1174  AIC: 1290
## log likelihood:  -586.886
## Nagelkerke R2:  0.03189073
## % pres/err predicted correctly:  -398.7117
## % of predictable range [ (model-null)/(1-null) ]:  0.02563249
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
##      (Intercept)           stimlen          log_freq  stimlen:log_freq
##          2.43732          -0.19164          -0.06994           0.02104
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1171  AIC: 1290
## log likelihood:  -585.2906
## Nagelkerke R2:  0.03609067
## % pres/err predicted correctly:  -397.086
## % of predictable range [ (model-null)/(1-null) ]:  0.02959549
## ****************************
## model index:  14
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##        1.13
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1145 Residual
## Null Deviance:       1198
## Residual Deviance: 1198  AIC: 1317
## log likelihood:  -598.8578
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -409.2269
## % of predictable range [ (model-null)/(1-null) ]:  0
## ****************************
```

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

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

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

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

Model	AIC	DeltaAIC	expAIC	wAIC	NagR2	(Intercept)	stimlen	log_freq	stimlen:log_freq	pos	log_freq:pos	I(pos^2)	log_freq:I(pos^2)	stimlen:pos	stimlen:I(pos^2)
preserved ~ stimlen + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA
preserved ~ (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.0487820	NA	[illegible]	[illegible]	NA	[illegible]	[illegible]	NA	NA
preserved ~ stimlen * log_freq + (I(pos^2) + pos) * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0150488	[illegible]	[illegible]	NA	-0.0085471	[illegible]	0.0105456	NA	NA
preserved ~ stimlen * (I(pos^2) + pos)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.5561625	NA	NA	[illegible]	NA		-0.0704543	0.0001089
preserved ~ stimlen * log_freq + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.8112864	NA	NA	[illegible]	NA	NA	NA	
preserved ~ stimlen + I(pos^2) + pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0759403	[illegible]	-0.7986860	NA	NA	[illegible]	NA	NA	NA	
preserved ~ stimlen + I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.0960609	NA	-0.7970437	NA	NA	[illegible]	NA	NA	NA	
preserved ~ I(pos^2) + pos	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	NA	-0.7674843	NA	NA	[illegible]	NA	NA	NA	
preserved ~ pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	-0.1109561	-0.3103901	[illegible]	NA	NA	NA	NA	NA	
preserved ~ stimlen + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	-0.2968634	[illegible]	NA	NA	NA	NA	NA	
preserved ~ stimlen * log_freq + pos * log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	NA	NA	NA	NA	
preserved ~ pos + log_freq	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	NA	[illegible]	-0.3223718	NA	NA	NA	NA	NA	NA	

Model	AIC DeltaAIC AICexp AICw NagR2 (Intercept) stimlen log_freq stimlen:log_freq pos log_freq:pos I(pos^2) log_freq:I(pos^2) stimlen:pos stimlen:I(pos^2) [overlapping, illegible]											
preserved ~ stimlen + pos + log_freq	1236.[illegible]	-0.0314728	0.0776548	NA	-0.3103333	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq + pos + pos	1236.[illegible]	-0.025[illegible]	-[illegible]16741	0.0250212	-0.3114902	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen + pos	1236.[illegible]	-0.0516204	NA	NA	-0.3095010	NA	NA	NA	NA	NA	NA	NA
preserved ~ pos	1236.[illegible]	NA	NA	NA	-0.3302605	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * pos	1238.[illegible]	-0.1345149	NA	NA	-0.4882851	NA	NA	NA	NA	NA	0.0235621	NA
preserved ~ stimlen + log_freq	1289.[illegible]	-0.1964662	0.0736070	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen	1289.[illegible]	-0.2152113	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ stimlen * log_freq	1289.[illegible]	-0.19163[illegible]	-[illegible]9439	0.0210363	NA	NA	NA	NA	NA	NA	NA	NA
preserved ~ 1	1317.[illegible]	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

```r
print(BestFLPModelFormula)
```

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

```r
print(BestFLPModel)
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
##       (Intercept)            stimlen           I(pos^2)                pos           log_freq
##          3.715788          -0.088948           0.074310          -0.879807          -0.069920
## I(pos^2):log_freq       pos:log_freq
##          0.009419          -0.005064
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1139 Residual
## Null Deviance:       1198
## Residual Deviance: 1097  AIC: 1223
```

```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_preserv
```

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

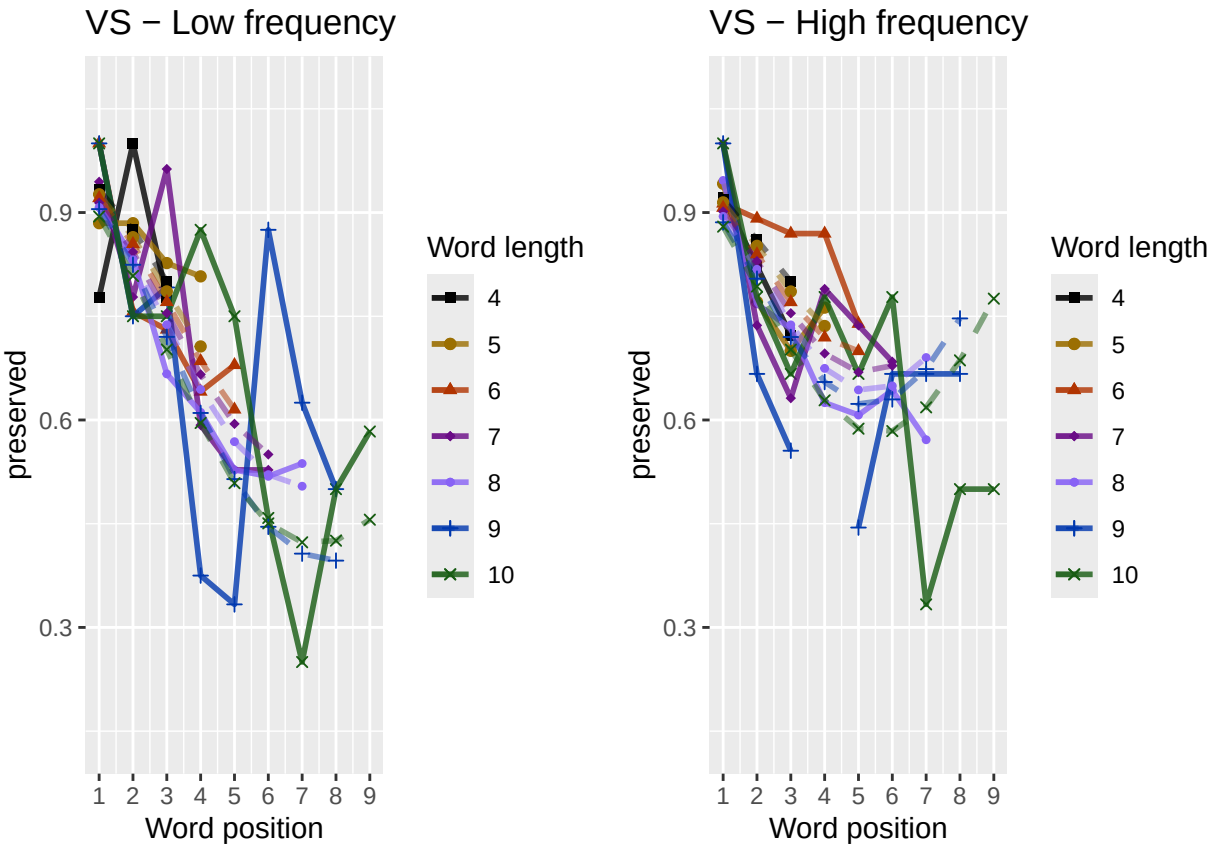
```
LF_Plot <- plot_len_pos_obs_predicted(LFDat,paste0(CurPat, " - Low frequency"),"FLPFitted",c(min_preserv
```

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

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

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

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




```
# only main effects
MEModelEquations<-c(
  "preserved ~ CumPres",
  "preserved ~ CumErr",
```

```
  "preserved ~ (I(pos^2)+pos)",
  "preserved ~ pos",
  "preserved ~ stimlen",
  "preserved ~ 1"
)
MERes<-TestModels(MEModelEquations,PosDat)
```

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6111      -0.8785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1065  AIC: 1170
## log likelihood:  -532.3105
## Nagelkerke R2:  0.1691178
## % pres/err predicted correctly:  -351.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.1411276
## ****************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.03432      0.05256     -0.76748
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1112  AIC: 1231
## log likelihood:  -555.7954
## Nagelkerke R2:  0.1116646
## % pres/err predicted correctly:  -377.7723
## % of predictable range [ (model-null)/(1-null) ]:  0.07667614
## ****************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
```

```
##
## Coefficients:
## (Intercept)          pos
##      2.3047      -0.3303
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1120  AIC: 1237
## log likelihood:  -560.1503
## Nagelkerke R2:  0.1007497
## % pres/err predicted correctly:  -379.715
## % of predictable range [ (model-null)/(1-null) ]:  0.07194047
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5773      -0.2152
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1174  AIC: 1290
## log likelihood:  -586.886
## Nagelkerke R2:  0.03189073
## % pres/err predicted correctly:  -398.7117
## % of predictable range [ (model-null)/(1-null) ]:  0.02563249
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.4409      -0.1606
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1185  AIC: 1306
## log likelihood:  -592.4999
## Nagelkerke R2:  0.01701924
## % pres/err predicted correctly:  -404.7941
## % of predictable range [ (model-null)/(1-null) ]:  0.01080578
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
```

```
##        1.13
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1145 Residual
## Null Deviance:       1198
## Residual Deviance: 1198  AIC: 1317
## log likelihood:  -598.8578
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -409.2269
## % 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(posˆ2)	pos	stimlen
preserved ~ CumErr	1170.246	0.00000	1	1	0.1691178	8.611131	NA	-0.8785357	NA	NA	NA
preserved ~ (I(posˆ2) + pos)	1231.178	60.93274	0	0	0.1116648	3.034318	NA	NA	0.0525631	-0.7674843	NA
preserved ~ pos	1236.896	66.64387	0	0	0.1007492	7.304664	NA	NA	NA	-0.3302605	NA
preserved ~ stimlen	1289.536	119.28984	0	0	0.0318907	7.577300	NA	NA	NA	NA	-0.2152113
preserved ~ CumPres	1306.484	136.23789	0	0	0.0170192	2.440891	-0.1606413	NA	NA	NA	NA
preserved ~ 1	1317.068	146.82275	0	0	0.0000000	0.129883	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	0.7944820	148
O	0.7869539	557
P	1.0000000	14
S	0.7760417	64
V	0.6792929	363

```
#  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
##      1.5754      -0.8871
##
## Degrees of Freedom: 1067 Total (i.e. Null);  1066 Residual
## Null Deviance:       1124
## Residual Deviance: 1004  AIC: 1106
## log likelihood:  -502.1047
## Nagelkerke R2:  0.1631089
## % pres/err predicted correctly:  -332.7825
## % of predictable range [ (model-null)/(1-null) ]:  0.1359889
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.85699      0.04146     -0.67386
##
## Degrees of Freedom: 1067 Total (i.e. Null);  1065 Residual
## Null Deviance:       1124
## Residual Deviance: 1047  AIC: 1162
## log likelihood:  -523.3755
## Nagelkerke R2:  0.1073089
## % pres/err predicted correctly:  -356.4348
## % of predictable range [ (model-null)/(1-null) ]:  0.07476398
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2890      -0.3313
##
## Degrees of Freedom: 1067 Total (i.e. Null);  1066 Residual
## Null Deviance:       1124
## Residual Deviance: 1052  AIC: 1164
## log likelihood:  -525.8577
## Nagelkerke R2:  0.1006515
## % pres/err predicted correctly:  -357.4172
## % of predictable range [ (model-null)/(1-null) ]:  0.07222094
```

```
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5927      -0.2207
##
## Degrees of Freedom: 1067 Total (i.e. Null);  1066 Residual
## Null Deviance:       1124
## Residual Deviance: 1100  AIC: 1212
## log likelihood:  -550.1985
## Nagelkerke R2:  0.03370043
## % pres/err predicted correctly:  -374.8246
## % of predictable range [ (model-null)/(1-null) ]:  0.0271611
## **************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.4041      -0.1665
##
## Degrees of Freedom: 1067 Total (i.e. Null);  1066 Residual
## Null Deviance:       1124
## Residual Deviance: 1112  AIC: 1229
## log likelihood:  -556.2223
## Nagelkerke R2:  0.01665509
## % pres/err predicted correctly:  -381.2262
## % of predictable range [ (model-null)/(1-null) ]:  0.01059022
## **************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.106
##
## Degrees of Freedom: 1067 Total (i.e. Null);  1067 Residual
## Null Deviance:       1124
## Residual Deviance: 1124  AIC: 1239
## log likelihood:  -562.0432
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -385.3174
## % of predictable range [ (model-null)/(1-null) ]:  0
## **************************
```

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

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	1105.733	0.00000	1	1	0.1631089	1.575407	NA	-0.8871398	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1161.913	56.17983	0	0	0.1073089	2.856994	NA	NA	0.0414611	-0.6738558	NA
preserved ~ pos	1163.974	58.24171	0	0	0.1006512	3.288962	NA	NA	NA	-0.3313005	NA
preserved ~ stimlen	1211.596	105.86302	0	0	0.0337002	4.592733	NA	NA	NA	NA	-0.2206723
preserved ~ CumPres	1228.912	123.17939	0	0	0.0166551	1.404107	-0.1665184	NA	NA	NA	NA
preserved ~ 1	1238.555	132.82220	0	0	0.0000000	0.106117	NA	NA	NA	NA	NA

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.4373      -0.8372
##
## Degrees of Freedom: 919 Total (i.e. Null);  918 Residual
## Null Deviance:       979.9
## Residual Deviance: 904.3     AIC: 992.3
```

```
## log likelihood:  -452.1577
## Nagelkerke R2:  0.1203675
## % pres/err predicted correctly:  -303.2061
## % of predictable range [ (model-null)/(1-null) ]:  0.1008214
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     2.96013      0.05435     -0.77864
##
## Degrees of Freedom: 919 Total (i.e. Null);  917 Residual
## Null Deviance:       979.9
## Residual Deviance: 903.1     AIC: 1003
## log likelihood:  -451.5583
## Nagelkerke R2:  0.122198
## % pres/err predicted correctly:  -308.4632
## % of predictable range [ (model-null)/(1-null) ]:  0.08528248
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)          pos
##      2.2460      -0.3345
##
## Degrees of Freedom: 919 Total (i.e. Null);  918 Residual
## Null Deviance:       979.9
## Residual Deviance: 911.1     AIC: 1007
## log likelihood:  -455.5531
## Nagelkerke R2:  0.1099537
## % pres/err predicted correctly:  -310.4523
## % of predictable range [ (model-null)/(1-null) ]:  0.07940309
## **************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##       2.547       -0.220
##
## Degrees of Freedom: 919 Total (i.e. Null);  918 Residual
## Null Deviance:       979.9
## Residual Deviance: 958.8     AIC: 1055
## log likelihood:  -479.4124
## Nagelkerke R2:  0.03456837
## % pres/err predicted correctly:  -327.6004
```

```
## % of predictable range [ (model-null)/(1-null) ]:  0.02871625
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.5186      -0.2952
##
## Degrees of Freedom: 919 Total (i.e. Null);  918 Residual
## Null Deviance:       979.9
## Residual Deviance: 956.5     AIC: 1057
## log likelihood:  -478.2568
## Nagelkerke R2:  0.03831034
## % pres/err predicted correctly:  -328.5059
## % of predictable range [ (model-null)/(1-null) ]:  0.02603998
## ***************************
## model index:  6
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)
##       1.069
##
## Degrees of Freedom: 919 Total (i.e. Null);  919 Residual
## Null Deviance:       979.9
## Residual Deviance: 979.9     AIC: 1080
## log likelihood:  -489.9527
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -337.3156
## % 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)
```

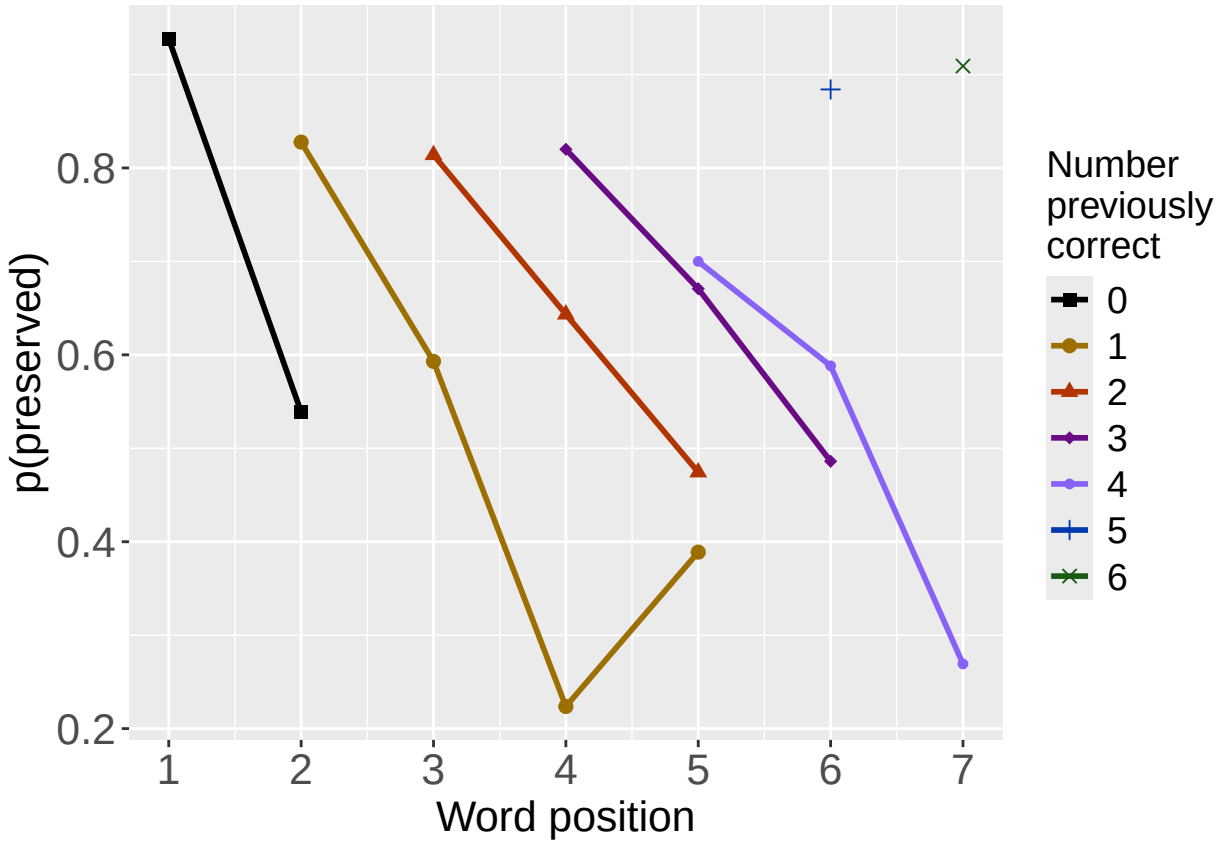
Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ CumErr	992.2974	0.000000	1.000000	0.9934218	0.1203675	1.437278	NA	-0.8371733	NA	NA	NA
preserved ~ (I(pos^2) + pos)	1002.530	10.23310	0.005996	7.005957e-02	0.1221982	2.960128	NA	NA	0.0543548	-0.7786380	NA
preserved ~ pos	1007.052	14.75516	0.000625	1.000621e-02	0.1099537	7.245996	NA	NA	NA	-0.3345486	NA
preserved ~ stimlen	1054.684	62.38735	0.000000	0.000000e+00	0.0345682	4.547273	NA	NA	NA	NA	-0.2200231
preserved ~ CumPres	1056.873	64.57565	0.000000	0.000000e+00	0.0383103	3.518570	-0.2952052	NA	NA	NA	NA

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumPres	CumErr	I(pos^2)	pos	stimlen
preserved ~ 1	1079.619	37.32195	0.0000000	0.0000000	0.0000000	1.069357	NA	NA	NA	NA	NA

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

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

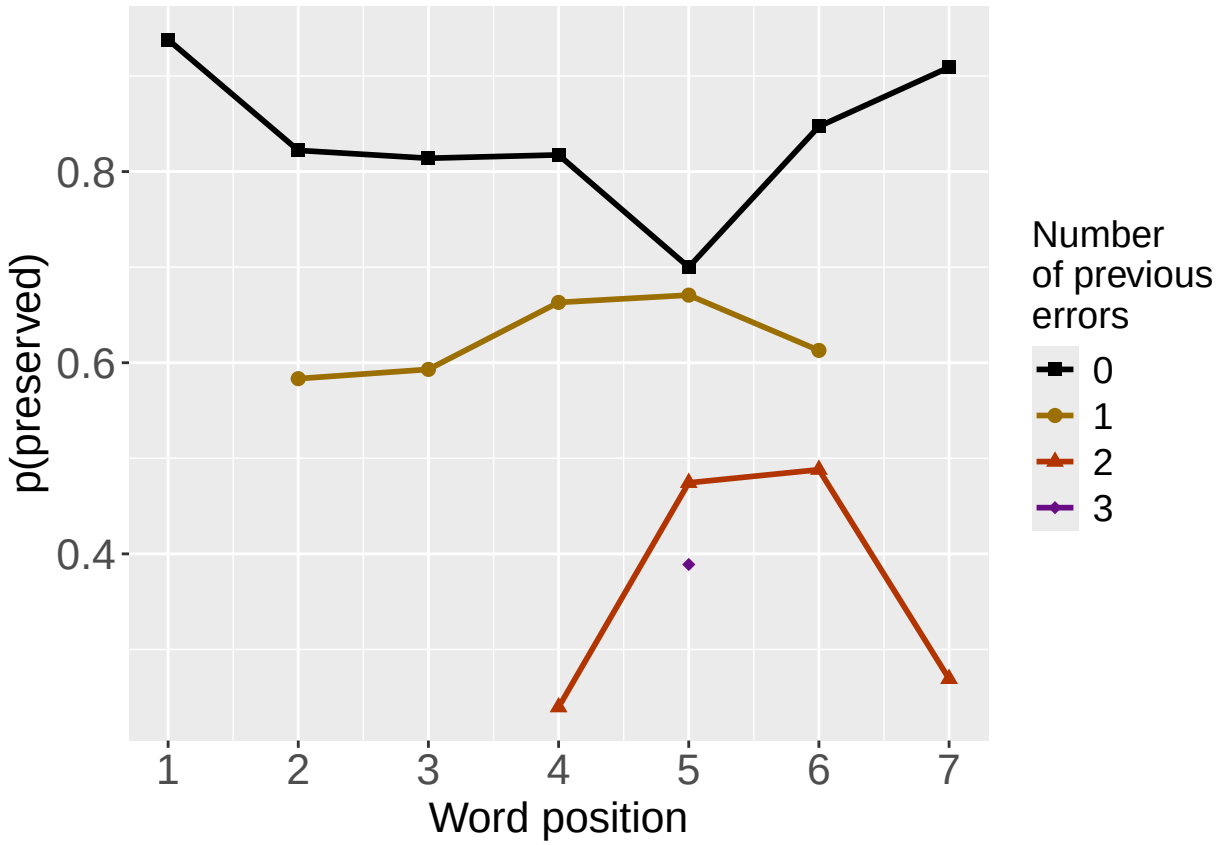
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

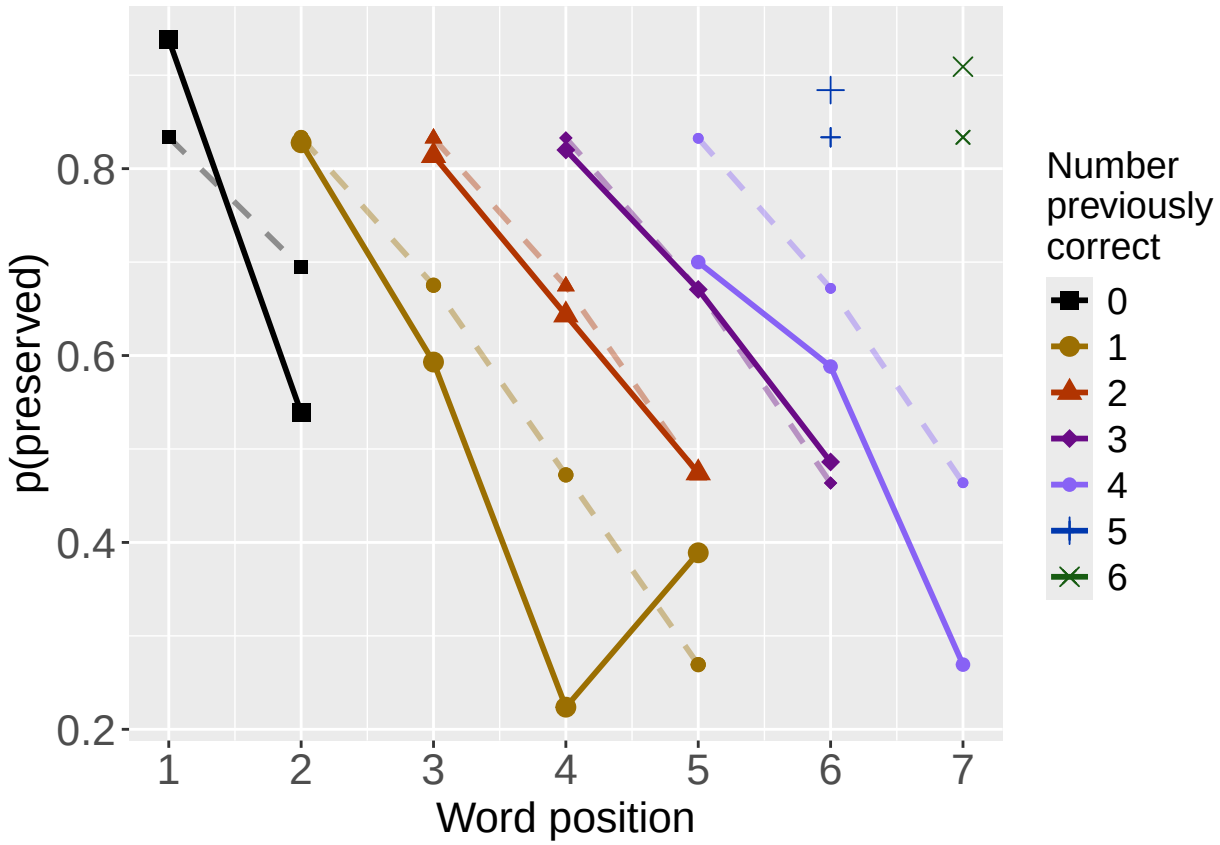
```
PlotName<-paste0(FigDir,CurPat,"_",CurTask,"_PrevErrPlot_Obs")
ggsave(filename=paste0(PlotName,".tif"),plot=PrevErrPlot,device="tiff",compression = "lzw")
```

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

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

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

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

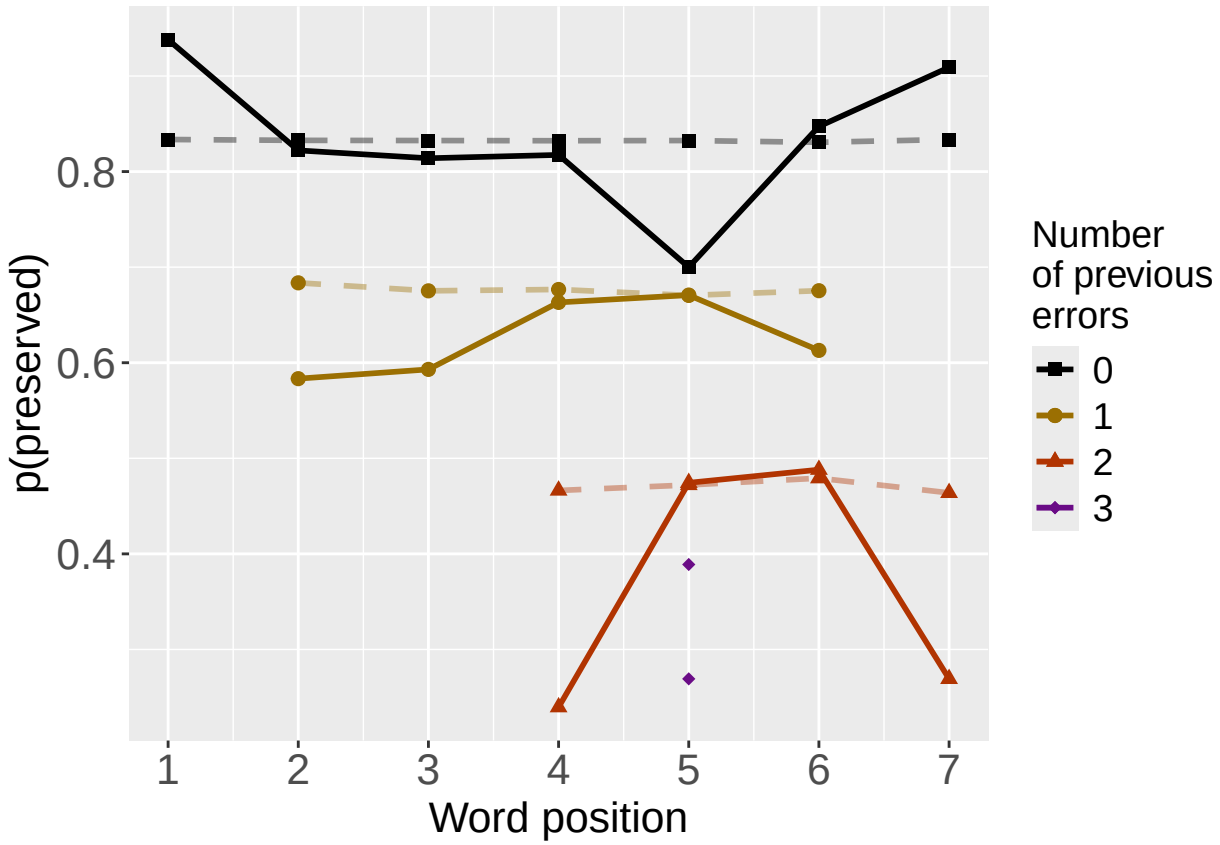
```
print(PrevCorPlot)
```




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

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

```
print(PrevErrPlot)
```




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

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

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

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

```r
CumAICSummary <- NULL
```

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

# After establishing the primary variable, see about additions

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

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

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

```
## ****************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.98067     -0.78414      0.08064     -0.75785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1041  AIC: 1152
## log likelihood:  -520.5332
## Nagelkerke R2:  0.1970551
## % pres/err predicted correctly:  -345.4428
## % of predictable range [ (model-null)/(1-null) ]:  0.1554848
```

```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6111      -0.8785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1065  AIC: 1170
## log likelihood:  -532.3105
## Nagelkerke R2:  0.1691178
## % pres/err predicted correctly:  -351.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.1411276
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)     I(pos^2)          pos
##     3.03432      0.05256     -0.76748
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1112  AIC: 1231
## log likelihood:  -555.7954
## Nagelkerke R2:  0.1116646
## % pres/err predicted correctly:  -377.7723
## % of predictable range [ (model-null)/(1-null) ]:  0.07667614
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos
preserved ~ CumErr + I(pos^2) + pos	1152.370	0.00000	1.0000000	0.9998687	0.1970551	2.980666	-0.7841392	0.0806417	-0.7578550

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(posˆ2)	pos
preserved ~ CumErr	1170.246	17.87587	0.0001313	0.0001313	0.1691178	1.611131	-0.8785357	NA	NA
preserved ~ I(posˆ2) + pos	1231.178	78.80861	0.0000000	0.0000000	0.1116646	3.034318	NA	0.0525631	-0.7674843

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"stimlen"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      stimlen
##     2.06209     -0.84261     -0.07035
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1063  AIC: 1169
## log likelihood:  -531.2849
## Nagelkerke R2:  0.1715735
## % pres/err predicted correctly:  -350.4463
## % of predictable range [ (model-null)/(1-null) ]:  0.1432879
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6111      -0.8785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1065  AIC: 1170
## log likelihood:  -532.3105
## Nagelkerke R2:  0.1691178
## % pres/err predicted correctly:  -351.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.1411276
## ***************************
## model index:  3
```

```
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      stimlen
##      2.5773      -0.2152
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1174  AIC: 1290
## log likelihood:  -586.886
## Nagelkerke R2:  0.03189073
## % pres/err predicted correctly:  -398.7117
## % of predictable range [ (model-null)/(1-null) ]:  0.02563249
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	stimlen
preserved ~ CumErr + stimlen	1168.919	0.000000	1.0000000	0.6600369	0.1715735	2.062087	-0.8426129	-0.0703521
preserved ~ CumErr	1170.246	1.326917	0.5150668	0.3399631	0.1691178	1.611131	-0.8785357	NA
preserved ~ stimlen	1289.536	120.616762	0.0000000	0.0000000	0.0318907	2.577300	NA	-0.2152113

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

BestMEFactor<-BestMEModelFormula
OtherFactor<-"CumPres"

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr      CumPres
##      1.8321      -0.8535      -0.1185
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
```

```
## Null Deviance:       1198
## Residual Deviance: 1059  AIC: 1167
## log likelihood:  -529.3611
## Nagelkerke R2:  0.1761683
## % pres/err predicted correctly:  -350.26
## % of predictable range [ (model-null)/(1-null) ]:  0.1437421
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6111      -0.8785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1065  AIC: 1170
## log likelihood:  -532.3105
## Nagelkerke R2:  0.1691178
## % pres/err predicted correctly:  -351.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.1411276
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)      CumPres
##      1.4409      -0.1606
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1185  AIC: 1306
## log likelihood:  -592.4999
## Nagelkerke R2:  0.01701924
## % pres/err predicted correctly:  -404.7941
## % of predictable range [ (model-null)/(1-null) ]:  0.01080578
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	CumPres
preserved ~ CumErr + CumPres	1167.082	0.000000	1.0000000	0.8294589	0.1761683	1.832128	-0.8534941	-0.1185270
preserved ~ CumErr	1170.246	3.163595	0.2056052	0.1705411	0.1691178	1.611131	-0.8785357	NA
preserved ~ CumPres	1306.484	139.401484	0.0000000	0.0000000	0.0170192	1.440891	NA	-0.1606413

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

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

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

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

```
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      1.9507      -0.7350      -0.1185
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1059  AIC: 1167
## log likelihood:  -529.3611
## Nagelkerke R2:  0.1761683
## % pres/err predicted correctly:  -350.26
## % of predictable range [ (model-null)/(1-null) ]:  0.1437421
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6111      -0.8785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1065  AIC: 1170
## log likelihood:  -532.3105
## Nagelkerke R2:  0.1691178
## % pres/err predicted correctly:  -351.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.1411276
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
```

```
## Coefficients:
## (Intercept)          pos
##      2.3047      -0.3303
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1120  AIC: 1237
## log likelihood:  -560.1503
## Nagelkerke R2:  0.1007497
## % pres/err predicted correctly:  -379.715
## % of predictable range [ (model-null)/(1-null) ]:  0.07194047
## ***************************
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	pos
preserved ~ CumErr + pos	1167.082	0.000000	1.0000000	0.8294589	0.1761683	1.950655	-0.7349671	-0.1185270
preserved ~ CumErr	1170.246	3.163595	0.2056052	0.1705411	0.1691178	1.611131	-0.8785357	NA
preserved ~ pos	1236.890	69.807462	0.0000000	0.0000000	0.1007497	2.304664	NA	-0.3302605

```
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	1152.370	0.000000	1.000000	0.9998687	0.1970552	1.980666	-0.7841392	0.0806417	-0.7578550	NA	NA
preserved ~ CumErr + CumPres	1167.082	0.000000	1.000000	0.8294589	0.1761683	1.832128	-0.8534941	NA	NA	NA	-0.1185270
preserved ~ CumErr + pos	1167.082	0.000000	1.000000	0.8294589	0.1761683	1.950655	-0.7349671	NA	-0.1185270	NA	NA
preserved ~ CumErr + stimlen	1168.910	0.000000	1.000000	0.6600369	0.1715735	2.062087	-0.8426129	NA	NA	-0.0703521	NA
preserved ~ CumErr	1170.246	7.875869	0.000131	0.000131	0.1691178	1.611131	-0.8785357	NA	NA	NA	NA
preserved ~ CumErr	1170.246	6.326917	0.515066	0.3399631	0.1691178	1.611131	-0.8785357	NA	NA	NA	NA
preserved ~ CumErr	1170.246	6.163595	0.205605	0.1705411	0.1691178	1.611131	-0.8785357	NA	NA	NA	NA
preserved ~ CumErr	1170.246	6.163595	0.205605	0.1705411	0.1691178	1.611131	-0.8785357	NA	NA	NA	NA
preserved ~ I(pos^2) + pos	1231.178	68.808614	0.000000	0.0000000	0.1116646	2.034318	NA	0.0525631	-0.7674843	NA	NA
preserved ~ pos	1236.890	69.807462	0.000000	0.0000000	0.1007497	2.304664	NA	NA	-0.3302605	NA	NA
preserved ~ stimlen	1289.536	120.616762	0.000000	0.0000000	0.0031890	2.577300	NA	NA	NA	-0.2152113	NA
preserved ~ CumPres	1306.484	139.401484	0.000000	0.0000000	0.0017019	2.440891	NA	NA	NA	NA	-0.1606413

```r
# explore influence of frequency and length

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

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

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

```
## ***************************
## model index:  5
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen     log_freq
##     3.44035     -0.78359      0.08927     -0.79206     -0.06988      0.08060
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1140 Residual
## Null Deviance:       1198
## Residual Deviance: 1036  AIC: 1149
## log likelihood:  -517.8019
## Nagelkerke R2:  0.2034527
## % pres/err predicted correctly:  -343.1954
## % of predictable range [ (model-null)/(1-null) ]:  0.1609632
## ***************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
```

```
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos      stimlen
##     3.56660     -0.78272      0.08853     -0.78661     -0.09099
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1141 Residual
## Null Deviance:       1198
## Residual Deviance: 1038  AIC: 1150
## log likelihood:  -519.1757
## Nagelkerke R2:  0.2002386
## % pres/err predicted correctly:  -344.0054
## % of predictable range [ (model-null)/(1-null) ]:  0.1589889
## ***************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos     log_freq
##     2.99135     -0.78502      0.08362     -0.77175      0.09392
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1141 Residual
## Null Deviance:       1198
## Residual Deviance: 1037  AIC: 1150
## log likelihood:  -518.5631
## Nagelkerke R2:  0.2016728
## % pres/err predicted correctly:  -344.1117
## % of predictable range [ (model-null)/(1-null) ]:  0.1587296
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr     I(pos^2)          pos
##     2.98067     -0.78414      0.08064     -0.75785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1041  AIC: 1152
## log likelihood:  -520.5332
## Nagelkerke R2:  0.1970551
## % pres/err predicted correctly:  -345.4428
## % of predictable range [ (model-null)/(1-null) ]:  0.1554848
## ***************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
```

```
## (Intercept)
##        1.13
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1145 Residual
## Null Deviance:       1198
## Residual Deviance: 1198  AIC: 1317
## log likelihood:  -598.8578
## Nagelkerke R2:  0
## % pres/err predicted correctly:  -409.2269
## % of predictable range [ (model-null)/(1-null) ]:  0
## ***************************
```

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

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

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

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

kable(AICSummary)
```

Model	AIC	DeltaAIC	AICexp	AICwt	NagR2	(Intercept)	CumErr	I(pos^2)	pos	log_freq	stimlen
preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq	1149.45	0.0000000	1.0000000	0.3717840	0.2034537	1.7440348	- 0.7835935	0.0892651	- 0.7920621	0.0806025	- 0.0698845
preserved ~ CumErr + I(pos^2) + pos + stimlen	1149.86	0.4159649	0.8122220	0.3019707	0.2002386	1.5666601	- 0.7827196	0.0885295	- 0.7866131	NA	- 0.0909915
preserved ~ CumErr + I(pos^2) + pos + log_freq	1150.32	0.8760404	0.6453102	0.2399170	0.2016728	1.3991351	- 0.7850241	0.0836186	- 0.7717537	0.0939217	NA
preserved ~ CumErr + I(pos^2) + pos	1152.37	2.9203169	0.2321995	0.0863281	0.1970521	1.1980666	- 0.7841392	0.0806417	- 0.7578550	NA	NA
preserved ~ 1	1317.06	167.6189364	0.0000000	0.0000000	0.0000000	0.1129883	NA	NA	NA	NA	NA

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

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

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

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

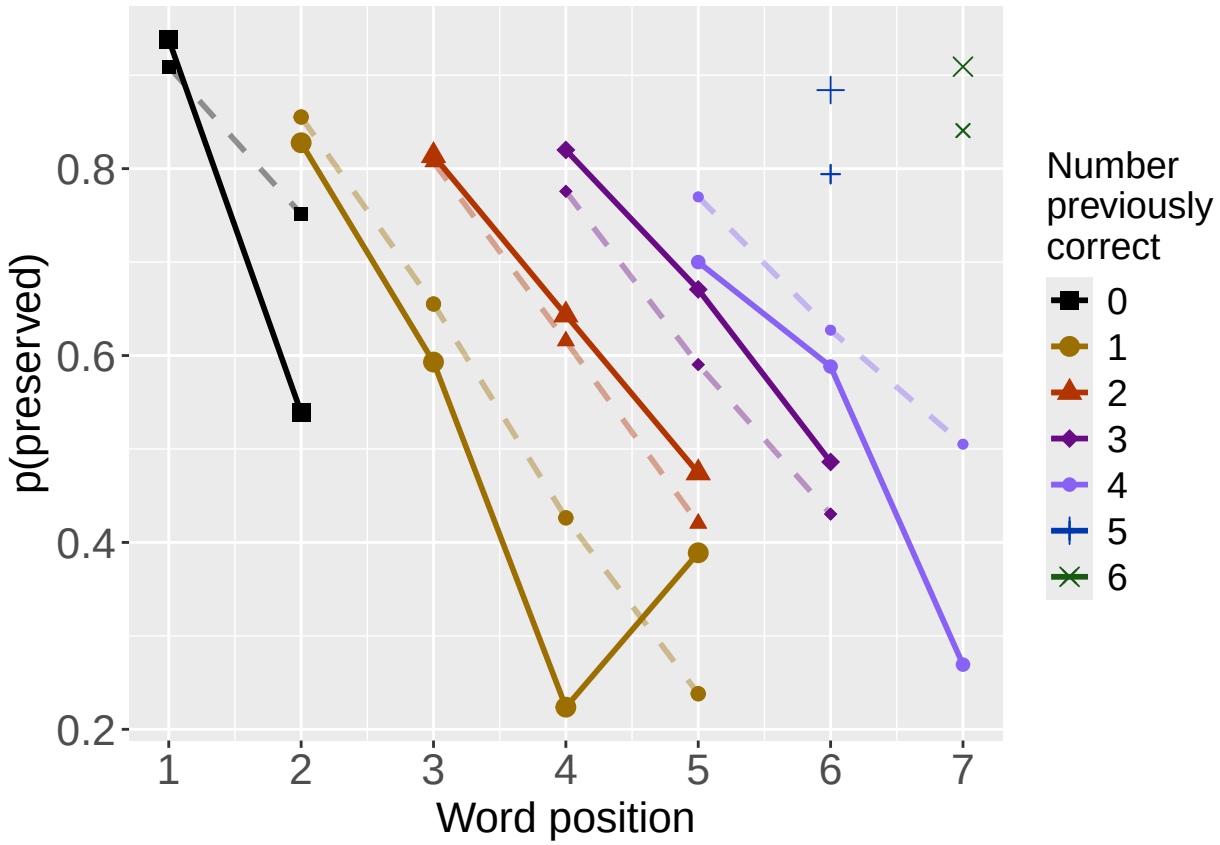
```
## Single term deletions
##
## Model:
## preserved ~ CumErr + I(pos^2) + pos + stimlen + log_freq
##          Df Deviance    AIC
## CumErr    1   1105.7 1217.5
## pos       1   1059.8 1171.7
## I(pos^2)  1   1055.8 1167.6
## log_freq  1   1038.3 1150.2
## <none>        1035.6 1149.5
## stimlen   1   1037.1 1149.0
```

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

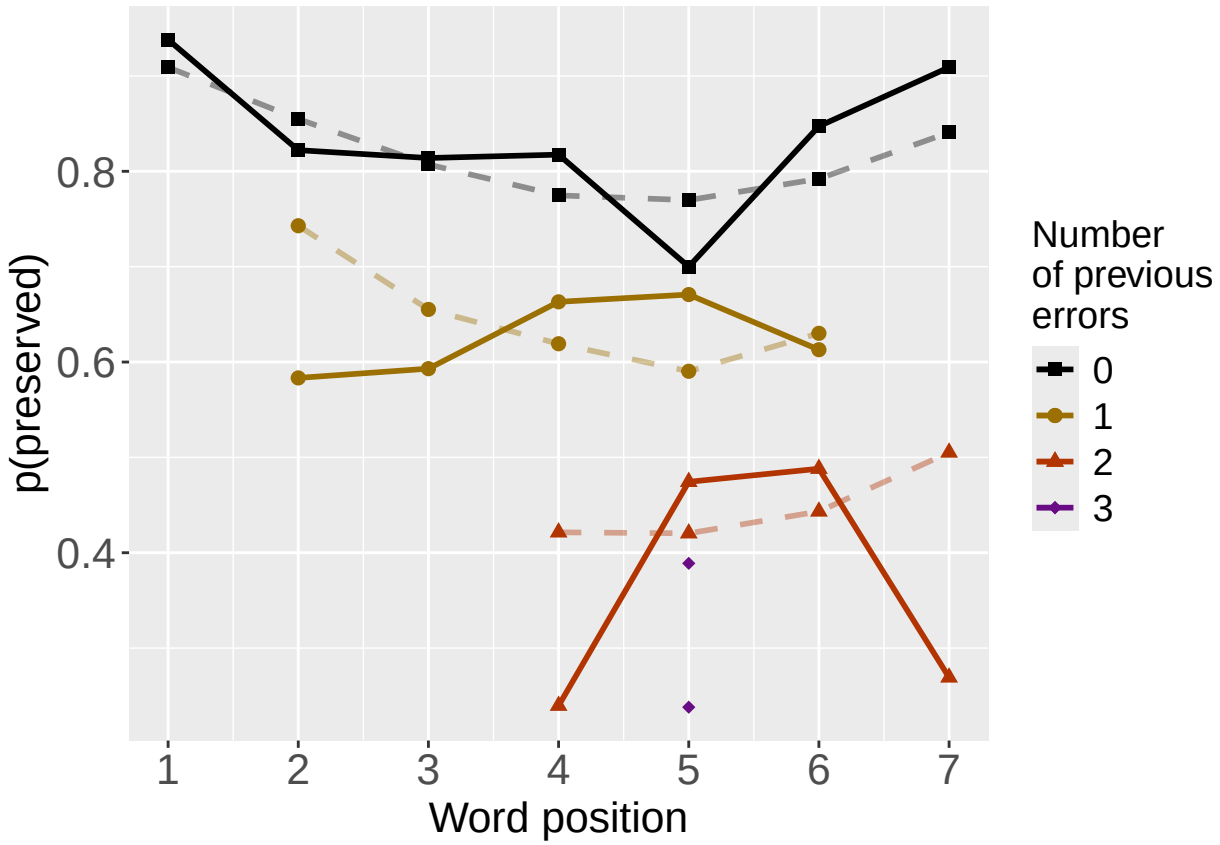
```
print(PrevCorPlot)
```




```
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!
```

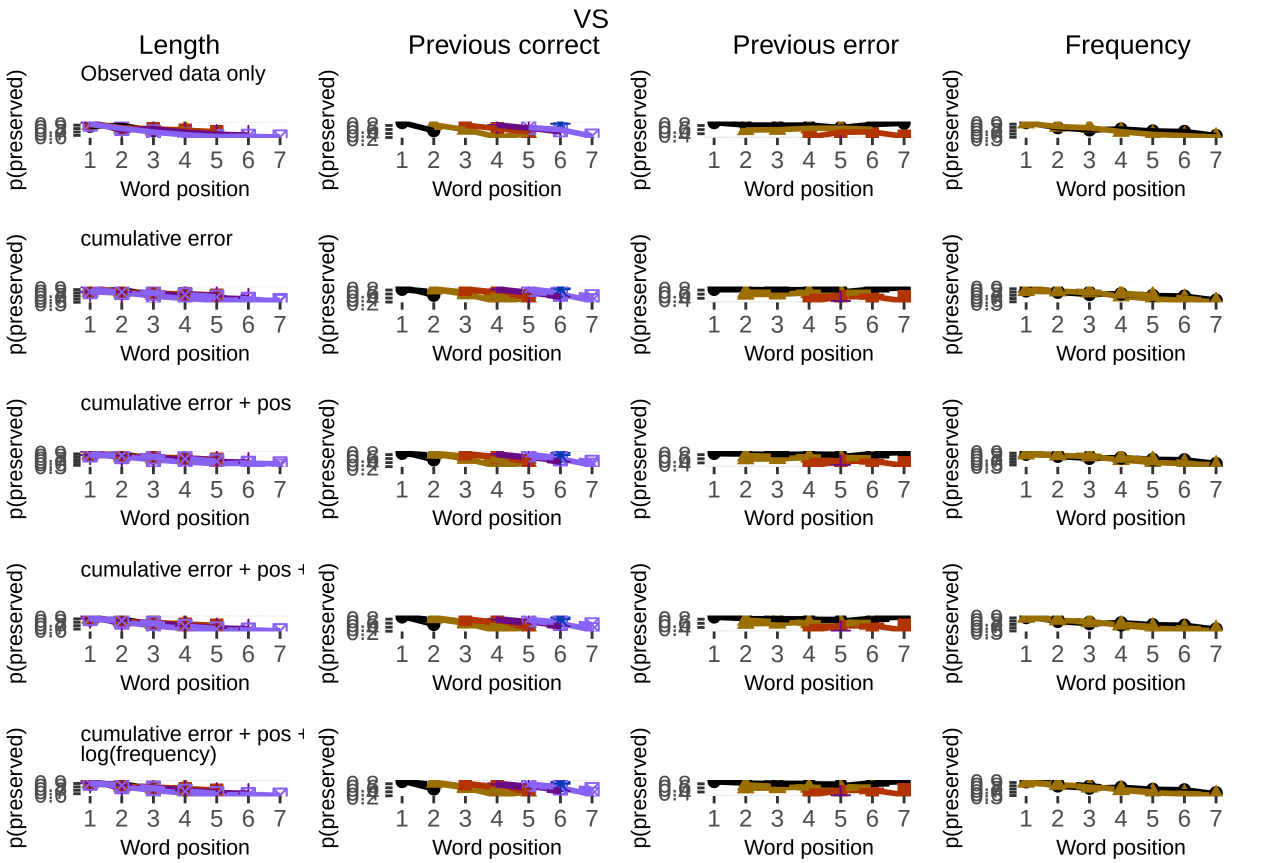
```
## ***************************
## model index:  1
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr
##      1.6111      -0.8785
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1144 Residual
## Null Deviance:       1198
## Residual Deviance: 1065  AIC: 1170
## log likelihood:  -532.3105
```

```
## Nagelkerke R2:  0.1691178
## % pres/err predicted correctly:  -351.3325
## % of predictable range [ (model-null)/(1-null) ]:  0.1411276
## **************************
## model index:  2
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos
##      1.9507      -0.7350      -0.1185
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1143 Residual
## Null Deviance:       1198
## Residual Deviance: 1059  AIC: 1167
## log likelihood:  -529.3611
## Nagelkerke R2:  0.1761683
## % pres/err predicted correctly:  -350.26
## % of predictable range [ (model-null)/(1-null) ]:  0.1437421
## **************************
## model index:  3
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)
##     2.98067     -0.78414     -0.75785      0.08064
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1142 Residual
## Null Deviance:       1198
## Residual Deviance: 1041  AIC: 1152
## log likelihood:  -520.5332
## Nagelkerke R2:  0.1970551
## % pres/err predicted correctly:  -345.4428
## % of predictable range [ (model-null)/(1-null) ]:  0.1554848
## **************************
## model index:  4
##
## Call:  glm(formula = as.formula(ModelEquations[Index]), family = "binomial",
##     data = PosDat)
##
## Coefficients:
## (Intercept)       CumErr          pos     I(pos^2)     log_freq
##     2.99135     -0.78502     -0.77175      0.08362      0.09392
##
## Degrees of Freedom: 1145 Total (i.e. Null);  1141 Residual
## Null Deviance:       1198
## Residual Deviance: 1037  AIC: 1150
## log likelihood:  -518.5631
## Nagelkerke R2:  0.2016728
## % pres/err predicted correctly:  -344.1117
## % of predictable range [ (model-null)/(1-null) ]:  0.1587296
```

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

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

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

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

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

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

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

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

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

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

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

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




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

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

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
## Warning in eval(family$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	log_freq
McFadden	0.0759200	0.0208194	0.0284821	0.0068281	0.0030148
SquaredCorrelation	0.0796437	0.0224921	0.0306337	0.0075654	0.0032417
Nagelkerke	0.1166804	0.0329516	0.0448792	0.0110835	0.0047492

	CumErr	I(posˆ2)	pos	stimlen	log_freq
Estrella	0.0860661	0.0236940	0.0323968	0.0077976	0.0034283

```
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) + log_freq CumErr + pos + I(pos^2) + log_freq 1037.126
## CumErr + pos + I(pos^2)                       CumErr + pos + I(pos^2) 1041.066
## CumErr + pos                                            CumErr + pos 1058.722
## CumErr                                                        CumErr 1064.621
## null                                                            null 1197.716
##                                    deviance_explained percent_explained
## CumErr + pos + I(pos^2) + log_freq           160.5894          13.40798
## CumErr + pos + I(pos^2)                      156.6492          13.07899
## CumErr + pos                                 138.9935          11.60488
## CumErr                                       133.0946          11.11237
## null                                           0.0000           0.00000
##                                    percent_of_explained_deviance increment_in_explained
## CumErr + pos + I(pos^2) + log_freq                     100.00000               2.453622
## CumErr + pos + I(pos^2)                                 97.54638              10.994282
## CumErr + pos                                            86.55210               3.673309
## CumErr                                                  82.87879              82.878788
## null                                                          NA               0.000000
```

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

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

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

	deviance	deviance_explained
CumErr + pos + I(pos^2) + log_freq	1037.126	160.5894
CumErr + pos + I(pos^2)	1041.066	156.6492
CumErr + pos	1058.722	138.9935
CumErr	1064.621	133.0946
null	1197.716	0.0000

	percent_explained	percent_of_explained_deviance	increment_in_explained
CumErr + pos + I(pos^2) + log_freq	13.40798	100.00000	2.453622
CumErr + pos + I(pos^2)	13.07899	97.54638	10.994282
CumErr + pos	11.60488	86.55210	3.673309
CumErr	11.11237	82.87879	82.878788
null	0.00000	NA	0.000000

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

```
##           Nagelkerke
## CumErr    0.55471232
## I(pos^2)  0.15665602
## pos       0.21336120
## stimlen   0.05269236
## log_freq  0.02257810
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

model	p_accounted_for	model_deviance
preserved ~ CumErr+pos	0.8486780	1058.722
preserved ~ CumErr	0.8573515	1064.621
preserved ~ CumErr+pos+I(pos^2)	0.9090574	1041.066
preserved ~ CumErr+pos+I(pos^2)+log_freq	0.9103233	1037.126

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

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

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

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



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

```
##                                     model p_accounted_for model_deviance diff_CumErr+pos
## 1                   preserved ~ CumErr+pos       0.8486780       1058.722     0.000000000
## 2                       preserved ~ CumErr       0.8573515       1064.621     0.008673562
## 3          preserved ~ CumErr+pos+I(pos^2)       0.9090574       1041.066     0.060379447
## 4 preserved ~ CumErr+pos+I(pos^2)+log_freq       0.9103233       1037.126     0.061645322
##    diff_CumErr diff_CumErr+pos+I(pos^2) diff_CumErr+pos+I(pos^2)+log_freq
## 1 -0.008673562             -0.060379447                      -0.061645322
## 2  0.000000000             -0.051705886                      -0.052971760
## 3  0.051705886              0.000000000                      -0.001265875
## 4  0.052971760              0.001265875                       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)
```

model	diff_CumErr+pos	diff_CumErr	diff_CumErr+pos+I(pos^2)
preserved ~ CumErr+pos	0.0000000	-0.0086736	-0.0603794
preserved ~ CumErr	0.0086736	0.0000000	-0.0517059
preserved ~ CumErr+pos+I(pos^2)	0.0603794	0.0517059	0.0000000
preserved ~ CumErr+pos+I(pos^2)+log_freq	0.0616453	0.0529718	0.0012659

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